Oracle® Database Express Edition

Application Express User's Guide

Release 2.1

B25309-01

February 2006

ORACLE®

Oracle Database Express Edition Application Express User’s Guide, Release 2.1

B25309-01

Primary Author: Terri Winters

Contributors: Carl Backstrom, Christina Cho, Michael Hichwa, Christopher Jones, Joel Kallman, Sharon Kennedy, Syme Kutz, Sergio Leunissen, Raj Mattamal, Tyler Muth, Kris Rice, Marc Sewtz, Scott Spadafore, Scott Spendolini, Jason Straub, Simon Watt, and Steve Fogel

Contents

3 Application Builder Concepts

Preface

Oracle Database Express Edition Application Express User's Guide describes how to use the Oracle Application Express client within Oracle Database Express Edition (Oracle Database XE).

This preface contains these topics:

- Documentation Topics
- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Documentation Topics

This document contains the following chapters:

Title	Description
Quick Start	Offers a quick introduction to using the Oracle Application Express client within Oracle Database XE.
Running a Demonstration Application	Describes how to run and modify the demonstration applications that install with Application Builder.
Application Builder Concepts	Provides basic conceptual information about Application Builder. Use Application Builder to assemble an HTML interface (or application) on top of database objects such as tables and procedures.
Using Application Builder	Provides important background information about using Application Builder to build dynamically rendered applications.
Building an Application	Describes how to use Application Builder to build an application and application components.
Controlling Page Layout and User Interface	Describes different approaches to customizing an application's user interface and page layout including customizing regions, editing item attributes, customizing templates, and incorporating cascading style sheets and images.
Adding Navigation	Describes how to implement navigation in your application using different types of navigation controls, including navigation bar entries, tabs, breadcrumbs, lists, and trees.

Title	Description
Understanding Application Administration	Explains how to mange your application development environment, including managing application services, session state, user preferences, log files, application models, Application Express end users, application development activity, and explains how to send email from an application.
Managing User Interface Defaults	Describes how to use user interface defaults to assign default user interface properties to a table, column, or view.
Debugging an Application	Describes approaches to debugging your Application Builder application, including viewing Debug Mode, enabling SQL tracing, viewing page reports, and how to manually remove a control or a component to isolate a problem.
Deploying an Application	Explains how to deploy an application built within Application Builder.
Managing Application Security	Describes how to provide security for an Application Builder application by utilizing cross-site scripting protection, session state protection, authentication, and authorization.
Advanced Programming Techniques	Provides information about advanced programming techniques including establishing database links, using collections, running background SQL, utilizing Web services, and managing user preferences.
Managing Application Globalization	Explains how to translate an application built-in Application Builder.
Oracle Application Express APIs	Describes the APIs available in Oracle Application Express.
Managing Database Objects with Object Browser	Describes how to use Object Browser to browse, create, and edit objects in an Oracle Database Express Edition database.
Building Queries with Query Builder	Explains how to use Query Builder's graphical user interface to search and filter database objects, select objects and columns, create relationships between objects, view formatted query results, and save queries.
Using SQL Scripts	Provides information on how to use SQL Scripts to create, edit, view, run, and delete script files.
Using SQL Commands	Explains on how to use SQL Commands to create, edit, view, run, and delete SQL commands.
Using SQL Utilities	Describes how to use SQL Utilities to load and unload data from the Oracle Database XE database, generate DDL, view object reports, and restore dropped database objects.
Administering the Database	Describes how to administer the Oracle Database XE database.

Audience

Oracle Database Express Edition Application Express User's Guide is intended for application developers who are building database-centric Web applications using the Oracle Application Express client within Oracle Database XE. Oracle Database XE includes an intuitive browser-based user interface. This browser-based user interface is built within the Oracle Application Express client. The guide describes how to use the

Oracle Database XE browser-based user interface to administer the database, run scripts and queries, and build Web-based application.

To use this guide, you need to have a general understanding of relational database concepts as well as an understanding of the operating system environment under which you are running the Oracle Database XE.

> **See Also:** "Introducing Oracle Database XE" in *Oracle Database Express Edition 2 Day DBA Guide*

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Related Documents

For more information, see these Oracle resources:

- *Oracle Database Express Edition Installation Guide for Linux*
- *Oracle Database Express Edition Installation Guide for Microsoft Windows*
- *Oracle Database Express Edition 2 Day DBA Guide*
- *Oracle Database Express Edition 2 Day Developer Guide*
- *Oracle Database Express Edition 2 Day Plus Application Express Developer Guide*

For the most recent version of the Oracle Database Express Edition documentation, see the Oracle Database XE online library:

`http://www.oracle.com/technology/xe/documentation`

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Part I

Getting Started with Oracle Application Express

Part I provides an introduction to Oracle Application Express. These chapters introduce you to basic Oracle Application Express concepts.

Part I contains the following chapters:

1 Quick Start

This section offers a quick introduction to using the Oracle Application Express client within Oracle Database Express Edition (Oracle Database XE). It is assumed you have already completed the installation process.

This section contains the following topics:

- What is Oracle Application Express?
- About the Database Home Page
- Navigation Alternatives
- Using Online Help

See Also: "Running a Demonstration Application" on page 2-1 and "Application Builder Concepts" on page 3-1

What is Oracle Application Express?

Oracle Database XE includes an intuitive browser-based user interface built within the Oracle Application Express client.

What is Oracle Application Express? Oracle Application Express is a hosted declarative development environment for developing and deploying database-centric Web applications. Thanks to built-in features such as design themes, navigational controls, form handlers, and flexible reports, Oracle Application Express accelerates the application development process.

The Application Express engine renders applications in real time from data stored in database tables. When you create or extend an application, Oracle Application Express creates or modifies metadata stored in database tables. When the application is run, the Application Express engine then reads the metadata and displays the application.

To provide stateful behavior within an application, Oracle Application Express transparently manages session state in the database. Application developers can get and set session state using simple substitutions as well as standard SQL bind variable syntax.

About the Database Home Page

Oracle Database XE includes an intuitive browser-based user interface for administering the database, running scripts and queries, building Web-based applications. The starting point for this interface is the Database Home Page.

The Database Home Page features the following components:

- **Administration**. Use Administration to manage database user accounts, manage storage, monitor database activity, and view configuration information. See "Administering the Database" on page 21-1.
- **Object Browser**. View, create, modify, browse, and drop database objects. Use the PL/SQL editor to edit and compile packages, procedures, functions, and triggers while taking advantage of error reporting. See "Managing Database Objects with Object Browser" on page 16-1.
- **SQL**. Click **SQL** to access the following SQL tools:
  - **SQL Commands**. Run SQL commands and anonymous PL/SQL, scripts, and saved queries. See "Using SQL Commands" on page 19-1.
  - **SQL Scripts**. Use SQL Scripts to create, edit, view, run, and delete script files. You can also upload and download scripts from your local file system. See "Using SQL Scripts" on page 18-1.
  - **Query Builder**. Use Query Builder's graphical user interface to search and filter database objects, select objects and columns, create relationships between objects, view formatted query results, and save queries with little or no SQL knowledge. See "Building Queries with Query Builder" on page 17-1.
- **Utilities**. Use Utilities to import and export data from the database, generate DDL, view object reports, and restore dropped database objects. See "Using SQL Utilities" on page 20-1.
- **Application Builder**. Use Application Builder to assemble an HTML interface (or application) on top of database objects such as tables and procedures. See "Application Builder Concepts" on page 3-1 and "Using Application Builder" on page 4-1.

See Also: "Introducing Oracle Database XE" in *Oracle Database Express Edition 2 Day DBA Guide*

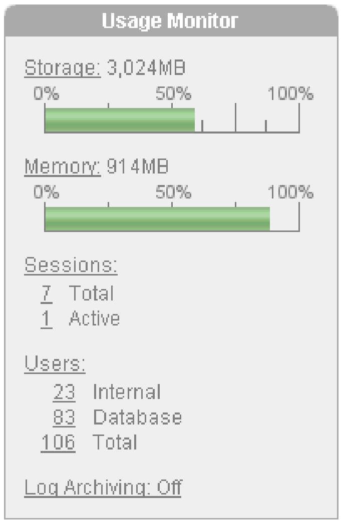
About the Links List

The Links list displays on the right side of Database Home Page. Use this list to access the following Oracle Database XE resources:

- **License Agreement** displays the Oracle Database Express Edition license agreement.
- **Getting Started** displays *Oracle Database Express Edition Getting Started Guide*. Use this guide to get up and running using Oracle Database XE by creating a simple application.
- **Learn More** displays a list of suggested titles from the Oracle Database XE Documentation Library.
- **Documentation** links to the Oracle Database Express Edition Documentation Library.
- **Forum Registration** links to the Oracle Database Express Edition Registration page. Completing registration entitles you for membership in the Oracle Database 10g Express Edition Discussion Forum.
- **Discussion Forum** links to Oracle Database Express Edition Discussion Forum. Use this forum to post questions about using Oracle Database XE or to help other members solve answers their questions. Access to the forum require registration.
- **Product Page** links the Oracle Database Express Edition home page on Oracle Technology Network (OTN).

About the Usage Monitor

Oracle Database XE is limited to 4 GB of user data and 1 GB of memory. You can use the Usage Monitor on the right side of the Database Home Page to monitor the storage space available, memory allocation, the current number of database sessions, and users, and database log archive status.



To view more detailed information, click the following links on the Usage Monitor:

- **Storage**. See "Monitoring Database Storage" on page 21-3.
- **Memory**. See "Monitoring Memory" on page 21-8.
- **Sessions**. See "Sessions Report" on page 21-16.
- **Users**. See "Viewing Database Users" on page 21-11.
- **Log Archiving**. See "Viewing Database Logging Status" on page 21-6.

Customizing the Appearance of the Database Home Page

You can control what regions display on the Database Home Page by clicking the **Customize** link in the upper right corner of the page. By default, both the Links list and Usage Monitor appear.

To hide the Links list and Usage Monitor:

1. Click the **Customize** link in the upper right corner of the Database Home Page.
2. Deselect the region you wish hide and click **Apply Changes**.

Navigation Alternatives

This section describes alternative methods for navigating between pages in the Oracle Database XE browser-based user interface.

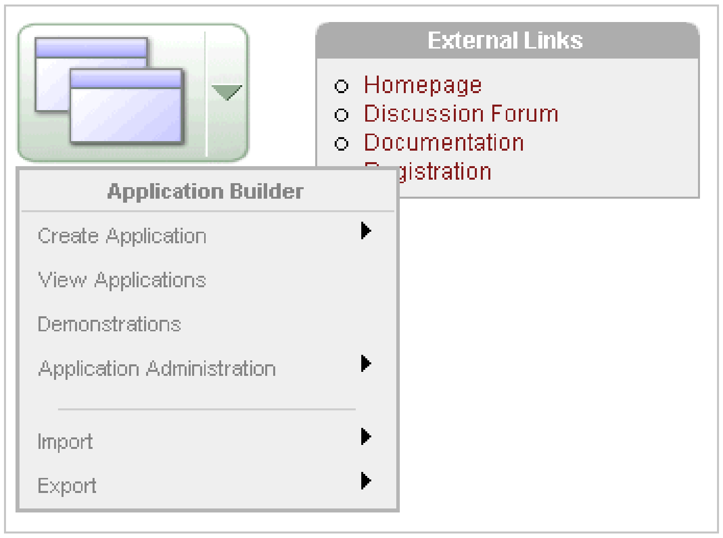
Navigating Using Icons or Drop Down Menus

You can move between pages in user interface by clicking large graphical icons. When using these icons, you have two navigation options:

- **Primary Navigation (drill-down).** Click the large icons in the center of the page to drill-down to the appropriate page.

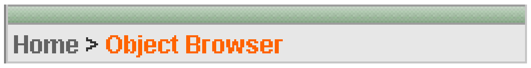
- **Secondary Navigation (drop down menus).** Click the down arrow on the right side of the icon to view a drop down menu. Select an option from the menu.

Note: For the purposes of consistency, this document uses the primary navigation path (or drill-down approach) when explaining navigation.



Navigating Using Breadcrumbs

Breadcrumbs (also called locator links) appear at the top of every page within the Oracle Database XE browser-based user interface. Each breadcrumb entry indicates where the current page is relative to other pages within the user interface. You can use breadcrumbs to instantly link to a previous page. For example, clicking on **Home** takes you to the Database Home Page.



See Also: "Introducing Oracle Database XE" in *Oracle Database Express Edition 2 Day DBA Guide*

Using Online Help

The Oracle Database XE browser-based user interface features three types of online help:

- Procedural Online Help
- Page-Level Help
- Field Level Help

See Also: "Introducing Oracle Database XE" in *Oracle Database Express Edition 2 Day DBA Guide*

Procedural Online Help

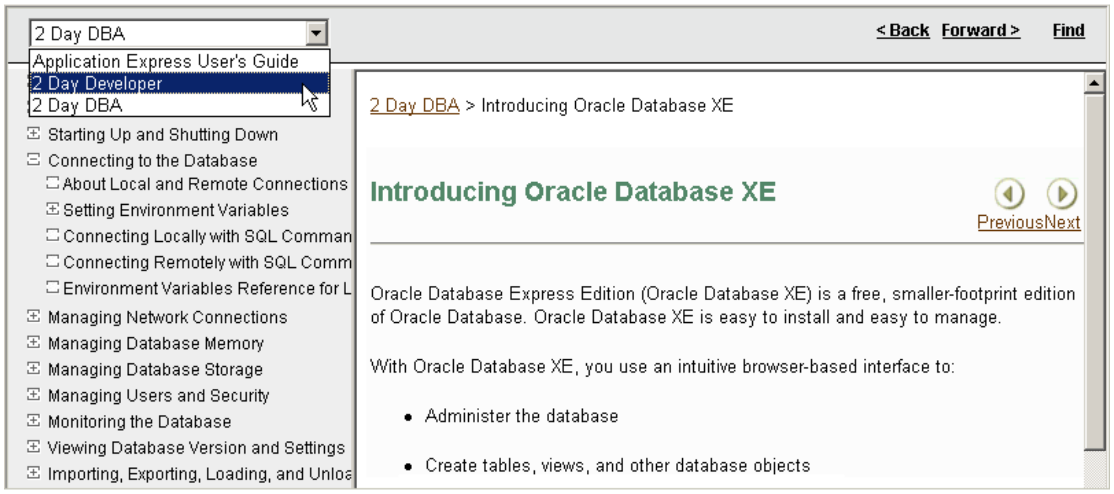
You can access an HTML-based online Help system by clicking the Help icon in the upper right corner of the window.

Help

Procedural help includes content from the following sources:

- *Oracle Database Express Edition 2 Day DBA Guide* discusses common day-to-day administrative tasks.
- *Oracle Database Express Edition 2 Day Developer Guide* explains how to take advantage of the Oracle Database Express Edition development environment.
- *Oracle Database Express Edition Application Express User's Guide* describes how to use the Oracle Database XE browser-based user interface to administer the database, run scripts and queries, and build Web-based applications. See "Documentation Topics" on page -xxvii.

When you click the Help icon, a help topic that describes the current page appears. To view the table of contents of another help set, select it from the list in the upper left side of the window.


2 Day DBA
Application Express User's Guide
2 Day Developer
2 Day DBA
< Back
Forward >
Find
Starting Up and Shutting Down
Connecting to the Database
About Local and Remote Connections
Setting Environment Variables
Connecting Locally with SQL Comman
Connecting Remotely with SQL Comm
Environment Variables Reference for L
Managing Network Connections
Managing Database Memory
Managing Database Storage
Managing Users and Security
Monitoring the Database
Viewing Database Version and Settings
Importing, Exporting, Loading, and Unloa
2 Day DBA > Introducing Oracle Database XE
Previous
Next

Introducing Oracle Database XE

Oracle Database Express Edition (Oracle Database XE) is a free, smaller-footprint edition of Oracle Database. Oracle Database XE is easy to install and easy to manage.

With Oracle Database XE, you use an intuitive browser-based interface to:

- Administer the database
- Create tables, views, and other database objects

You can browse through help topics by:

- Expanding and collapsing the table of contents. To view a topic, simply select it.
- Clicking the breadcrumb links at the top of each help topic.
- Clicking the **Previous** and **Next** buttons within a topic. Click these buttons to access the previous and next help topic within the structure of the help set.

The top of the window features a gray bar. Click **Back** and **Forward** to return to a previously viewed page. These controls work similarly to the Back and Forward controls in a Web browser.

Click **Find** to perform a keyword search of the entire help system. When the search field appears, enter a case insensitive query in the field provided and click **Find**. To search for an exact phrase, enclose the phrase in double quotation marks.

Page-Level Help

Many pages with in the Oracle Database XE browser-based user interface include page-level Help. Page-level Help displays in a text box on the right side of the page and offers a brief description of the page functionality.

Field Level Help

Most select lists, check boxes, and fields within the Oracle Database XE browser-based user interface include item help. When item help is available, the item label changes to red when you pass your cursor over it. Click the item label to display a description in a separate window.

2 Running a Demonstration Application

This section describes how to run and modify the demonstration applications that install with Application Builder. Running and analyzing how an application works is an effective way to better understand how you can use Application Builder to build your own applications.

This section contains the following topics:

See Also:

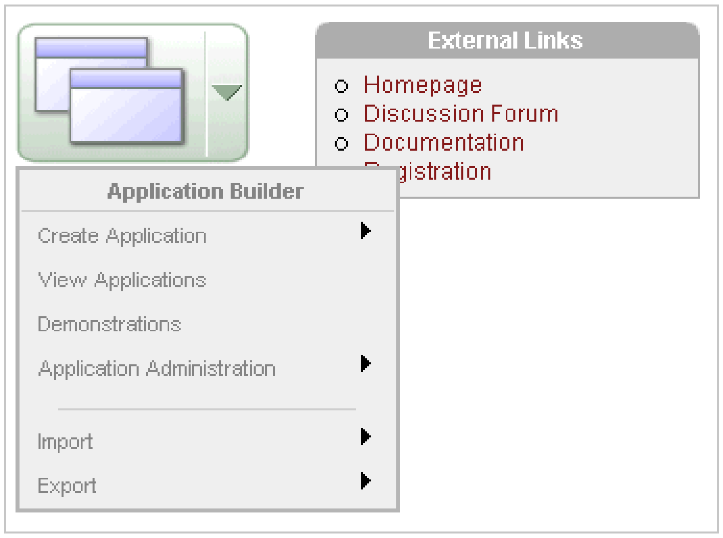
- "Quick Start" on page 1-1
- "Application Builder Concepts" on page 3-1
- "Using Application Builder" on page 4-1

Installing a Demonstration Application

Application Builder includes a number of demonstration applications you can install. Use these applications to learn more about the different types of functionality you can include in your applications.

To install the demonstration applications:

1. From the Database Home Page, click the down arrow on the right side of the Application Builder icon.
2. From the menu, select **Demonstrations**.



The Demonstration Applications page appears, displaying the following applications:

- *Sample Application* offers a working demonstration that highlights basic design concepts.
- *Collection Showcase* demonstrates shopping cart concepts.
- *Web Services* serves as an example of how you can use Web Services.

3. To install a demonstration application, scroll down to the application you want to install and click **Install**.

   Installed appears in the Status column.

4. To edit an installed demonstration application, click **Edit**.
5. To run an installed demonstration application, click **Run**.
6. To reinstall a demonstration application, click **Re-Install**.

Note: Alternatively, you can access the demonstration applications page running the Create Application Wizard.

See Also: "About Demonstration Applications" on page 5-6 and "Running a Demonstration Application" on page 2-2

Running a Demonstration Application

Application Builder installs with a number of demonstration applications. Once you have installed a demonstration application, you can run it from the Demonstration Applications page or from the Application Builder home page.

Topics in this section include:

- Running an Application from Demonstration Applications
- Running an Application from the Application Home Page

See Also: "Installing a Demonstration Application" on page 2-1 and "Running a Page or Application" on page 5-11

Running an Application from Demonstration Applications

The simplest way to run a demonstration application is to navigate to the Demonstration Applications page.

To run a demonstration application from the Demonstration Applications page:

1. From the Database Home Page, click the down arrow on the right side of the Application Builder icon.
2. From the menu, select **Demonstrations**.
3. On the Demonstration Applications page, locate the application you want to run.
4. In the Action column, click **Run**.
5. Enter the appropriate username and password and click **Login**.

   For Sample Application, enter either `demo` or `admin` for the user name and enter the current database user name in lowercase letters for the password.

   For other demonstration applications, enter your database user name and password.

Running an Application from the Application Home Page

Once you have installed a demonstration application, you can run it from the Application Builder home page.

To run a demonstration application from the Application Builder home page:

1. Log in to the Database Home Page user interface.

   The Database Home Page appears.
2. Click the **Application Builder** icon.
3. Select an application.

   The Application appears.
4. Click the **Run Application** icon.
5. Enter the appropriate username and password and click **Login**.

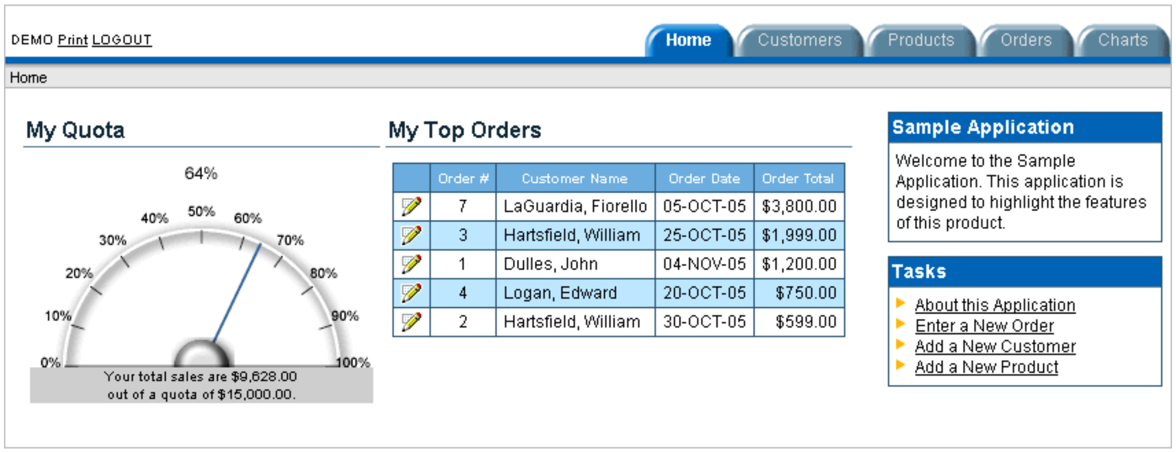
   For the demonstration application *Sample Application*, enter either `demo` or `admin` for the user name and enter your database user name in lowercase letters for the password.

   For other demonstration applications, enter your database user name and password.

Understanding Sample Application

Each demonstration application shows a different set of features. This section describes the demonstration application, *Sample Application*.

Sample Application shows an easy-to-use interface for viewing, updating, and searching order and customer information for electronic and computer products. Users can navigate among the pages using the Home, Customers, Products, Orders, and Charts tabs.



My Quota

Your total sales are $9,628.00 out of a quota of $15,000.00.

My Top Orders

	Order #	Customer Name	Order Date	Order Total
✎	7	LaGuardia, Fiorello	05-OCT-05	$3,800.00
✎	3	Hartsfield, William	25-OCT-05	$1,999.00
✎	1	Dulles, John	04-NOV-05	$1,200.00
✎	4	Logan, Edward	20-OCT-05	$750.00
✎	2	Hartsfield, William	30-OCT-05	$599.00

Sample Application

Welcome to the Sample Application. This application is designed to highlight the features of this product.

Tasks

- About this Application
- Enter a New Order
- Add a New Customer
- Add a New Product

Sample Application demonstrates the following functionality:

- Examples of ways to display summary information, including a dial chart and summary reports
- Reports for viewing, updating, and adding customers, products, and orders
- A Calendar report
- SVG charts available in Oracle Application Express including cluster bar, pie chart, and stacked bar
- Printer friendly mode

The following sections describe specific functionality available on each page.

See Also: "What Is a Page?" on page 3-1

About the Home Page

The Home page contains four regions:

- My Quota
- My Top Orders
- Sample Application
- Tasks

My Quota demonstrates the use of a new SVG chart called a Dial Chart. This chart displays a value based on an underlying SQL statement. Although not demonstrated in this example, you can enable an asynchronous refresh by editing the attributes of any SVG chart.

My Top Orders is a simple report based on a SQL query. This report displays a subset of the information that appears on the Orders page. Users can link to order details by selecting the **Edit** icon.

Sample Application is a simple HTML region that displays static text. You can create this type of region to display explanatory information to users.

Tasks contains an list with links to other pages in *Sample Application*. Links available on the Home page Tasks list include:

- **About this Application** links to an informational page that describes this application.

- **Enter a New Order** links to a wizard for creating a new order.
- **Add a New Customer** links to a form for entering new customer information.
- **Add a New Product** links to a form for adding new products.

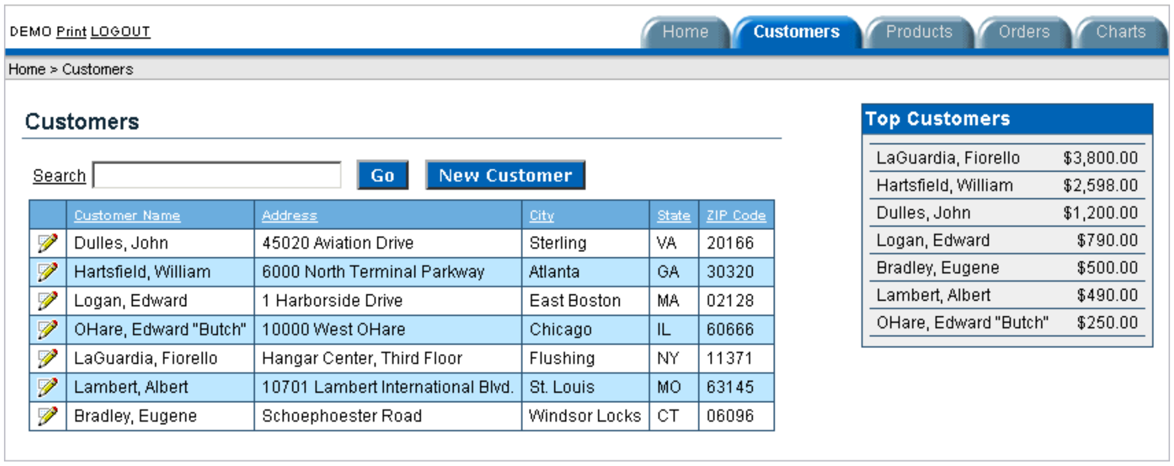
See Also:

- "Creating Charts" on page 5-39
- "Creating a Report Using a Wizard" on page 5-17
- "Creating a Region" on page 7-2
- "Creating Lists" on page 6-13

About the Customers Page

The Customers page enables users to view and edit customer information. The Customers page consists of two main regions:

- Customers
- Top Customers



	Customer Name	Address	City	State	ZIP Code
✎	Dulles, John	45020 Aviation Drive	Sterling	VA	20166
✎	Hartsfield, William	6000 North Terminal Parkway	Atlanta	GA	30320
✎	Logan, Edward	1 Harborside Drive	East Boston	MA	02128
✎	OHare, Edward "Butch"	10000 West OHare	Chicago	IL	60666
✎	LaGuardia, Fiorello	Hangar Center, Third Floor	Flushing	NY	11371
✎	Lambert, Albert	10701 Lambert International Blvd.	St. Louis	MO	63145
✎	Bradley, Eugene	Schoephoester Road	Windsor Locks	CT	06096

LaGuardia, Fiorello	$3,800.00
Hartsfield, William	$2,598.00
Dulles, John	$1,200.00
Logan, Edward	$790.00
Bradley, Eugene	$500.00
Lambert, Albert	$490.00
OHare, Edward "Butch"	$250.00

Customers is an updatable report for tracking customer information. This region is also based on a SQL query. To search for a customer, enter a customer name in the Search field and click **Go**. To sort by customer name, click the column heading. A Sort icon appears to the right of the heading, Customer Name. To update existing customer information, click the **Edit** icon.

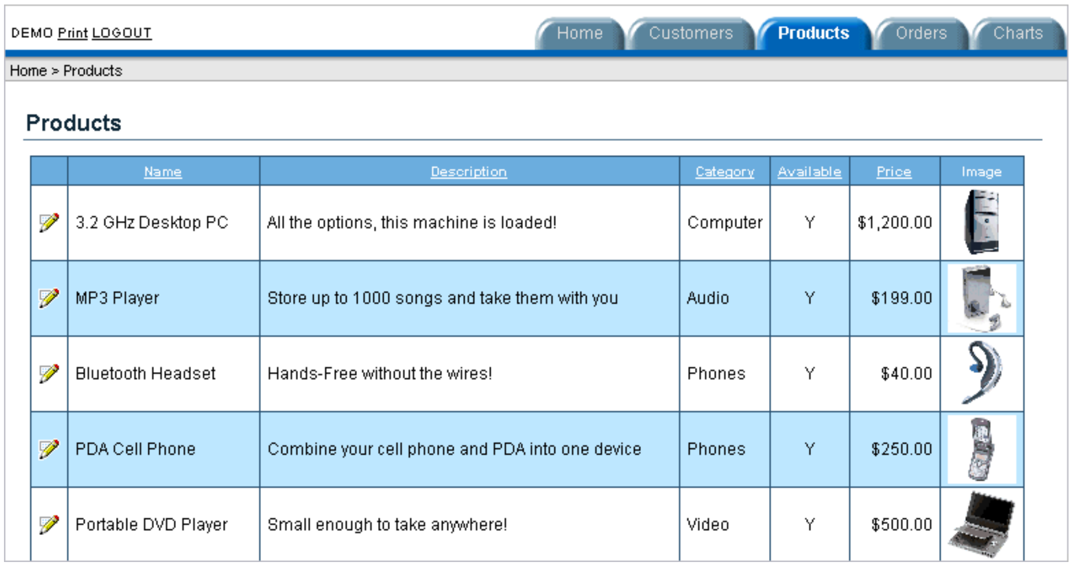
Top Customers ranks customers by order amount. This report is based on a SQL query that returns top customers based on their orders.

See Also: "Creating Reports" on page 5-17

About the Products Page

The Products page enables users to view and edit product information. The Products page consists of two main regions:

- Products
- Top 10 Products

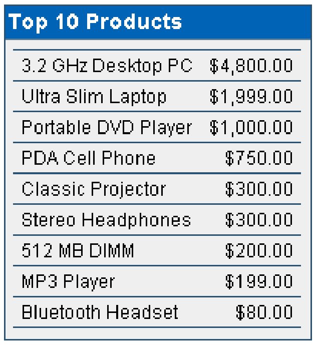


Products

	Name	Description	Category	Available	Price	Image
	3.2 GHz Desktop PC	All the options, this machine is loaded!	Computer	Y	$1,200.00	
	MP3 Player	Store up to 1000 songs and take them with you	Audio	Y	$199.00	
	Bluetooth Headset	Hands-Free without the wires!	Phones	Y	$40.00	
	PDA Cell Phone	Combine your cell phone and PDA into one device	Phones	Y	$250.00	
	Portable DVD Player	Small enough to take anywhere!	Video	Y	$500.00	

Products displays an updatable report for tracking product information. This region is based on a SQL query that uses a custom function for displaying images stored in the database. To sort by product category, click the column heading. A Sort icon appears to the right of the heading. To edit a product description, click the **Edit** icon. To add a new product, click the **Create Product** button at the bottom of the page. Users can export the data in the Products report to a spreadsheet, by clicking **Export to Spreadsheet**.

Top 10 Products is also a SQL report. This report outlines the top ten products based on quantities sold.



Top 10 Products

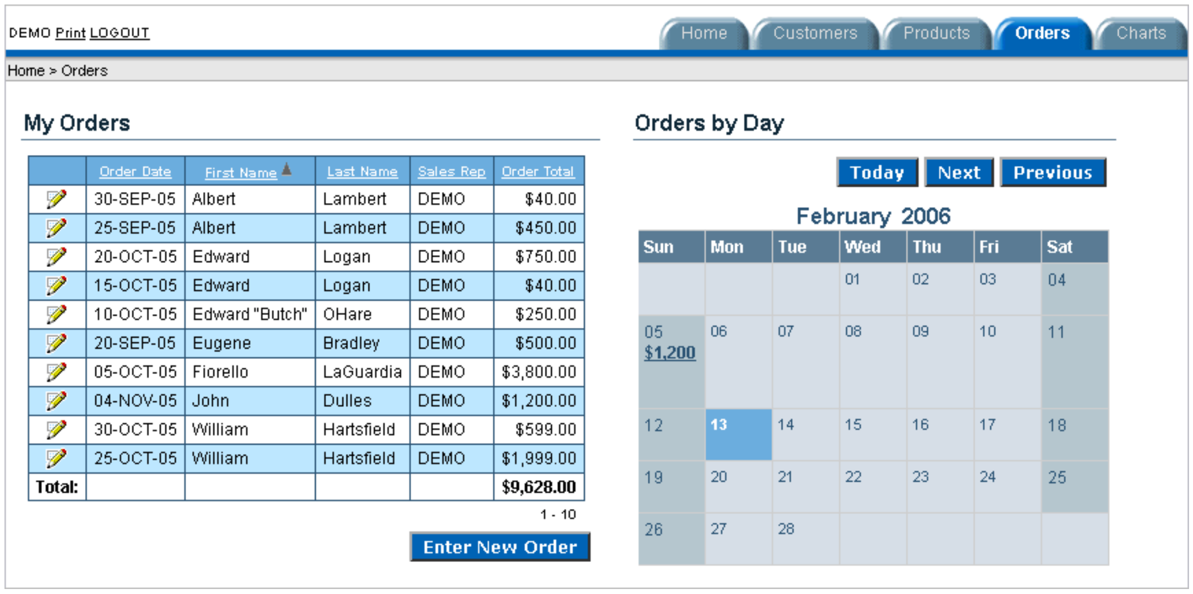
3.2 GHz Desktop PC	$4,800.00
Ultra Slim Laptop	$1,999.00
Portable DVD Player	$1,000.00
PDA Cell Phone	$750.00
Classic Projector	$300.00
Stereo Headphones	$300.00
512 MB DIMM	$200.00
MP3 Player	$199.00
Bluetooth Headset	$80.00

See Also: "Creating Reports" on page 5-17

About the Orders Page

The Orders page enables users to view and edit customer orders. The Orders page contains two regions:

- My Orders
- Order by Day



My Orders

	Order Date	First Name	Last Name	Sales Rep	Order Total
✎	30-SEP-05	Albert	Lambert	DEMO	$40.00
✎	25-SEP-05	Albert	Lambert	DEMO	$450.00
✎	20-OCT-05	Edward	Logan	DEMO	$750.00
✎	15-OCT-05	Edward	Logan	DEMO	$40.00
✎	10-OCT-05	Edward "Butch"	OHare	DEMO	$250.00
✎	20-SEP-05	Eugene	Bradley	DEMO	$500.00
✎	05-OCT-05	Fiorello	LaGuardia	DEMO	$3,800.00
✎	04-NOV-05	John	Dulles	DEMO	$1,200.00
✎	30-OCT-05	William	Hartsfield	DEMO	$599.00
✎	25-OCT-05	William	Hartsfield	DEMO	$1,999.00
Total:					$9,628.00

1 - 10

Orders by Day

February 2006

Sun	Mon	Tue	Wed	Thu	Fri	Sat
			01	02	03	04
05 $1,200	06	07	08	09	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28				

My Orders is a wizard report which summarizes the current orders in the system. To sort a column, click the column heading. A Sort icon appears next to column heading. To edit an existing order, click the Edit icon. To add a new order, click the **Enter New Order** button.

Orders by Day is a Calendar report. This report displays the each order on the appropriate date in a calendar. Users can select a calendar entry to view order details.

> **See Also:** "Creating Calendars" on page 5-34 and "Creating Reports" on page 5-17

About the Charts Page

The Charts page illustrates three of the several types of SVG charts available in Application Builder: cluster bar, pie chart, and stacked bar. To view a chart, select a chart type.

> **See Also:** "Creating Charts" on page 5-39

About the Admin Page

The Admin page displays only if you log in to *Sample Application* using the user name `admin`. Sample Application makes use of a custom authentication scheme that stores user names and obfuscated passwords in a table. The Manage Users page enables you to manage additional users.

Note that this custom authentication scheme does not use any user names or passwords associated with Oracle Application Express application developers.

Viewing Pages in Printer Friendly Mode

Clicking **Print** in the upper right corner of the page displays the current page in Printer Friendly mode. When in Printer Friendly mode, the Application Express engine displays all text within the HTML form fields as text.

To enable your application to display in Printer Friendly mode, you need to create and then specify a Print Mode Page Template on the Edit Application Attributes page.

See Also: "Optimizing a Page for Printing" on page 7-42

Modifying a Demonstration Application

Once you understand the type of functionality available in a demonstration application, the next step is to learn more about how each page is constructed. You edit an application using Application Builder. Using Application Builder you can edit existing pages in an application, add pages to an application, or create entirely new applications.

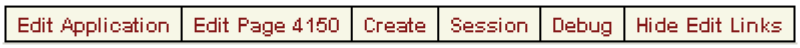
Topics in this section include:

- About the Developer Toolbar
- Editing a Demonstration Application

About the Developer Toolbar

The Developer toolbar is a quick way to edit the current application, the current running page, create a new page, control, or component, view session state, or turn edit links on or off.

See Also: "Using the Developer Toolbar" on page 4-18



The Developer toolbar consists of the following links:

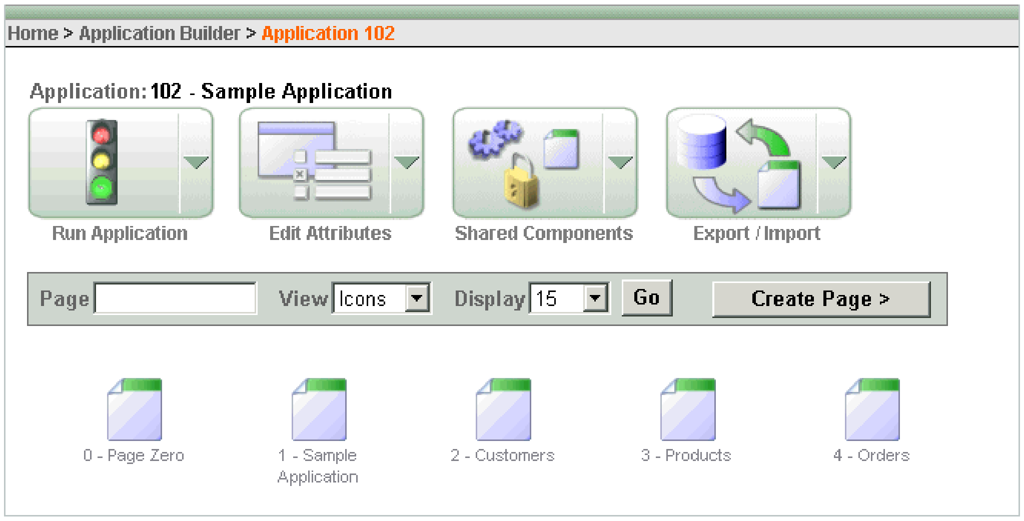
- **Edit Application** links you to the Application Builder home page. See "Application Builder Concepts" on page 3-1.
- **Edit Page** accesses the Page Definition for the current running page. See "About the Page Definition" on page 4-16.
- **Create** links to a wizard for creating a new page, region, page control (item, button, branch, computation, process, or validation), or a shared control (navigation bar icon, tab, list of values, list, or breadcrumb). See "Building an Application" on page 5-1.
- **Session** links you to session state information for the current page. See "Viewing Session State" on page 3-5.
- **Debug** toggles the page between Debug and No Debug mode. See "Accessing Debug Mode" on page 10-2.
- **Show Edit Links** toggles between **Show Edit Links** and **Hide Edit Links**. Clicking **Show Edit Links** displays edit links next to each object on the page that can be edited. Each edit link resembles two colons (::) and appears to the right of navigation bar items, tabs, region titles, buttons, and items. Clicking on the link displays another window in which to edit the object.

Editing a Demonstration Application

There are two ways to edit a demonstration application:

- From Demonstration Applications page, click **Edit** next to the desired application.
- If you are running an application, click **Edit Page** on the Developer toolbar.

The Application Builder appears. The application ID and application name display at the top of the page.



You can run the current application, edit application attributes, create shared components, export and import information, or create a new page by clicking one of the following:

- **Run Application** submits the pages in the current application to the Application Express engine to render viewable HTML.
- **Edit Attributes** displays the Edit Application Attributes page.
- **Shared Components** links to a new page for building shared application components and user interface controls.
- **Export/Import** links you to the Export/Import Wizard.
- **Create Page** links to a wizard for creating a new page.

The pages that make up the application display at the bottom of the page. To access a specific page, select it. To search for a specific page, enter a case insensitive query for the page title or page ID in the Page field and click **Go**.

> **See Also:** "About the Application Home Page" on page 4-1 and "About the Page Definition" on page 4-16

Viewing Underlying Database Objects

The Application Express engine renders applications in real time based on data stored in database tables. You can view the database objects for any demonstration application in Object Browser.

> **See Also:** "Managing Database Objects with Object Browser" on page 16-1

To view the database objects used for an application:

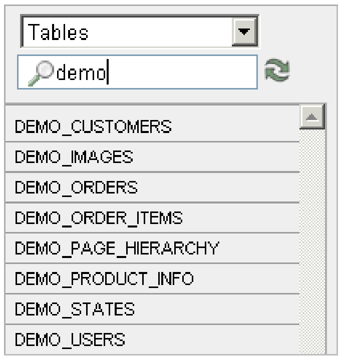
1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. Select an object type from the Object list in the upper left corner of the page. For example, to view tables, select **Tables**.

3. To search for an object name, enter keywords in the search field beneath the Object list.

   A list of matching objects appears.



DEMO_CUSTOMERS
DEMO_IMAGES
DEMO_ORDERS
DEMO_ORDER_ITEMS
DEMO_PAGE_HIERARCHY
DEMO_PRODUCT_INFO
DEMO_STATES
DEMO_USERS

4. To perform a specific task related to the selected object, select the object and the appropriate task button.

   For example, to modify a column in the `DEMO_CUSTOMERS` table:

   a. From the Objects list, select **Tables**.

   b. From the Tables list, select `DEMO_CUSTOMERS`.

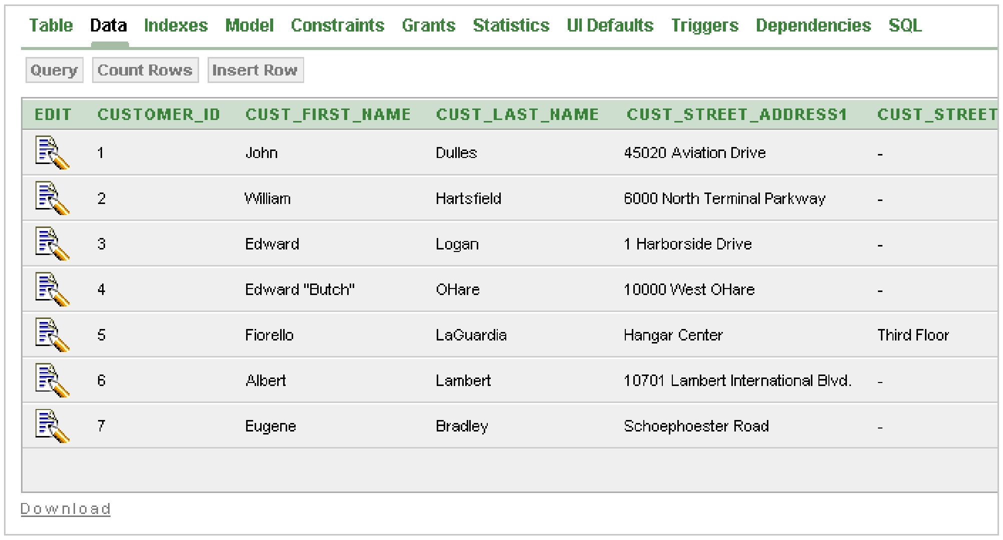
   c. Click **Modify Column**.

5. To view additional object details, select a tab beneath the object name. For example, to view the data in the `DEMO_CUSTOMERS` table:

   a. From the Tables list, select `DEMO_CUSTOMERS`.

   b. Select the **Data** tab.

      A report appears that displays the data in the `DEMO_CUSTOMERS` table appears.



EDIT	CUSTOMER_ID	CUST_FIRST_NAME	CUST_LAST_NAME	CUST_STREET_ADDRESS1	CUST_STREET
	1	John	Dulles	45020 Aviation Drive	-
	2	William	Hartsfield	6000 North Terminal Parkway	-
	3	Edward	Logan	1 Harborside Drive	-
	4	Edward "Butch"	OHare	10000 West OHare	-
	5	Fiorello	LaGuardia	Hangar Center	Third Floor
	6	Albert	Lambert	10701 Lambert International Blvd.	-
	7	Eugene	Bradley	Schoephoester Road	-

Part II

Application Development

Part II describes how to use Application Builder to develop database-driven applications.

Part II contains the following chapters:

3 Application Builder Concepts

This section provides basic conceptual information about Application Builder. Use Application Builder to assemble an HTML interface (or application) on top of database objects such as tables and procedures. Each application is a collection of pages linked together using tabs, buttons, or hypertext links.

This section contains the following topics:

See Also: "Using Application Builder" on page 4-1 and "Building an Application" on page 5-1

Note: The Application Builder icon does not appear when you log in to the Database Home Page as user `SYSTEM`. It is expected that this user performs administration tasks only. To develop Application Express applications, log in as a user other than `SYSTEM`.

What Is a Page?

A page is the basic building block of an application. When you build an application in Application Builder, you create pages that contain user interface elements, such as tabs, lists, buttons, items, and regions.

You add controls to a page on the Page Definition.

To view the Page Definition of an existing page:

1. Navigate to the Database Home Page.
2. Click the **Application Builder** icon.
3. Select an application.
4. Select a page.

   The Page Definition appears and is divided into three main sections:

- **Page Rendering** lists user interface controls and logic that is executed when the page is rendered. See "About Page Rendering Controls" on page 4-23.
- **Page Processing** lists logic controls (such as computations and processes) that are evaluated and executed when the page is processed. See "About Page Processing Controls" on page 4-25.
- **Shared Components** lists common components that can display or be applied on every page within an application. See "About Shared Components" on page 4-26.

See Also: "About the Page Definition" on page 4-16 and "Editing a Page Definition" on page 4-19

Understanding Page Processing and Page Rendering

When you create an application in Application Builder, you link pages together using tabs, buttons, or hypertext links. Each page can have buttons and items and can include application logic. You can branch from one page to the next using conditional navigation, perform calculations, validations, and display reports, calendars, and charts. You can generate reports, charts, and forms using built-in wizards, static HTML, or deliver more custom rendering with PL/SQL programming.

Topics in this section include:

- How the Application Express Engine Renders and Processes Pages
- Understanding Conditional Rendering and Processing
- Verifying User Identity
- Controlling Access to Controls and Components

How the Application Express Engine Renders and Processes Pages

The Application Express engine dynamically renders and processes pages based on data stored in Oracle database tables. To view a rendered version of your application, you request it from the Application Express engine. When you run an application, the Application Express engine relies on two processes:

- **Show Page** is the page rendering process. It assembles all the page attributes (including regions, items, and buttons) into a viewable HTML page.
- **Accept Page** performs page processing. It performs any computations, validations, processes, and branching.

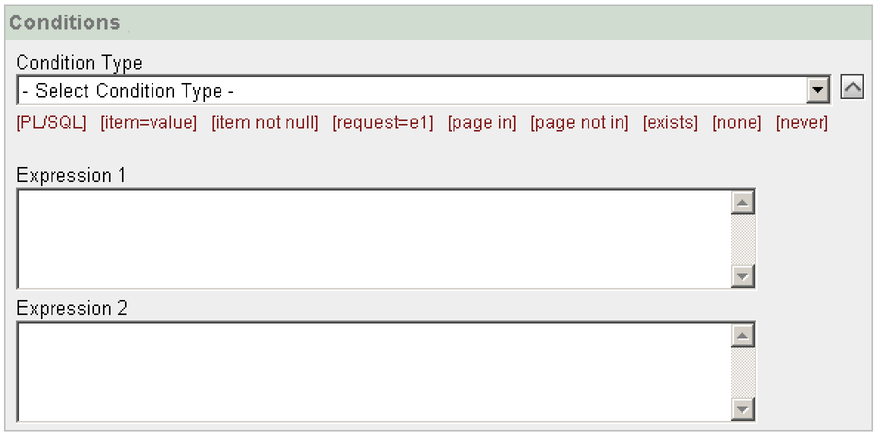
When you request a page using a URL, the engine is running Show Page. When you submit a page, the Application Express engine is running Accept Page or performing page processing during which it saves the submitted values in the session cache and then performs any computations, validations, or processes.

Understanding Conditional Rendering and Processing

A condition is a small unit of logic that helps you control the display of regions, items, buttons, and tabs as well as the execution of processes, computations, and validations. For example, when you apply a condition to a button, the rendering engine evaluates the condition during the rendering (or Show Page) process. Whether the condition passes or fails determines if the page control (such as a button) displays.

You specify a condition by selecting a condition type when you create the control or component (for example, the region, item, button, or tab) or by making a selection from the Condition Type attribute. Depending upon the Condition Type you select, enter the appropriate values in the Expressions fields. The condition evaluates to true or false based on the values you enter in the Expression fields.

Note: Whether you use the Expression fields depends upon the selected condition type. Some condition types do not require values in either field, others require a value only for Expression 1, and other condition types require values in both fields. Although these fields are labeled "Expression 1" and "Expression 2", the values for a given condition type do not necessarily conform to any formal definition of the term **expression**. They are simply text values appropriate for the selected condition type.


Conditions
Condition Type
- Select Condition Type -
[PL/SQL] [item=value] [item not null] [request=e1] [page in] [page not in] [exists] [none] [never]
Expression 1
Expression 2

To view a complete listing of all available conditions for a given component or control click the small icon that resembles an upward arrow to the right of the Condition Type list. Shortcuts to common selections appear directly beneath the list. If your condition requires an expression, enter it in the appropriate field.

The following sections offer examples of some commonly used condition types.

See Also: Appendix A, "Available Conditions" on page A-1 for a detailed listing of available condition types

Current Page in Expression 1

Current page in Expression 1 evaluates to true if the current page ID is contained within the comma-delimited list of pages in Expression 1. For example:

```
100
```

If the current page is 100, then this condition evaluates to true and the condition passes.

Exists

Exists (SQL query returns at least one row) is expressed as a SQL query. If the query returns at least one row, then the condition evaluates as true. For example:

```
SELECT 1 FROM emp WHERE deptno = :P101_DEPTNO
```

This example references item `P101_DEPTNO` as a bind variable. You can use bind variables within application processes and SQL query regions to reference item session state. If one or more employees are in the department identified by the value of P101_DEPTNO, then the condition evaluates as true.

See Also: "About Bind Variables" on page 3-9

PL/SQL Expression

Use **PL/SQL Expression** to specify an expression in valid PL/SQL syntax that evaluates to true or false. For example:

```
NVL(:MY_ITEM,'NO') = 'YES'
```

If the value of :MY_ITEM is Yes, then the condition evaluates as true. Otherwise, it evaluates as false.

Verifying User Identity

Authentication is the process of establishing users' identities before they can access an application. Authentication may require a user to enter a user name and password, or may involve the use of a digital certificate or a secure key.

Oracle Application Express supports modular authentication, making it easy to switch authentication methods when needed. You can establish a user's identity by selecting from a number of built-in authentication methods, or by using a wizard to create your own custom authentication approach.

See Also: "Establishing User Identity Through Authentication" on page 12-14 for more information

Controlling Access to Controls and Components

While conditions control the rendering and processing of specific controls or components on a page, authorization schemes control user access. Authorization is a broad term for controlling access to resources based on user privileges.

Authorization schemes extend the security of your application's authentication scheme. You can specify an authorization scheme for an entire application, a page, or specific page control such as a region, item, or button. For example, you could use an authorization scheme to selectively determine which tabs, regions, or navigation bar entries a user sees.

See Also: "Providing Security Through Authorization" on page 12-20

Understanding Session State Management

HTTP, the protocol over which HTML pages are most often delivered, is a stateless protocol. A Web browser is only connected to the server for as long as it takes to download a complete page. In addition, each page request is treated by the server as an independent event, unrelated to any page requests that happened previously or may occur in the future. This means that to access form values entered on one page on a subsequent page, some form of session state management needs to occur. Typically, when a user enters values into a form on one page, those values are not accessible on later pages. Oracle Application Express transparently maintains session state and provides developers with the ability to get and set session state values from any page in the application.

Topics in this section include:

- What Is a Session?
- Understanding Session IDs
- Referencing Session State

What Is a Session?

A **session** is a logical construct that establishes persistence (or stateful behavior) across page views. Each session is assigned a unique identifier. The Application Express engine uses this identifier (or session ID) to store and retrieve an application's working set of data (or session state) before and after each page view.

Because sessions are entirely independent of one another, any number of sessions can exist in the database at the same time. Because sessions persist in the database until purged by an administrator, a user can return to an old session and continue running an application long after first launching it. A user can also run multiple instances of an application simultaneously in different browser sessions.

Sessions are logically and physically distinct from Oracle database sessions used to service page requests. A user runs an application in a single Oracle Application Express session from log in to log out with a typical duration measured in minutes or hours. Each page requested during that session results in the Application Express engine creating or reusing an Oracle database session to access database resources. Often these database sessions last just a fraction of a second.

Understanding Session IDs

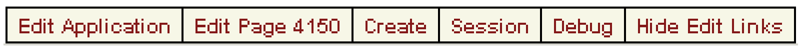
The Application Express engine establishes the identity (or anonymity) of the user for each page request and the session ID to fetch session state from the database. The most visible location of the session ID is in the URL for a page request. Another visible location is in the page's HTML POST data and indirectly in the contents of a session cookie. This cookie is sent by the Application Express engine during authentication and is maintained for the life of the application (or browser) session.

Oracle Application Express assigns new session IDs during authentication processing, records the authenticated user's identity with the session ID, and continually checks the session ID in each page request's URL or POST data with the session cookie and the session record in the database. These checks provide users with flexibility and security.

While the session ID is the key to session state, the session cookie (where applicable) and the session record safeguard the integrity of the session ID and the authentication status of the user.

Viewing Session State

The behavior of an Oracle Application Express application is usually driven by values in session state. For example, a button may display conditionally based on the value of an item session state. You can view the session state for a page by clicking **Session** on the Developer toolbar.



See Also: "Using the Developer Toolbar" on page 4-18 for more information about the Developer toolbar

About the Session State Page

The Session State page provides valuable information about the current page. To locate a specific page, enter the page ID in the page field and click **Go**. Table 3–1 describes the various types of information available on the Session State page.

Table 3–1 Information Available on the Session State Page

Heading	Description
Application	Identifies the application name, session ID, current user, workspace ID (or database user ID), and browser language.
Page Items	Identify attributes of the page item, including the application and page IDs, item name, how the item displays (hidden, popup, button, display only HTML), the item value in session state, and status. The Status column indicates the status of the session state. Available values include: ■ I - Inserted ■ U - Updated ■ R - Reset
Application Items	Application items are items that do not reside on a page. Application items are session state variables without the associated user interface properties. **See Also**: "Creating an Application-Level Item" on page 5-58 and "Using Substitution Strings" on page 3-13 for information about referencing item values
Session State	Summarizes session state for the current session. Lists applicable application IDs, page IDs, item names, display type, item values, and display labels.

See Also: "Managing Session State Values" on page 3-6

Managing Session State Values

When building interactive, data-driven Web applications, the ability to access and manage session state values is critical. In Oracle Application Express, session state is automatically managed for every page and easily referenced in static HTML or logic controls such as processes or validations.

Topics in this section include:

- Referencing Session State
- Setting Session State
- Clearing Session State
- About Bind Variables

See Also: "Items" on page 4-25 and "Referencing Item Values" on page 5-55

Referencing Session State

Referencing the value of an item is one of the most common examples of referencing session state. An item can be a field, a text area, a password, a select list, or check box. Table 3–2 describes the supported syntax for referencing item values.

Table 3–2 Syntax for Referencing Item Values

Type	Syntax	Description
SQL	`:MY_ITEM`	Standard bind variable syntax for items whose names are no longer than 30 characters. Use this syntax for references within a SQL query and within PL/SQL.
PL/SQL	`V('MY_ITEM')`	PL/SQL syntax referencing the item value using the `V` function. **See Also:** "Oracle Application Express APIs" on page 15-1
PL/SQL	`NV('MY_NUMERIC_ITEM')`	Standard PL/SQL syntax referencing the numeric item value using the `NV` function. **See Also:** "Oracle Application Express APIs" on page 15-1
Static text (exact)	`&MY_ITEM.`	Static text. Exact substitution.

Setting Session State

When a user submits a page, the Application Express engine automatically stores values typed into fields (items) in session state. For example, suppose you have an application containing two pages. The first page of the application contains a form in which a user can enter a phone number. You defined this form by creating an item named *`P2_PhoneNo`*. On the second page, you want to display the information the user enters in the form.

When the page is submitted, Oracle Application Express captures the value entered in the phone number field and stores the value for future use. The phone number entered by the user can then be retrieved from session state by referencing the item associated with the field on the page.

Clearing Session State

As you develop your applications, you may find it useful to clear the cached value for specific items, all items on a page, all pages in an application, or the current user session. Clearing a cached value resets the value to null. The topics that follow offer specific examples of clearing session state.

Topics in this section include:

- Clearing Cache by Item
- Clearing Cache by Page
- Clearing Cache for an Entire Application
- Clearing Cache for the Current User Session

Clearing Cache by Item

Clearing cache for a single item resets the value of the item to null. For example, you might use this approach to make sure a specific item's value is null when a page is prepared for rendering.

The following example uses standard `f?p` syntax to clear the cache for an item. This example calls page 5 of application 100. Placing `MY_ITEM` in the `ClearCache` position of the `f?p` syntax resets the value of `MY_ITEM` to `NULL`.

```
f?p=100:5:&SESSION.::NO:MY_ITEM
```

The following example resets the value of the items *`THE_EMPNO`* and *`THE_DEPTNO`*.

```
f?p=100:5:&SESSION.::NO:THE_EMPNO,THE_DEPTNO
```

Clearing Cache by Page

Caching application items provides an effective way to maintain session state. However, there are occasions when you may want to clear the cache for all items on a page. For example, suppose you needed to clear all fields on a page when a user clicks a link that creates a new order. By clearing the cache for an entire page, you set the value of all items on the page to null.

Clearing Session Cache for Two Pages While Resetting Pagination This example clears the session cache for two pages and resets pagination.

```
f?p=6000:6003:&SESSION.::NO:RP,6004,6014
```

This example:

- Runs page 6003 of application 6000 and uses the current session ID
- Indicates to not show debug information (`NO`)
- Clears all values maintained by the current session's cache for items of pages 6004 and 6014
- Resets region pagination (`RP`) on page 6003 (the requested page)

> **See Also:** "Controlling Report Pagination" on page 5-19

Clearing Session Cache on a Page and Passing an Item Value This example shows how to implement an update form. It clears existing information and sets the item's value (typically a primary key).

```
f?p=6000:6003:&SESSION.::NO:6003:MY_ITEM:1234
```

This example:

- Runs page 6003 of application 6000 and use the current session ID
- Indicates to not show debug information (`NO`)
- Clears all values maintained by the current session's cache for items on page 6003
- Sets the session state of an item called `MY_ITEM` to the value `1234`

Clearing Session Cache on a Page and Passing Values to Multiple Items This example is similar to the previous example, except it passes values to multiple items.

```
f?p=6000:6004:&SESSION.::NO:6003:MY_ITEM1,MY_ITEM2,MY_ITEM3:1234,,5678
```

This example:

- Runs page 6004 of application 6000 and use the current session ID
- Clears the current session's cache for items on page 6003
- Indicates debug information should be hidden (`NO`)
- Sets the value of `MY_ITEM1` to 1234, sets the value of `MY_ITEM2` to null (indicated by the comma used as placeholder), and sets the value of `MY_ITEM3` to 5678

Clearing Cache for an Entire Application

You can clear an application's cache by using `f?p` syntax and creating a `Clear Cache` argument using the keyword *APP*. For example:

```
f?p=App:Page:Session::NO:APP
```

Note: Resetting the cache for an entire application does not restore the application to a completely reset state. For example, if an application includes on-new instance computations or on-new instance processes, the Application Express engine runs these computations and processes when the application session is created. Then, it processes the clear cache request and displays the requested page.

The only way to reset the application completely without a session ID (if no cookie is used to track the session ID) is to request it using a URL without a session ID, or by calling `HTMLDB_APPLICATION.CLEAR_APP_CACHE` from another application. If the session ID is tracked using a cookie, you will need to logout to reset the state.

Clearing Cache for the Current User Session

Another approach to clearing an application's cache is to create a `Clear Cache` argument using the keyword `SESSION`. For example:

```
f?p=6000:6004:12507785108488427528::NO:SESSION
```

About Bind Variables

You can use bind variables within an application process or SQL query to reference session state of a specified item. For example:

```
SELECT * FROM emp WHERE name like '%' || :SEARCH_STRING || '%'
```

In this example, the search string is a page item. If the region type is defined as SQL Query, you can reference the value using standard SQL bind variable syntax. Using bind variables ensures that parsed representations of SQL queries are reused by the database, optimizing memory usage by the server.

When using bind variable syntax, remember the following rules:

- Bind variable names must correspond to an item name.
- Bind variable names are not case-sensitive.
- Bind variable names cannot be longer than 30 characters (that is, they must be a valid Oracle identifier).

  Although page item and application item names can be up to 255 characters, if you intend to use an application item within SQL using bind variable syntax, the item name must be 30 characters or less.

Using Bind Variables in Regions Based on a SQL Query or LOV

If your region type is defined as a SQL Query, SQL Query (plsql function body returning SQL query), or list of values (LOV), you can reference session state using the following syntax:

```
:MY_ITEM
```

One common way to do this is to incorporate a session state variable in a `WHERE` clause. The following example shows how to bind the value of the item `THE_DEPTNO` into a region defined from a SQL Query.

```
SELECT ename, job, sal
FROM emp
WHERE deptno = :THE_DEPTNO
```

> **See Also:** "Customizing Regions" on page 7-2 for information about creating regions

Using Bind Variables in PL/SQL Procedures

For region types defined as a PL/SQL Procedure, regions are constructed using PL/SQL anonymous block syntax. In other words, the beginning and ending are added automatically around the PL/SQL. For example:

```
INSERT INTO emp (empno, ename, job)
VALUES (:P1_empno, :P1_name, :P1_job);
```

In this example, the values of the `empno`, `ename`, and `job` are populated by the values of `P1_empno`, `P1_name`, and `P1_job`.

Understanding URL Syntax

Each application has a number (called an application ID) that uniquely identifies it. Similarly, each page also has a unique number (called a page ID). Applications and pages may also have alphanumeric aliases. Application aliases are unique within a a database user account and page aliases are unique within each application. When you run an application, the Application Express engine generates a session number that serves as a key to the user's session state.

Topics in this section include:

- Understanding the URL that Displays for a Page
- Using f?p Syntax to Link Pages
- Calling a Page Using an Application and Page Alias
- Calling a Page from a Button URL

Understanding the URL that Displays for a Page

The URL that displays for each page indicates the location of Oracle Application Express and identifies the address of Oracle Application Express, the application ID, page ID, and session ID. For example:

```
http://apex.oracle.com/pls/apex/f?p=4350:1:220883407765693447
```

This example indicates:

- `apex.oracle.com` is the URL of the server
- `pls` is the indicator to use the `mod_plsql` cartridge
- `apex` is the database access descriptor (DAD) name. The DAD describes how HTTP Server connects to the database server so that it can fulfill an HTTP request. The default value is `apex`.

- `f?p=` is a prefix used by Oracle Application Express
- `4350` is application being called
- `1` is the page within the application to be displayed
- `220883407765693447` is the session number

See Also: "About Publishing the Application URL" on page 11-14

Using f?p Syntax to Link Pages

You can create links between pages in your application using the following syntax:

```
f?p=App:Page:Session:Request:Debug:ClearCache:itemNames:itemValues:PrinterFriendly
```

Table 3–3 describes the arguments you can pass when using `f?p` syntax.

Table 3–3 f?p Syntax Arguments

Syntax	Description
`App`	Indicates an application ID or alphanumeric alias.
`Page`	Indicates a page ID or alphanumeric alias.
`Session`	Identifies a session ID. You can reference a session ID to create hypertext links to other pages that maintain the same session state by passing the session number. You can reference the session ID using the syntax: ■ Short substitution string: `&SESSION`. ■ PL/SQL: `V('SESSION')` ■ Bind variable: `:APP_SESSION`
`Request`	Sets the value of `REQUEST`. Each application button sets the value of `REQUEST` to the name of the button. This enables accept processing to reference the name of the button when a user clicks it. You can reference `REQUEST` using the syntax: ■ Substitution string: `&REQUEST`. ■ PL/SQL: `V('REQUEST')` ■ Bind variable: `:REQUEST`
`Debug`	Displays application processing details. Valid values for the DEBUG flag are `YES` or `NO`. Setting this flag to `YES` displays details about application processing. You can reference the Debug flag using the following syntax: ■ Short substitution string: `&DEBUG`. ■ PL/SQL: `V('DEBUG')` ■ Bind variable: `:DEBUG`
`ClearCache`	Clears the cache. This sets the value of items to null. To clear cached items on a single page, specify the numeric page ID. To clear cached items on multiple pages, use a comma-separated list of page IDs. Clearing a page's cache also resets any stateful processes on the page. Individual or comma-separated values can also include collection names to be reset or the keyword `RP`, which resets region pagination on the requested page. The keyword `APP` clears cache for all pages and all application-level items in the current application and removes sort preferences for the current user. The keyword `SESSION` achieves the same result as the `APP` keyword, but clears items associated with all applications that have been used in the current session.

Table 3–3 (Cont.) f?p Syntax Arguments

Syntax	Description
`itemNames`	Comma-delimited list of item names used to set session state with a URL.
`itemValues`	List of item values used to set session state within a URL. Item values cannot include colons, but can contain commas if enclosed with backslashes. To pass a comma in an item value, enclose the characters with backslashes. For example: `\123,45\`
`PrinterFriendly`	Determines if the page is being rendered in printer friendly mode. If PrinterFriendly is set to Yes, then the page is rendered in printer friendly mode. The value of PrinterFriendly can be used in rendering conditions to remove elements such as regions from the page to optimize printed output.You can reference the printer friendly preference by using the following syntax: `V('PRINTER_FRIENDLY')` When referenced, the Application Express engine will not display tabs or navigation bars, and all items will be displayed as text and not as form elements.

Although it is important to understand how `f?p` syntax works, you rarely have to construct this syntax yourself. Application Builder includes many wizards that automatically create these references for you. The following sections describe specific instances that utilize `f?p` syntax to link pages.

Calling a Page Using an Application and Page Alias

Application and page aliases must consist of valid Oracle identifiers, cannot contain any whitespace, and are not case-sensitive. The following example calls a page using an application and a page alias from within an application. It runs the page *`home`* of the application *`myapp`* and uses the current session ID.

```
f?p=myapp:home:&SESSION.
```

Application aliases must be unique within a database user account (or workspace). If an application in a different database user account has the same application alias, use the `&c` argument to specify the database user account (or workspace) name. For example:

```
f?p=common_alias:home:&session.&c=WORKSPACE_A
```

Calling a Page from a Button URL

When you create a button, you can specify a URL to redirect to when the user clicks the button. This example runs page 6001 of application 6000 and uses the current session ID.

```
f?p=6000:6001:&SESSION.
```

Note that this is only one approach to using a button. This method bypasses page submission and acts as a hyperlink on the page. Another method is to submit the page first. In that approach, clicking the button submits the page for processing, allowing forms to be submitted and session state to be saved.

See Also: "Creating Buttons" on page 5-47

Using Substitution Strings

You can use substitution strings within a page template or region source to replace a character string with another value. As you design your application and enable users to edit items, you will need to use substitution strings in order to pass information.

You can use substitution strings in Application Builder in the following ways.

- Include a substitution string within a template
- Reference page or application items using `&ITEM.` syntax
- Use built-in substitution strings to achieve a specific type of functionality

Substitution strings used within a template are delimited by the number symbol (#). For example:

```
#ABC#
```

To reference page or application items using substitution variables:

1. Precede the item name with an ampersand (&).
2. Append a period (.) to the item name.

For example, you would refer to an application item named `F101_X` in an HTML region, a region title, an item label, or in any of numerous other contexts as:

```
&F101_X.
```

Notice the required trailing period. When the page is rendered, Application Express engine replaces value the substitution string with the value of item `F101_X`.

Built-in Substitution Strings

Application Builder supports a number of built-in substitution strings. You may need to reference these values to achieve specific types of functionality.

The following sections describe these substitution strings, when to use them, and what supported syntax is currently available. Note that bind variable `:USER` has special meaning within the database.

Topics in this section include:

- HOME_LINK
- LOGIN_URL
- IMAGE_PREFIX
- Application Express SCHEMA OWNER
- PRINTER_FRIENDLY
- LOGOUT_URL
- PROXY_SERVER
- PUBLIC_URL_PREFIX
- REQUEST
- SQLERRM
- SYSDATE_YYYYMMDD
- WORKSPACE_IMAGES

See Also:

- "Substitutions" on page 4-10 for information about defining static substitution strings as an application attribute
- "Establishing User Identity Through Authentication" on page 12-14 for information about authentication

APP_ALIAS

APP_ALIAS is an alphanumeric name for the current application. APP_ALIAS is different from the APP_ID in that the APP_ID must be unique over all workspaces (or database user accounts) and all applications hosted in one database. In contrast, APP_ALIAS must be unique within a workspace. For example, by using the same APP_ALIAS you can create the application, ABC, in two different database user accounts. You can use APP_ALIAS almost anywhere APP_ID can be used. For example, f?p syntax can use an APP_ALIAS or an application ID as demonstrated in this example:

```
f?p=ABC:1:&SESSION.
```

This example runs application ABC, page 1 using the current session.

Table 3–4 describes the supported syntax for referencing APP_ALIAS.

Table 3–4 APP_ALIAS Syntax

Reference Type	Syntax
Bind variable	`:APP_ALIAS`
PL/SQL	`V('APP_ALIAS')`
Substitution string	`&APP_ALIAS.`

The following is an HTML example:

```
Click me to go to page 1 <a href="f?p=&APP_ALIAS.:1:&SESSION."> of the current
application</a>
```

APP_ID

`APP_ID` identifies the application ID of the currently executing application. Table 3–5 describes the supported syntax for referencing `APP_ID`.

Table 3–5 APP_ID Syntax

Reference Type	Syntax
Bind variable	`:APP_ID`
Direct PL/SQL	`HTMLDB_APPLICATION.G_FLOW_ID (A NUMBER)`
PL/SQL	`NV('APP_ID')`
Substitution string	`&APP_ID.`

The following is an example of a substitution string reference:

```
f?p=&APP_ID.:40:&SESSION.
```

APP_IMAGES

Use this substitution string to reference uploaded images, JavaScript, and cascading style sheets that are specific to a given application and are not shared over many applications. If you upload a file and make it specific to an application, then you must use this substitution string, or bind variable. Table 3–6 describes the supported syntax for referencing `APP_IMAGES`.

Table 3–6 APP_IMAGES Syntax

Reference Type	Syntax
Bind variable	`:APP_IMAGES`
Direct PL/SQL	Not available.
PL/SQL	`V('APP_IMAGES')`
Substitution string	`&APP_IMAGES.`
Template substitution	`#APP_IMAGES#`

See Also: "IMAGE_PREFIX" on page 3-19, "WORKSPACE_IMAGES" on page 3-23, and "Managing Images" on page 7-45

APP_PAGE_ID

`APP_PAGE_ID` is the current application ID. For example, if your application was on page 3, then the result would be 3. Using this syntax is useful when writing application components that need to work generically in multiple applications. Table 3–7 describes the supported syntax for referencing `APP_PAGE_ID`.

Table 3–7 APP_PAGE_ID Syntax

Reference Type	Syntax
Bind variable	`:APP_PAGE_ID`
Direct PL/SQL	`HTMLDB_APPLICATION.G_FLOW_STEP_ID (A NUMBER)`
Direct PL/SQL	`:APP_PAGE_ID`
PL/SQL	`NV('APP_PAGE_ID')`

Table 3–7 (Cont.) APP_PAGE_ID Syntax

Reference Type	Syntax
Substitution string	`&APP_PAGE_ID.`

The following is an example of a substitution string reference:

```
f?p=&APP_ID.:&APP_PAGE_ID.:&SESSION.
```

APP_SESSION

`APP_SESSION` is one of the most commonly used built-in substitution strings. You can use this substitution string to create hypertext links between application pages that maintain a session state by passing the session number. Table 3–8 describes the supported syntax for referencing `APP_SESSION`.

Table 3–8 APP_SESSION Syntax

Reference Type	Syntax
Bind variable	`:APP_SESSION`
PL/SQL	`V('APP_SESSION')`
Short PL/SQL	`V('SESSION')`
Short substitution string	`&SESSION.`
Substitution string	`&APP_SESSION.`

Consider the following examples:

- From within an HTML region:

  ```
  <a href="f?p=100:5:&SESSION.">click me</a>
  ```

- Using PL/SQL:

  ```
  htf.anchor('f?p=100:5:'||V('SESSION'),'click me');
  ```

- Using a SQL query:

  ```
  SELECT htf.anchor('f?p=100:5:'||:app_session,'clickme') FROM DUAL;
  ```

APP_UNIQUE_PAGE_ID

`APP_UNIQUE_PAGE_ID` is an integer generated from an Oracle sequence which is unique for each page view. This number is used by applications to prevent duplicate page submissions and can be used for other purposes. For example, if you want to make a unique URL to avoid browser caching issues, you can embed this number in the request or debug column in calls to the `f` procedure. Table 3–9 describes the supported syntax for referencing `APP_UNIQUE_PAGE_ID`.

Table 3–9 APP_UNIQUE_PAGE_ID Syntax

Reference Type	Syntax
Bind variable	`:APP_UNIQUE_PAGE_ID`
PL/SQL	`V('APP_UNIQUE_PAGE_ID')`
Substitution string	`&APP_UNIQUE_PAGE_ID.`

The following is an HTML example:

```
SELECT 'f?p=100:1:'||:APP_SESSION||':'||:APP_UNIQUE_PAGE_ID||
    ':::P1_EMPNO:'||empno,
   ename,
    job
FROM emp
```

Note the use of the `APP_UNIQUE_PAGE_ID` in the request column. This makes this URL unique and may avoid excessive browser caching problems.

APP_USER

`APP_USER` is the current user running the application. Depending upon your authentication model, the value of the user is set differently. If the application is running using database authentication, then the value of the user is the same as the database pseudo column USER. If the application uses an authentication scheme that requires the user to authenticate, the value of `APP_USER` is set by the authentication scheme, usually to the user name used during authentication. Table 3–10 describes the supported syntax for referencing `APP_USER`.

Table 3–10 APP_USER Syntax

Reference Type	Syntax
Bind variable	`:APP_USER`
PL/SQL	`V('APP_USER')`
Short PL/SQL	`V('USER')`
Substitution string	`&USER.`

Consider the following examples:

- From within an HTML region:

  ```
  Hello you are logged in as &USER.
  ```

- Using PL/SQL:

  ```
  htp.p('Hello you are logged in as'||V('USER'));
  ```

- As a bind variable:

  ```
  SELECT * FROM some_table WHERE user_id = :app_user
  ```

> **See Also:** "Authentication" on page 4-12 for information about the Public User attribute

AUTHENTICATED_URL_PREFIX

This application-level attribute identifies a valid authenticated prefix (that is, a logged in URL prefix). You can use a relative path or a full path beginning with `http`. This item is useful if your application can be run in both authenticated (logged in) and public (not logged in) modes. You can use `AUTHENTICATED_URL_PREFIX` to construct a link to an authenticated page. This item is most useful when using basic database authentication because changes to the URL can require authentication. Table 3–11 describes the supported syntax for referencing `AUTHENTICATED_URL_PREFIX`.

Table 3–11 AUTHENTICATED_URL_PREFIX Syntax

Reference Type	Syntax
Bind variable	`:AUTHENTICATED_URL_PREFIX`
PL/SQL	`V('AUTHENTICATED_URL_PREFIX')`
Substitution string	`&AUTHENTICATED_URL_PREFIX.`

BROWSER_LANGUAGE

`BROWSER_LANGUAGE` refers to the Web browser's current language preference. Table 3–12 describes the supported syntax for referencing `BROWSER_LANGUAGE`.

Table 3–12 BROWSER_LANGUAGE Syntax

Reference Type	Syntax
Bind variable	`:BROWSER_LANGUAGE`
Direct PL/SQL	`HTMLDB_APPLICATION.G_BROWSER_LANGUAGE`
PL/SQL	`V('BROWSER_LANGUAGE')`
Substitution string	`:BROWSER_LANGUAGE.`
Substitution string	`&BROWSER_LANGUAGE.`

CURRENT_PARENT_TAB_TEXT

`CURRENT_PARENT_TAB_TEXT` is most useful in page templates, but is only relevant for applications that use two-level tabs (that is, parent and standard tabs). Use this string to reference the parent tab label. This substitution string enables you to repeat the currently selected parent tab within the page template. Table 3–13 describes the supported syntax for referencing `CURRENT_PARENT_TAB_TEXT`.

Table 3–13 CURRENT_PARENT_TAB_TEXT Syntax

Reference Type	Syntax
Bind variable	Not Available.
Substitution string	`&CURRENT_PARENT_TAB_TEXT.`

DEBUG

Valid values for the `DEBUG` flag are Yes or No. Turning debug on shows details about application processing. If you write your own custom code, you may want to generate debug information only if the debug mode is set to Yes. Table 3–14 describes the supported syntax for referencing `DEBUG`.

Table 3–14 DEBUG Syntax

Reference Type	Syntax
Bind variable	`:DEBUG`
Direct PL/SQL	`HTMLDB_APPLICATION.G_DEBUG`
PL/SQL	`V('DEBUG')`
Substitution string	`&DEBUG.`

The following is an example of a substitution string reference that preserves the current value of `DEBUG`:

```
f?p=100:1:&SESSION.::&DEBUG
```

HOME_LINK

HOME_LINK is the home page of an application. The Application Express engine will redirect to this location if no page is given and if no alternative page is dictated by the authentication scheme's logic. You define the Home Link on the Application Attributes page.

Table 3–15 describes the supported syntax for referencing HOME_LINK.

Table 3–15 HOME_LINK Syntax

Reference Type	Syntax
Direct PL/SQL	`HTMLDB_APPLICATION.G_HOME_LINK`
PL/SQL	`V('HOME_LINK')`
Template Reference	`#HOME_LINK#`
Substitution String	`&HOME_LINK.`

> **See Also:** "Authentication" on page 4-12 for information about the Home Link attribute

LOGIN_URL

Use LOGIN_URL to display a link to a login page for users that are not currently logged in. Table 3–16 describes the supported syntax for LOGIN_URL.

> **See Also:** "Authentication" on page 4-12 and "Editing Security Attributes" on page 4-12

Table 3–16 LOGIN_URL Syntax

Reference Type	Syntax
Bind variable	`:LOGIN_URL`
Direct PL/SQL	`HTMLDB_APPLICATION.G_LOGIN_URL`
PL/SQL	`V('LOGIN_URL')`
Substitution string	`&LOGIN_URL.`
Template Substitution	`#LOGIN_URL#`

IMAGE_PREFIX

The value of IMAGE_PREFIX determines the virtual path the Web server uses to point to the images directory distributed with Oracle Application Express. If you want to reference uploaded images, use WORKSPACE_IMAGES and APP_IMAGES. Table 3–17 describes the supported syntax for referencing IMAGE_PREFIX.

> **See Also:** "APP_IMAGES" on page 3-15,"WORKSPACE_IMAGES" on page 3-23, and "Configuring Standard Application Attributes" on page 4-6

Table 3–17 IMAGE_PREFIX Syntax

Reference Type	Syntax
Bind variable	`:IMAGE_PREFIX`
Direct PL/SQL	`HTMLDB_APPLICATION.G_IMAGE_PREFIX`
PL/SQL	`V('IMAGE_PREFIX')`
Substitution string	`&IMAGE_PREFIX.`
Template Substitution	`#IMAGE_PREFIX#`

Application Express SCHEMA OWNER

If you are generating calls to applications from within your PL/SQL code, you may need to reference the owner of the Oracle Application Express schema. The following describes the correct syntax for a direct PL/SQL reference:

```
HTMLDB_APPLICATION.G_FLOW_SCHEMA_OWNER
```

You may also use `#FLOW_OWNER#` to reference this value in SQL queries and PL/SQL (for example, in a region or a process).

PRINTER_FRIENDLY

The value of `PRINTER_FRIENDLY` determines if the Application Express engine is running in print view mode. This setting can be referenced in conditions to eliminate elements not desired in a printed document from a page. Table 3–18 describes the supported syntax for referencing `PRINTER_FRIENDLY`.

Table 3–18 PRINTER_FRIENDLY Syntax

Reference Type	Syntax
Direct PL/SQL	`HTMLDB_APPLICATION.G_PRINTER_FRIENDLY (VARCHAR2 DATATYPE)`
PL/SQL	`V('PRINTER_FRIENDLY')`
Substitution string	`&PRINTER_FRIENDLY.`

LOGOUT_URL

`LOGOUT_URL` is an application-level attribute used to identify the logout URL. This is a URL that navigates the user to a logout page or optionally directly logs out a user. To create a logout navigation bar entry, add a trailing period to `&LOGOUT_URL` (`&LOGOUT_URL.`). If you are coding a page template, use `#LOGOUT_URL#`. Table 3–19 describes the supported syntax for referencing `LOGOUT_URL`.

Table 3–19 LOGOUT_URL Syntax

Reference Type	Syntax
Bind variable	`:LOGOUT_URL`
PL/SQL	`V('LOGOUT_URL')`
Substitution string	`&LOGOUT_URL.`
Template substitution	`#LOGOUT_URL#`

PROXY_SERVER

PROXY_SERVER is an application attribute. The attribute may be used by regions whose source comes from a URL. The following is the correct syntax for a direct PL/SQL reference used when you are writing PL/SQL to access remote Web servers from within the database (for example, when using the utl_http package shipped with the database).

```
HTMLDB_APPLICATION.G_PROXY_SERVER
```

PUBLIC_URL_PREFIX

PUBLIC_URL_PREFIX is an application-level attribute that identifies a URL to toggle out of a logged in mode to a public view. Table 3–20 describes the supported syntax for referencing PUBLIC_URL_PREFIX.

Table 3–20 PUBLIC_URL_PREFIX Syntax

Reference Type	Syntax
Bind variable	:PUBLIC_URL_PREFIX
PL/SQL	V('PUBLIC_URL_PREFIX')
Substitution string	&PUBLIC_URL_PREFIX.
Template substitution	#PUBLIC_URL_PREFIX#

REQUEST

Each application button sets the value of REQUEST to the name of the button or to the request value attribute associated with the button. This enables accept processing to reference the name of the button when a user clicks it. In the f?p syntax, REQUEST may be set using the fourth argument.

Referencing the Value of REQUEST REQUEST is typically referenced during Accept processing (that is, the processing that occurs when you post a page). Table 3–21 describes the supported syntax for referencing REQUEST.

Table 3–21 REQUEST Syntax

Reference Type	Syntax
Bind variable	:REQUEST
Direct PL/SQL	HTMLDB_APPLICATION.G_REQUEST
PL/SQL	V('REQUEST')
Substitution string	&REQUEST &REQUEST. (exact syntax match)

Scope and Value of REQUEST for Posted Pages When you post a page, you initiate Accept processing. Accept processing consists of computations, validations, processes, and branches. The value of REQUEST is available during each phase of the Accept processing. Once an application branches to a different page then REQUEST is set to NULL.

The value of REQUEST is the name of the button the user clicks, or the name of the tab the user selects. For example, suppose you have a button with a name of CHANGE, and a label Apply Change. When a user clicks the button, the value of REQUEST will be CHANGE.

Referencing REQUEST Using Declarative Conditions It is common to reference REQUEST using conditions. For example, you may want to reset pagination when a user clicks **Go** on a report page. You can reset pagination by creating an on-submit page process. The page process can be made conditional using the condition `Request = Expression 1`.

To create an on-submit page process:

1. Under Condition, select the condition type **Request = Expression 1**.
2. In Expression 1, enter **GO**.

Using REQUEST for Show Processing You can also use REQUEST for Show processing when navigating to a page using f?p syntax. For example:

```
f?p=100:1:&SESSION.:GO
```

Remember that the fourth argument in the f?p syntax is REQUEST. This example goes to application 100, page 1 for the current session, and sets the value of REQUEST to GO. Any process or region can reference the value of REQUEST using Show processing.

The following is a similar example using PL/SQL:

```
IF V ('REQUEST') = 'GO' THEN
   htp.p('hello');
END IF;
```

Note that `htp.p('hello')` is a call to a PL/SQL Web Toolkit package to print out the specified text string.

See Also:

- *Oracle Database Application Developer's Guide - Fundamentals* for information about developing Web applications with PL/SQL
- *Oracle Database PL/SQL Packages and Types Reference* for information about htp packages

SQLERRM

SQLERRM is a template substitution only available in the Applications Region Error Message. The following describes the correct syntax for a region template substitution reference:

```
#SQLERRM#
```

SYSDATE_YYYYMMDD

SYSDATE_YYYYMMDD represents the current date on the database server, with the *YYYYMMDD* format mask applied. You may use this value instead of repeated calls to the SYSDATE() function. The following list describes the supported syntax for referencing SYSDATE_YYYYMMDD.

- Bind variable

  ```
  :SYSDATE_YYYYMMDD
  ```

- PL/SQL

  ```
  V('SYSDATE_YYYYMMDD')
  ```

- Direct PL/SQL

  ```
  HTMLDB_APPLICATION.G_SYSDATE (DATE DATATYPE)
  ```

Table 3–22 SYSDATE_YYYYMMDD Syntax

Reference Type	Syntax
Bind variable	`:SYSDATE_YYYYMMDD`
Direct PL/SQL	`HTMLDB_APPLICATION.G_SYSDATE (DATE DATATYPE)`
PL/SQL	`V('SYSDATE_YYYYMMDD')`

WORKSPACE_IMAGES

Use this substitution string to reference uploaded images, JavaScript, and cascading style sheets that are shared over many applications within a workspace (or database user account). Table 3–23 describes the supported syntax for referencing `WORKSPACE_IMAGES`.

Table 3–23 WORKSPACE_IMAGES Syntax

Reference Type	Syntax
Bind variable	`:WORKSPACE_IMAGES`
Direct PL/SQL	Not available
PL/SQL	`V('WORKSPACE_IMAGES')`
Substitution string	`&WORKSPACE_IMAGES.`
Template substitution	`#WORKSPACE_IMAGES#`

See Also: "APP_IMAGES" on page 3-15 and "IMAGE_PREFIX" on page 3-19

4 Using Application Builder

This section provides important background information about using Application Builder to build dynamically rendered applications.

This section contains the following topics:

- Accessing Application Builder
- About the Application Builder Home Page
- About the Application Home Page
- About Application Attributes
- About the Page Definition
- Using the Developer Toolbar
- Editing a Page Definition
- Linking to Other Pages from the Page Definition
- About the Shared Components Page
- Understanding Application Processes
- Understanding Application Computations
- Viewing Application Reports

See Also:

- "Quick Start" on page 1-1
- "Application Builder Concepts" on page 3-1
- "Building an Application" on page 5-1
- "Controlling Page Layout and User Interface" on page 7-1
- "Adding Navigation" on page 6-1

Accessing Application Builder

An application is a collection of database-driven Web pages linked together using tabs, buttons, or hypertext links. The pages within an application share a common session state definition and authentication method. Application Builder is the tool you use to build the pages that comprise an application.

Note: The Application Builder icon does not appear when you log in to the Database Home Page as user `SYSTEM`. It is expected that this user performs administration tasks only. To develop Application Express applications, log in as a user other than `SYSTEM`.

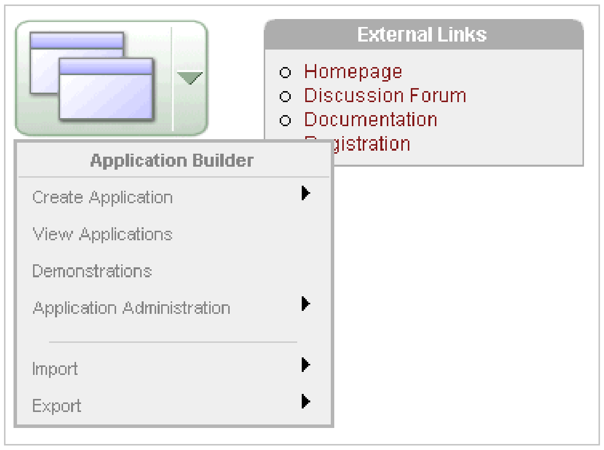
To access Application Builder:

1. Log in to the Database Home Page user interface.

   The Database Home Page appears.

2. To view the **Application Builder** home page you can either:

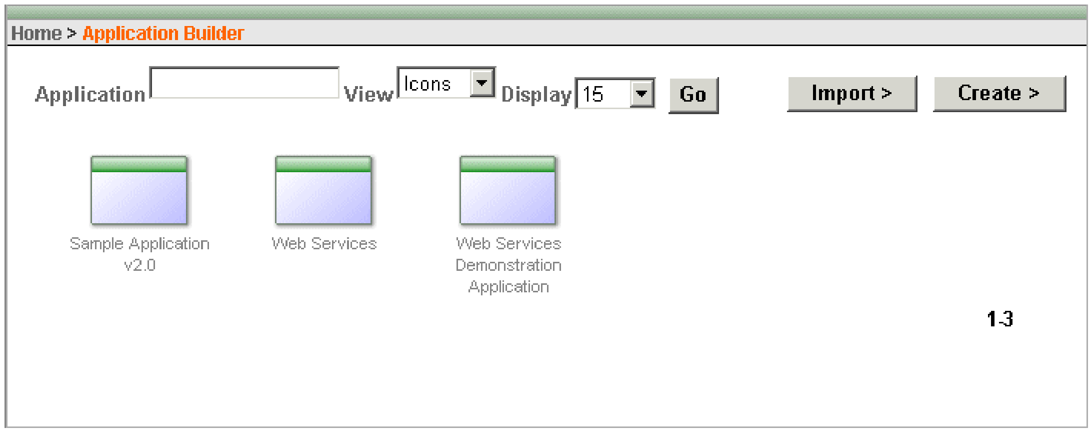
   - Click the Application Builder icon to link to the Application Builder home page.
   - Click the down arrow on the right side of the Application Builder icon to view a drop down menu. Then select the appropriate menu option.



Note: For the purposes of the consistency, this document uses the primary navigation path (or drill-down approach) when explaining navigation.

About the Application Builder Home Page

The Application Builder home page displays all installed or recently created applications.



You can use the controls at the top of the Application Builder home page to:

- **Search for an Application.** Enter keywords in the Application field and click **Go**. To view all applications, leave the Application field blank and click **Go**. You control how many rows display by making a selection from the Display list.
- **Change the Page View**. By default, each application displays as a large icon. You can change the appearance of the page by making a selection from the View list. Available View options include:
  - **Icons** (the default) displays each application as a large icon identified by the application name.
  - **Details** displays each application as a line in a report. Each line includes the application ID, the application name, when the application was last updated, the page count, and who last updated the application.

  You can control how many applications display by making a selection from the Display list.
- **Import Application Files.** Click **Import** to import an exported application file.
- **Create an Application.** Click **Create** to create a new application or install a demonstration application.

Additionally, click an application icon or application name at the bottom of the page to view a specific application.

> **See Also:** "Importing Export Files" on page 11-11, "Installing a Demonstration Application" on page 2-1, and "About Creating an Application Using a Wizard" on page 5-2

About the Administration Task List

The Administration task list appears on the right side of the Application Builder home page. Use the following links to administer your application development environment:

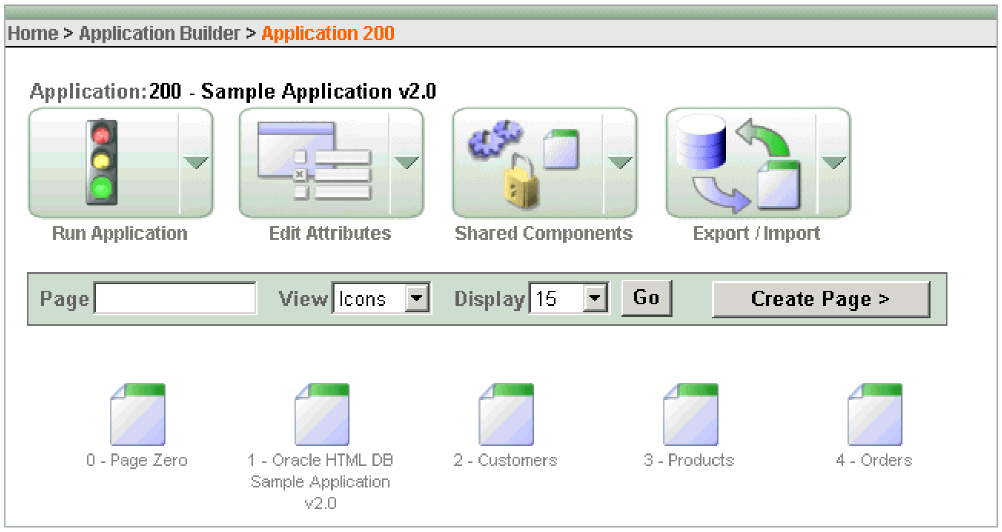
- **Manage Services** links to the Mange Services page. Use this page to manage session state, log files, preferences, and application models. See "About the Manage Services Page" on page 8-2.
- **Manage Application Express Users** links to the Manage Application Express Users page. Use this page to manage Application Express user accounts as well as user groups. See "Managing Application Express End Users" on page 8-11 and "Managing Application Express Users Using Groups" on page 8-13.

- **Monitor Activity** links to the Monitor Activity page. Use this page to monitor changes to page views and entire applications. See "Monitoring Activity" on page 8-15.
- **Email Configuration** links to the Email Configuration page. Use this page to learn how to send email from an Application Builder application. See "How to Send Email from an Application" on page 8-16.
- **About Application Express** links to the About Application Express page. See "About Application Express" on page 8-2.

See Also: "Understanding Application Administration" on page 8-1

About the Application Home Page

To view a specific application, select the application on the Application Builder home page. The Application home page appears. The application ID and the application name display at the top of the page.



The following four large icons appear next:

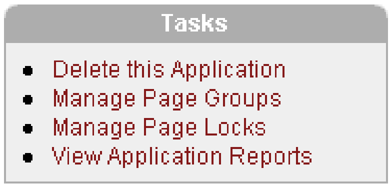
- **Run Application** submits the pages in the current application to the Application Express engine to render viewable HTML. See "How the Application Express Engine Renders and Processes Pages" on page 3-2.
- **Edit Attributes** links to the Application Attributes page where you can access Standard Attributes (attributes common to the entire application), Security Attributes, and Globalization Attributes. See "About Application Attributes" on page 4-6.
- **Shared Components** links to a list of shared components and user interface controls that can display or be applied on every page within an application. See "About the Shared Components Page" on page 4-40.
- **Export/Import** links you to the Export Import Wizard. Use this wizard to import and export an entire application as well as related files such as cascading style sheets, images, static files, script files, themes, user interface defaults, and workspaces (or database user accounts). "Exporting an Application and Related Files" on page 11-5.

A search bar appears in the center of the page and features the following controls:

- **Page.** Search for a page ID or name by entering a case insensitive keyword or phrase in the Page field and clicking **Go**. To view all pages in an application, leave the Page field blank and click **Go**. You control how many rows display by making a selection from the Rows list.
- **View.** By default, each page displays as a large icon. You can change the appearance of the page by making a selection from the View list. You can control how many pages display, by making a selection from the Display list. See "Page Display Alternatives" on page 4-5.
- **Create Page.** Click the **Create Page** button to launch a wizard to create a new page. See "Adding Pages to an Application" on page 5-8.

About the Tasks List

A Tasks list displays on the right side of the Application home page.



The Task list contains the following links:

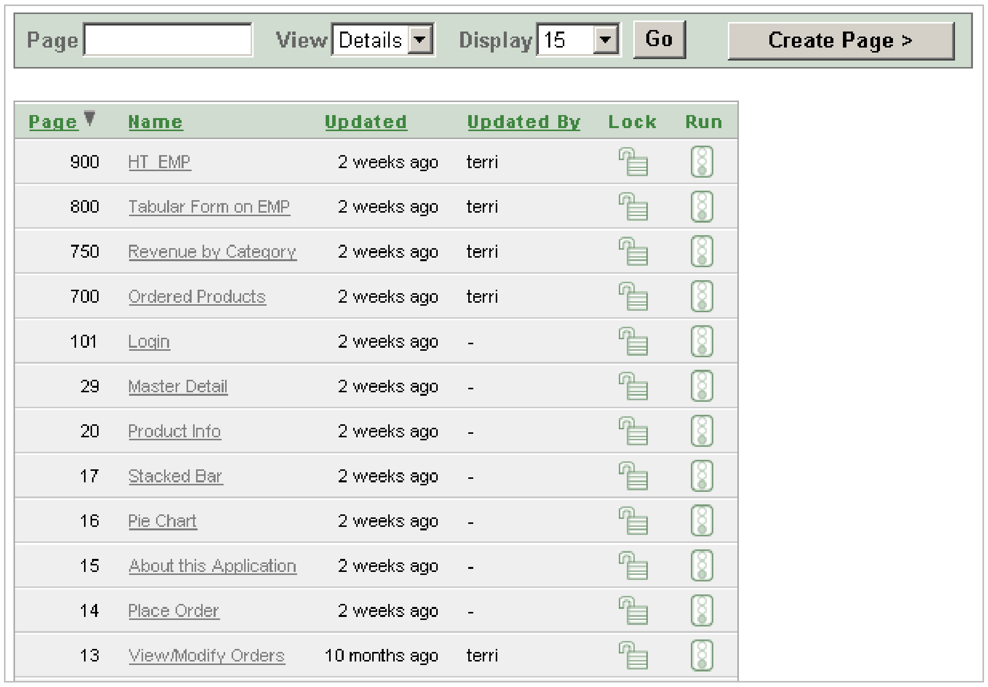
- **Delete this Application** deletes the current application. See "Deleting an Application" on page 5-7.
- **Manage Page Groups** links to the Page Groups page. Make the pages within your application easier to access by organizing them into page groups. See "Grouping Pages" on page 5-12.
- **Manage Page Locks** links to the Locked Pages page. Locking pages in an application prevents conflicts during application development. See "Locking and Unlocking a Page" on page 5-13.
- **View Application Reports** displays links to summary application reports. See "Viewing Application Reports" on page 4-49.

Page Display Alternatives

You can control how the pages display on the Application home page by making a selection from the View list. Available View options include:

- **Icons** (the default) displays each page as a large icon identified by the page name.
- **Details** displays each page as a line in a report. Each line includes the page ID, the page name, when the page was last updated, and who last updated the page.

  This view also includes a Lock and a Run icon. Use the Lock icon to prevent conflicts during application development. Click the **Run** icon to submit a page to the Application Express engine and render viewable HTML.



Page	Name	Updated	Updated By	Lock	Run
900	HT_EMP	2 weeks ago	terri		
800	Tabular Form on EMP	2 weeks ago	terri		
750	Revenue by Category	2 weeks ago	terri		
700	Ordered Products	2 weeks ago	terri		
101	Login	2 weeks ago	-		
29	Master Detail	2 weeks ago	-		
20	Product Info	2 weeks ago	-		
17	Stacked Bar	2 weeks ago	-		
16	Pie Chart	2 weeks ago	-		
15	About this Application	2 weeks ago	-		
14	Place Order	2 weeks ago	-		
13	View/Modify Orders	10 months ago	terri		

See Also: "Locking and Unlocking a Page" on page 5-13 and "Running a Page or Application" on page 5-11

About Application Attributes

Application attributes apply to an entire application. Once you create an application the next logical step is to review and possibly update application attributes.

Topics in this section include:

- Configuring Standard Application Attributes
- Configuring Security Attributes
- Configuring Globalization Attributes

Configuring Standard Application Attributes

Standard Application Attributes display on the Edit Application Attributes page. You use these attributes to control the application name and availability as well as defined substitution strings. Additionally, the Edit Application Attributes page displays defined build options, the associated theme, template defaults, and component defaults. Required values are marked with a red asterisk (*).

Topics in this section include:

- Accessing the Edit Application Attributes Page
- Editing Application Attributes

Accessing the Edit Application Attributes Page

To edit application attributes:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.

3. Click **Edit Attributes**.

   The Application Attributes page appears.

4. Click **Edit Standard Attributes**.

   The Edit Application Attributes page appears.

Editing Application Attributes

The following sections describe the attributes available on the Edit Application Attributes page.

Topics in this section include:

- Name
- Availability
- Global Notifications
- Substitutions
- Logo
- Build Options
- Theme
- Template Defaults
- Component Defaults

Name Use Name to define basic characteristics of your application, including the application name, an optional alphanumeric alias, a version number, and the application owner. Table 4–1 describes all Name attributes.

Table 4–1 Application Definition Attributes

Attribute	Description
Name	Provides a short descriptive name for the application to distinguish it from other applications in your development environment.
Application Alias	Assigns an alternate alphanumeric application identifier. You can use this identifier in place of the application ID. For example, suppose you create an alias of `myapp` for application 105. Using `f?p` syntax, you could call application 105 as either: ▪ `f?p=105:1` ▪ `f?p=myapp:1`
Version	Includes the application's version number on a page. You can also automatically tie the version to the date of last modification using the following format masks: ▪ `YYYY.MM.DD` ▪ `MM.DD.YYYY` ▪ `DD.MM.YYYY` If your application version uses YYYY.MM.DD then Application Builder replaces this format mask with the date of last modification of any application attribute.

Table 4–1 (Cont.) Application Definition Attributes

Attribute	Description
Image Prefix	Determines the virtual path the Web server uses to point to the images directory distributed with Application Builder. During installation, the virtual path is configured as `/i/`. When embedding an image in static text (for example, in page or region headers or footers) you can reference an image using the substitution string `#IMAGE_PREFIX#`. For example, to reference the image `go.gif` you would use the following syntax: `<img src="#IMAGE_PREFIX#go.gif">` **See Also:** "IMAGE_PREFIX" on page 3-19, "Managing Images" on page 7-45, and "Referencing Images" on page 7-46
Proxy Server	Use this field to specify a proxy server. For example, you may require a proxy server when using a region source type of URL. The URL region source embeds the results of the URL (that is, the page returned by navigating to the URL) as the region source. If you use a firewall and the target of a URL is outside the firewall relative to Application Builder, you may need to specify a proxy server. You can reference values entered into this field from PL/SQL using the PL/SQL package variable `HTMLDB_ APPLICATION.G_PROXY_SERVER`.
Logging	Determines whether or not user activity is recorded in the Oracle Application Express activity log. When set to **Yes**, every page view will be logged, allowing an administrator to monitor user activity for each application. Disabling logging may be advisable for high volume applications.
Parsing Schema	Specifies the schema that all SQL and PL/SQL in the application will be parsed as. You may use `#OWNER#` to reference this value in SQL queries and PL/SQL (for example, in a region or a process).

Exact Substitutions	Select whether or not only exact substitutions will be supported. For optimal run-time performance, it is recommended you use exact substitutions.
	
	Exact substitutions use the following sytnax:
	
	`&ITEM.`
	
	Non-exact substitutions use the following sytnax:
	
	`&ITEM`

See Also: "Using Substitution Strings" on page 3-13 and "Using f?p Syntax to Link Pages" on page 3-11

Availability Use Availability attributes to manage your application by defining an application status and build status. For example, if you select the status **Restricted Access**, you can specify which users have access and can run the application. Table 4–2 describes these attributes.

Table 4–2 Application Availability Attributes

Attribute	Description
Status	Specifies whether or not the application is available or unavailable for use. Options include: ▪ **Available** - Application is available with no restrictions. ▪ **Available with Edit Links** - Application is available for use. For developers, the Developer toolbar displays at the bottom of each page. Requires the developer to be logged in to the Application Builder in the same browser session. ▪ **Available to Developers Only** - Application is available to users having developer privileges. ▪ **Restricted Access** - Application is available to developers named in **Restrict to comma separated user list**. ▪ **Unavailable** - Application cannot be run or edited. The message in **Message for unavailable application** displays. when users attempt to access the application. ▪ **Unavailable (Status Shown with PL/SQL)** - Application cannot be run or edited. ▪ **Unavailable (Redirect to URL)** - Application cannot be run. The user is linked to the URL entered in **Message for unavailable application**.
Build Status	Identifies the build status of the current application: ▪ **Run and Build Application** - Developers can both run and develop the application. ▪ **Run Application Only** - Developers can only run the application.
Message for unavailable application	If you set Status to **Unavailable**, **Unavailable (Status Shown with PL/SQL)**, or **Unavailable (Redirect to URL)**, the text you enter in this attribute displays. If you set Status to **Available**, the text you enter in this attribute does not display.
Restrict to comma separated user list (Status must equal Restricted Access)	Use this attribute in conjunction with the Status **Restricted Access**. If you set Status to **Restricted Access**, only the users listed in this attribute can run the application. To use this attribute: 1. From the Status list, select **Restricted Access**. 2. Enter a comma-delimited list of users who can run the application in the field provided.

Global Notifications You can use the Global Notifications attribute to communicate system status to application users. For example, you can use this attribute to notify users of scheduled downtime, or communicate other messages regarding application availability. If the page templates used in your application contain the `#GLOBAL_NOTIFICATION#` substitution string, the text entered here will display in that string's place.

To create a global notification:

1. Include the `#GLOBAL_NOTIFICATION#` substitution string in your page template.
2. Navigate to the Edit Application Attributes page and enter a message in the Global Notifications attribute.
3. Click **Apply Changes**.

See Also: "Using Substitution Strings" on page 3-13

Substitutions Use these fields to define static substitution strings for your application. You can use static substitution string for phrases or labels that occur in many places within an application. Defining static substitution strings centrally enables you to change text strings in multiple places in your application by making a single change to the Substitution Value defined on this page.

See Also: "Using Substitution Strings" on page 3-13

Logo Use these attributes to identify an image to be used as the logo for this application.

In **Image**, identify the image preceded by the prefix identified in the Image Prefix attribute. The Image Prefix attribute determines the virtual path the Web server uses to point to the images directory distributed with Application Builder. The following example, includes the default prefix `/i/`:

```
/i/my_image.gif
```

If you specify an image in the Image attribute and include the `#LOGO#` substitution string in your page template, the Application Express engine generates an image tag. Use **Logo Image Attributes** to specify image attributes for the logo image. For example:

```
width="100" height="20" alt="Company Logo"
```

See Also: "Managing Images" on page 7-45, "Verifying the Prefix for the Virtual Image Directory" on page 7-46, "Customizing Templates" on page 7-17, and "Page Templates" on page 7-28

Build Options Displays existing build options. Most application attributes have a build option attribute. Do not specify a build option unless you plan to exclude that object from specific installations. Build Options have two possible values: `INCLUDE` and `EXCLUDE`. If you specify an attribute to be included, then the Application Express engine considers it at run time. However, if you specify an attribute to be excluded, then the Application Express engine treats it as if it does not exist.

See Also: "Using Build Options to Control Configuration" on page 11-15

Theme Displays the current theme applied to the application. Themes are collections of templates that can be used to define the layout and style of an entire application. Each theme provides a complete set of templates that accommodate every user interface pattern that may be needed in an application.

See Also: "Managing Themes" on page 7-8

Template Defaults Template Defaults list the default templates for this application. To specify a new template at the application level, you can either:

- Select a new theme
- Select a new default page template on the Define Theme page

You can also override this default by making a selection from the Page Template list on the Page Attributes page.

Table 4–3 describes template defaults for the current application.

Table 4–3 Application Template Defaults Attributes

Attribute	Description
Default Page Template	Indicates the default page template to display pages. You can override this selection by making a selection from the Page Template list on the Page Attributes page. **See Also:** "Editing Page Attributes" on page 4-20
Print Mode Page Template	Identifies the template to be used when the Application Express engine is in printer friendly mode. When calling the Application Express engine to render a page, you have the option to specify whether or not the page should be displayed using the Print Mode Page Template specified. If you specify Yes, then the page displays using a printer friendly template. The Application Express engine displays all text within HTML Form Fields as text. The printer friendly template does not need to have the `#FORM_OPEN#` or `#FORM_CLOSE#` substitution string. **See Also:** "Optimizing a Page for Printing" on page 7-42
Error Page Template	Optional. Specifies a page template to use for errors that display on a separate page, as opposed to those that display inline.

See Also: "Changing Default Templates in a Theme" on page 7-10 and "Customizing Templates" on page 7-17

Component Defaults Component Defaults identify default templates used when running wizards. You can override these settings on the attributes page for each control or component. Table 4–4 describes component defaults for the current application.

Table 4–4 Component Defaults

Attribute	Description
Calendar	Default calendar template used when you create a new calendar.
Label	Default label template used when you create new page items.
Report	Default report template used when you create new report.
List	Default template used when you create a list.
Breadcrumb	Default template used when you create a breadcrumb.
Button	Default template to be used when you create new buttons that are template controlled.
Region	Default region template used when you create a new region.
Chart Region	Default region template used when you create a chart.
Form Region	Default region template used when you create a form.
Report Region	Default region template used when you create a report.
Tabular Form Region	Default region template used when you create a tabular form.
Wizard Region	Default region template used when you create a new wizard component.
Breadcrumb Region	Default region template used when you create a new breadcrumb.
List Region	Default region template used when you create a new list.

See Also: "Changing Default Templates in a Theme" on page 7-10 and "Customizing Templates" on page 7-17

Configuring Security Attributes

You can provide security for your application by configuring attributes on the Edit Security Attributes page. The Security Attributes you choose apply to all pages within an application.

Topics in this section include:

- Accessing the Edit Security Attributes Page
- Editing Security Attributes

See Also: "Managing Application Security" on page 12-1

Accessing the Edit Security Attributes Page

To access the Edit Security Attributes page:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click Edit Attributes.

   The Application Attributes page appears.
4. Click **Edit Security Attributes**.

The Edit Security Attributes page appears.

Editing Security Attributes

The following sections describe the attributes available on the Edit Security Attributes page.

Topics in this section include:

- Authentication
- Authorization
- Database Schema
- Session State Protection
- Virtual Private Database (VPD)

Authentication **Authentication** is the process of establishing users' identities before they can access an application. Although you define multiple authentication schemes for your application, only one scheme can be current at a time. Table 4–5 describes the attributes available under Authentication.

Table 4–5 Authentication Attributes

Attribute	Descriptions
Home Link	Specifies an URL or procedure that should be run when you run the application. For example, Home Link could contain the relative URL used to locate the application home page. For example, `f?p=6000:600` would specify application 6000 with a home page ID of 600. In this example, the value you enter in Home Link replaces the `#HOME_LINK#` substitution string in application templates. You can also use this attribute to name a procedure. For example, you could create a procedure such as `personal_calendar` which calls an HTML page to serve as the application home. **Note:** Do not use the Home Link attribute to determine the page that displays after authentication. The page that displays after authentication is determined by other components within the application's authentication scheme. **See Also:** "HOME_LINK" on page 3-19
Login URL	Replaces the substitution strings &LOGIN_URL. in HTML or #LOGIN_URL# in templates. **See Also:** "LOGIN_URL" on page 3-19 and "Creating an Authentication Scheme" on page 12-16
Public User	Identifies the Oracle schema (or user) used to connect to the database through the database access descriptor (DAD). The default value is `ANONYMOUS` in environments where the database server version is Oracle Database Express Edition and it is `HTMLDB_PUBLIC_USER` for all other versions of the database server. Once a user has been identified, the Application Express engine keeps track of each user by setting the value of the built-in substitution string `APP_USER`. When `APP_USER` equals this value, the Application Express engine considers the current session to be a public user session. The Application Express engine supports the following built-in display conditions: ▪ USER_IS_PUBLIC_USER ▪ USER_IS_NOT_PUBLIC_USER If the current application user (`APP_USER`) equals the value of this attribute, then the user is logged on as a public user. Some applications have public (not logged in) and a private (logged in) modes. By determining if the user is the public user, you can conditionally display or hide information. For example, you can show a login button if the user is the public user and a logout link if the user is not a public user. Reference this value using `HTMLDB_APPLICATION.G_PUBLIC_USER`. The Application Express engine also has built in condition types `USER_IS_PUBLIC_USER` and `USER_IS_NOT_PUBLIC`. **See Also:** "HOME_LINK" on page 3-19 and "Understanding Conditional Rendering and Processing" on page 3-2
Authentication Scheme	Click this button to define a new authentication scheme. **See Also:** "Understanding How Authentication Works" on page 12-15 and "Creating an Authentication Scheme" on page 12-16

Authorization Authorization controls user access to specific controls or components based on user privileges. You can specify an authorization scheme for your application, by making a selection from the **Authorization Scheme** list. You can assign only one authorization to an entire application. However, you can assign an authorization scheme to individual pages, page controls (such as a region, a button, or an item), or a shared component (such as a menu, a list, or a tab).

To create a new authorization scheme, click **Define Authorization Schemes**.

An authorization scheme is a binary operation that either succeeds (equals true) or fails (equals false). If it succeeds, then the component or control can be viewed. If it fails, then the component or control cannot be viewed or processed. When you attach an authorization scheme to a page and it fails, an error message displays instead of the page. However, when you attach an authorization scheme to a page control (for example, a region, a button, or an item) and it fails, no error page displays. Instead, the control either does not display or is not processed or executed.

See Also: "Providing Security Through Authorization" on page 12-20

Database Schema Use **Parsing Schema** to specify the database scheme for the current application. Once defined, all SQL and PL/SQL commands issued by the application will be performed with the rights and privileges of the defined database schema.

Session State Protection Enabling Session State Protection can prevent hackers from tampering with URLs within your application. URL tampering can adversely affect program logic, session state contents, and information privacy.

To enable or disable Session State Protection for your application, make a selection from the Session State Protection list. Setting Session State Protection to **Enabled** turns on session state protection controls defined at the page and item level.

To configure Session State Protection, click **Manage Session State Protection.**

See Also: "Understanding Session State Protection" on page 12-3

Virtual Private Database (VPD) A Virtual Private Database (VPD) provides an application programming interface (API) that enables developers to assign security policies to database tables and views. Using PL/SQL, developers can create security policies with stored procedures, and bind the procedures to a table or view by means of a call to an RDBMS package. Such policies are based on the content of application data stored within the database, or are based on context variables provided by the Oracle database. In this way, VPD permits access security mechanisms to be removed from applications and centralized.

The PL/SQL you enter in this field is executed immediately after the user is authenticated. `V('USER')` is accessible from this function. Session state for the current call is not yet initialized when this call is made. If your application does not need to employ VPD to support multiple customers in the same database, leave this attribute null.

See Also: "Providing Security Through Authorization" on page 12-20 and *Oracle Label Security Administrator's Guide*

Configuring Globalization Attributes

In Application Builder you can develop applications that can run concurrently in different languages. A single application can be translated to support different languages. Use the attributes on the Edit Globalization Attributes page to specify globalization options such as the primary application language.

Topics in this section include:

- Accessing the Globalization Attributes Page
- Editing Globalization Attributes

> **See Also:** "Managing Application Globalization" on page 14-1

Accessing the Globalization Attributes Page

To access the Edit Globalization Attributes page:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.

   Application Builder appears.
3. Click Edit Attributes.

   The Application Attributes page appears.
4. Click **Edit Globalization Attributes**.

   The Edit Globalization Attributes page appears.

Editing Globalization Attributes

The following sections describe the attributes available on the Edit Globalization Attributes page.

Application Primary Language Identifies the language in which an application is developed. This language is the base language from which all translations are made. For example, suppose application 100 was authored in English, translated into French, and published as application 101. English would be the Application Primary Language.

All modifications to the application should be made to the primary language specified here.

Application Language Derived From Specifies how Application Builder determines or derives the application language. The application primary language can be static, derived from the Web browser language, or determined from a user preference or item. The database language setting also determines how the date is displayed and how certain information is sorted.

This option enables you to disable browser derived language support. You also have the option of having the application language derived from an application preference.

Automatic CSV Encoding Automatic CSV Encoding controls the encoding of all CSV report output in an application. The default value for Automatic CSV Encoding is **No**. If Automatic CSV Encoding is set to **Yes**, CSV report output will be properly converted to a character set compatible with localized desktop applications. The character set for the CSV encoding is determined by the Application Language Derived From setting.

The encoding of pages in Application Builder is determined by the character set of the database access descriptor (DAD) used to access the Database Home Page. For example, if the character set of the Database Access Descriptor is AL32UTF8, all pages in all applications in the Database Home Page user interface will be encoded in UTF-8.

By default, the CSV output from report regions is encoded in the same character set as the Database Access Descriptor. However, some desktop spreadsheet applications require that the data is encoded in the client desktop operating system character set. In the case of multibyte data, the CSV output from report regions will often appear corrupted when opened by a desktop spreadsheet application. This is because the CSV output is encoded differently that what is required by the desktop application. Enabling Automatic CSV Encoding resolves this issue.

For example, if the user's language preference for an application is `de`, the CSV data will be encoded in `Western European Windows 1252`, regardless of the Database Access Descriptor character set setting. If the user's language preference is `zh-cn`, the CSV data will be encoded in Chinese GBK.

About the Page Definition

A page is the basic building block of an application. Each page can have buttons and fields (called items) and application logic (or processes). You can branch from one page to the next using conditional navigation; perform calculations (called computations); perform validations (such as edit checks); and display reports, calendars, and charts. You view, create, and edit the controls that define a page by accessing the Page Definition.

Topics in this section include:

- Accessing a Page Definition
- Understanding the Page Definition

See Also: "Editing a Page Definition" on page 4-19

Accessing a Page Definition

You can view, create, and edit the controls that define a page by accessing the Page Definition.

To access the Page Definition for an existing page:

1. On the Database Home Page, click the **Application Builder** icon.

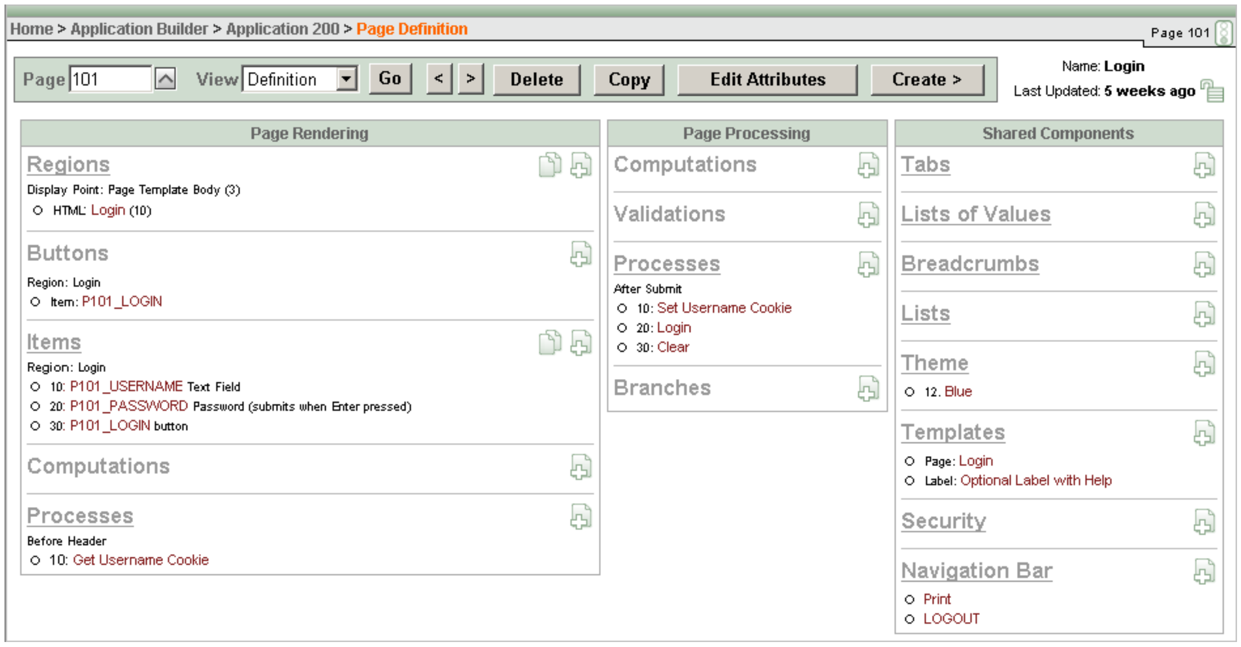
   Application Builder home page appears.

2. Select an application.

   The Application home page appears.

3. Select a page.

   The Page Definition appears.



Home > Application Builder > Application 200 > Page Definition

Page 101

Page 101 View Definition Go < > Delete Copy Edit Attributes Create >

Name: **Login**
Last Updated: **5 weeks ago**

Page Rendering

Regions
Display Point: Page Template Body (3)
- HTML: Login (10)

Buttons
Region: Login
- Item: P101_LOGIN

Items
Region: Login
- 10: P101_USERNAME Text Field
- 20: P101_PASSWORD Password (submits when Enter pressed)
- 30: P101_LOGIN button

Computations

Processes
Before Header
- 10: Get Username Cookie

Page Processing

Computations

Validations

Processes
After Submit
- 10: Set Username Cookie
- 20: Login
- 30: Clear

Branches

Shared Components

Tabs

Lists of Values

Breadcrumbs

Lists

Theme
- 12. Blue

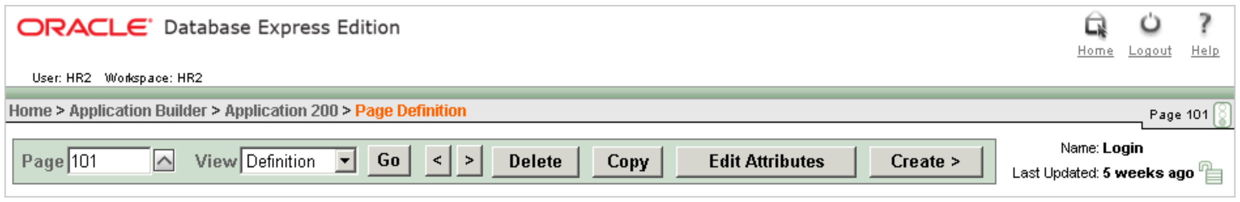
Templates
- Page: Login
- Label: Optional Label with Help

Security

Navigation Bar
- Print
- LOGOUT

Understanding the Page Definition

A breadcrumb trail displays at the top of each Page Definition. Breadcrumb trails appear on every page in the Database Home Page user interface. Each breadcrumb entry indicates your location relative to other pages in the current application and functions as an alternate navigation path. You can instantly go to another page by clicking a breadcrumb entry.



ORACLE Database Express Edition

Home Logout Help

User: HR2 Workspace: HR2

Home > Application Builder > Application 200 > Page Definition

Page 101

Page 101 View Definition Go < > Delete Copy Edit Attributes Create >

Name: **Login**
Last Updated: **5 weeks ago**

The current page identification number (or ID) displays on the far right side of the page, next to a small icon that resembles a traffic light. This icon is called the Run Page icon. Clicking this icon runs the current page and renders it into viewable HTML.

A page navigation bar appears next. Options available on the page navigation bar include:

- **Page.** Displays the current page identification number (ID). To view another page directly, enter the page ID in the Page field and click **Go**. To access the previous or next page, click the arrow buttons.
- **View.** Controls the current page view. To view an alternative report, make a selection from the list and click **Go**. See "Linking to Other Pages from the Page Definition" on page 4-38.
- **Previous and Next.** These buttons resemble less than (<) and greater than (>) signs. Click these buttons to move to the previous or next page ID.
- **Delete**. Deletes the current page.
- **Copy.** Creates a copy of the current page. You specify a new page ID and page name.

- **Edit Attributes.** Links to Page Attributes. Use this page to edit high-level page attributes such as the page name, an optional name alias, and view information about defined tab sets, specified templates, and defined authorization schemes. See "Editing Page Attributes" on page 4-20.
- **Create.** Links to a wizard for creating a new page. See "Creating a Page from the Page Definition" on page 5-9.

The page name and last update date display to the right of the page navigation bar.

> **See Also:** "Adding Pages to an Application" on page 5-8 and "Running a Page or Application" on page 5-11

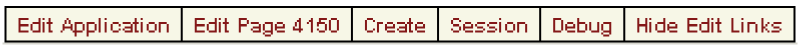
The center of every Page Definition is divided into three sections:

- **Page Rendering** lists user interface controls and logic that execute when the page is rendered.
- **Page Processing** lists list logic controls (such as computations and processes) that are evaluated and executed when the page is processed.
- **Shared Components** lists common components that can be called from multiple pages in an applications.

Using the Developer Toolbar

The Developer toolbar offers a quick way to edit the current page, create a new page, region, or page control, view session state, or turn edit links on an off. You can control whether the Developer toolbar displays by changing the Availability Status on the Edit Application Attributes page.

> **See Also:** "Editing Application Attributes" on page 4-7 for information on the Status list



The Developer toolbar consists of the following links:

- **Edit Application** links you to the Application home page. See "About the Application Builder Home Page" on page 4-2.
- **Edit Page** accesses the Page Definition for the current running page. See "Editing a Page Definition" on page 4-19.
- **Create** links to a wizard for creating a new page, region, page control (item, button, branch, computation, process, or validation), or a shared control (navigation bar icon, tab, list of values, list, or breadcrumb). See "Creating a Page from the Developer Toolbar" on page 5-10.
- **Session** displays a new window containing session state information for the current page. See "Viewing Session State" on page 3-5.
- **Debug** toggles the page between Debug and No Debug mode. See "Accessing Debug Mode" on page 10-2.
- **Show Edit Links** toggles between **Show Edit Links** and **Hide Edit Links**. Clicking **Show Edit Links** displays edit links next to each object on the page that can be edited. Each edit link resembles two colons (::) and appears to the right of navigation bar items, tabs, region titles, buttons, and items. Clicking on the link displays another window in which to edit the object.

Editing a Page Definition

A page is the basic building block of an application. Each page has page ID, a name, and typically some text attributes such as a header, title and footer. You add content to your page by creating page controls (regions, items, and buttons). Page templates and page region templates control the exact look and feel of each page.

Topics in this section include:

- Accessing a Summary View of Controls, Components, and Application Logic
- Copying or Creating a Control or Component
- Editing Page Attributes
- About Page Rendering Controls
- About Page Processing Controls
- About Shared Components
- Understanding Page Computations
- Understanding Validations
- Understanding Page Processes
- Understanding Branches

See Also: "About the Page Definition" on page 4-16

Accessing a Summary View of Controls, Components, and Application Logic

Each Page Definition serves a a central navigation point for all the controls, components, and application logic that define a page.

You can access a summary view of all defined controls or components by selecting the title (for example, Regions, Button, Items, Computations, Processes, and so on). For example, selecting **Regions** displays a summary report of all currently defined regions on the current page. You can use this summary view to:

- Edit the multiple attributes at once by making new selections from the available fields and select lists.
- Link to a definition page by clicking the **Edit** icon.

You can access additional summary views by clicking the buttons at the top of each page. To save your edits to any summary view, click **Apply Changes**.

You can also view the attributes of a specific control or component by selecting its name on the Page Definition. For example, suppose your Page Definition contains a region named *Customers*. Clicking the region name Customers would display an attributes page for that region.

Copying or Creating a Control or Component

You can copy or create new controls or components by clicking the Copy and Create icons. The Create icon resembles a plus (+) sign that overlaps a small page. Click the Create icon to create a new control or component.

The Copy icon resembles two small overlapping pages. Click the Copy icon to make a copy of an existing control or component.

Editing Page Attributes

Page attributes only apply to a specific page. You access page attributes from the Page Definition.

To edit page attributes:

1. Navigate to the Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. At the top of the page, click **Edit Attributes**.

   The Page Attributes page appears. Required values are marked with a red asterisk (*).

The topics that follow describe the specific sections of the Page Attributes page.

See Also: "Adding Pages to an Application" on page 5-8 for information about creating a new page

Name

Use these attributes to define general attributes for the current page such as a page name, an optional alphanumeric alias, and associated page groups. Table 4–7 describes these attributes.

Table 4–6 Page Attributes: Name

Attributes	Descriptions
Name	Identifies the name of the current page for application developers. This name is used in numerous pages and reports, along with the page ID and page title.
Page Alias	Enter an alphanumeric alias for this page. This alias must be unique within the current application. For example, if you were working on page 1 of application 100, you could create an alias called `home`. You could then access this page from other pages using the following `f?p` syntax: `f?p=100:home`
Group	Identify the page group you would like to associate with this page. Page groups do not affect functionality, but help developers manage the pages within an application. **See Also:** "Grouping Pages" on page 5-12

Display Attributes

Use these attributes to define general display attributes for the current page such as the selected page template, standard tab set, title, and cursor focus. Table 4–7 describes these attributes.

Table 4–7 Page Attributes: Display Attributes

Attributes	Descriptions
Page Template	Select a page template to control the appearance of this page. Making a selection here overrides the default page template defined within the current theme. **See Also:** "Changing Default Templates in a Theme" on page 7-10
Standard Tab Set	Select a standard tab set to be used for this page. A standard tab set is associated with a specific page and page ID. You can use standard tabs to link users to a specific page. **See Also:** "Creating Tabs" on page 6-5
Title	Enter a title to display in the title bar of the browser window. The Application Express engine uses the title you specify here in place of the `#TITLE#` substitution string used in the page template. This title is inserted between the HTML tag `<TITLE></TITLE>`.
Cursor Focus	Select whether or not you want the cursor focus to be placed in the first field on the page. Select **Do not focus cursor** if you do not want to include JavaScript.

Header and Footer

Use these attributes to define page header, body header, body footer, and page footer text. Table 4–8 describes these attributes.

Table 4–8 Page Header, Footer and Text Attributes

Attribute	Description
Header Text	Enter the text of the HTML to display after the page template header and before page template body.
Body Header	Enter the text of the HTML to display before showing regions. Displays before the page template `#BOX_BODY#` substitution string.
Footer	Enter the text of the HTML to display after page template body and before page template footer.

HTML Header

Use this attribute to replace the `#HEAD#` substitution string in the page template header. The values entered here are inserted after the HTML `<HEAD>` tag. Common uses of these attributes:

- Code page-specific inline cascading style classes
- Add additional style sheets for a specific page
- Code page-specific JavaScript
- Code page-specific meta tag page refresh

On Load

Use this attribute to add events when the page is being loaded, such as calls to JavaScript. In the Page HTML Body Attribute, enter JavaScript or text to be substituted for your page template's `#ONLOAD#` substitution string. To use this feature, your page template must include the `#ONLOAD#` substitution string.

You can use the Page HTML Body attribute to write into the contents of the opening `<body>` tag. A typical page template might use `#ONLOAD#` within the opening `<body>` tag as shown in the following example:

```
<html>
<head>
...
</head>
<body #ONLOAD# >
```

See Also: "Incorporating JavaScript into an Application" on page 5-65

Security

Use these attributes to specify an authorization scheme, authentication, and URL access protection for the current page. Table 4–9 describes these attributes

Table 4–9 Page Attributes: Security

Attribute	Description
Authorization Scheme	Select an authorization scheme to be applied to the page. Authorization schemes are defined at the application level and can be applied to many elements within the application. An authorization scheme is evaluated either once for each application session (at session creation), or once for each page view. If the selected authorization scheme evaluates to true, then the page displays and is subject to other defined conditions. If it evaluates to false, then the page will not display and an error message displays. **See Also:** "Providing Security Through Authorization" on page 12-20
Authentication	Specifies whether this page has been defined as public or requires authentication. If a page is identified as public, the page can be viewed before authentication. This attribute only applies if the application uses SCHEME authentication. The application's page sentry function can access this page attribute to identify pages that do not require prior authentication to view. The implementation of the authentication scheme's page sentry function determines if this attribute has any effect. **See Also:** "Establishing User Identity Through Authentication" on page 12-14

Duplicate Submission

Use the **Allow duplicate page submissions** list to specify whether or not users may process a page multiple times in a row. Set this attribute to **No** to prevent duplicate page submissions from being processed multiple times.

Examples of duplicate page submissions include:

- A user clicks the Submit button multiple times.

- You create a branch of type Branch to Page, and the user clicks the browser reload button.

Configuration

Build options allow you to enable or disable functionality. Most application attributes have a build option attribute. Do not specify a build option for the current page unless you plan to exclude the page in certain configurations.

Build options have two possible values: `INCLUDE` and `EXCLUDE`. If you specify an attribute as being included, then the Application Express engine considers it part of the application definition at run time. Conversely, if you specify an attribute as being excluded, then the Application Express engine treats it as if it does not exist

> **See Also:** "Using Build Options to Control Configuration" on page 11-15

On Error Text

Use this attribute to specify the error text that displays in the `#NOTIFICATION_MESSAGE#` template substitution string in the event an error occurs on the page.

> **See Also:** "Page Templates" on page 7-28

Help

Use this attribute to enter Help text for the current page.

Help text is displayed using a help system that you must develop. To show the Help for a specific page, call the `HTMLDB_APPLICATION.HELP` procedure from a page that you create for displaying Help text. For example, you could use a navigation bar icon similar to:

```
f?p=4000:4600:&APP_SESSION.::&DEBUG::LAST_STEP:&APP_PAGE_ID
```

Page-level help supports shortcuts using the following syntax:

```
"SHORTCUT_NAME"
```

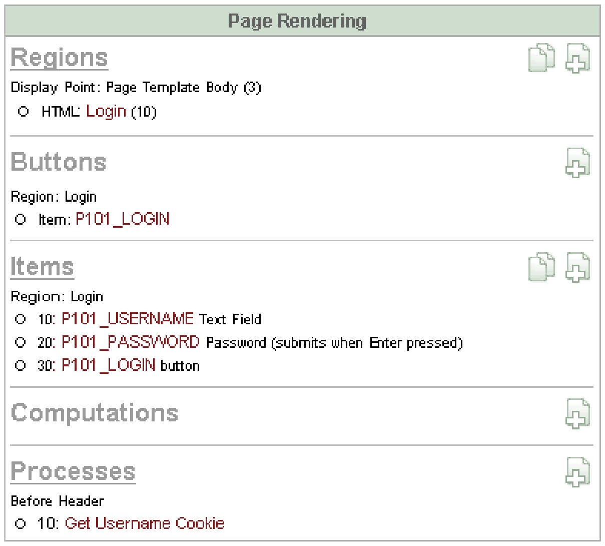
> **See Also:** "Creating a Help Page" on page 5-68 and "Using Shortcuts" on page 5-63

Comments

Use this attribute to record developer comments about the current page. These comments never display when the application is running.

About Page Rendering Controls

Use the Page Rendering section of the Page Definition to specify attributes for defined regions, buttons, items, page rendering computations, and page processes.


Page Rendering

Regions

Display Point: Page Template Body (3)

- HTML: Login (10)

Buttons

Region: Login

- Item: P101_LOGIN

Items

Region: Login

- 10: P101_USERNAME Text Field
- 20: P101_PASSWORD Password (submits when Enter pressed)
- 30: P101_LOGIN button

Computations

Processes

Before Header

- 10: Get Username Cookie

Topics in this section include:

- Regions
- Buttons
- Items
- Computations
- Processes

> **See Also:** "About the Page Definition" on page 4-16, "Accessing a Summary View of Controls, Components, and Application Logic" on page 4-19 and "Copying or Creating a Control or Component" on page 4-19

Regions

A region is a area on a page that serves as a container for content. Each page can have any number of regions. The content of a region is determined by the region source. For example, a region may contain a report based on a SQL query you define, or it may contain static HTML.

You control the appearance of a region through a specific region template. You can use regions to group page controls (such as items or buttons). You can also create simple regions that do not generate additional HTML, or create elaborate regions that frame content within HTML tables or images.

> **See Also:**
>
> - "Customizing Regions" on page 7-2 for information about creating specific types of regions
> - *Oracle Database Application Developer's Guide - Fundamentals* for information about developing Web applications with PL/SQL
> - *Oracle Database PL/SQL Packages and Types Reference* for information about htp packages

Buttons

As you design your application, you can use buttons to direct users to a specific page or URL, or submit a page. When you submit a page, the Application Express engine posts or process information. A button can be implemented as an HTML button, an image, or by using a template. Buttons can be placed in predefined region template positions or among items in a form.

See Also: "Creating Buttons" on page 5-47

Items

Items are HTML form elements such as text fields, select lists, and check boxes with an associated session state. Item attributes affect the display and behavior of items on a page. For example, these attributes can impact where a label displays, how large an item will be, and whether or not the item will display next to, or below the previous item.

There are two categories of items: page items and application items. Page items are placed on a page and have associated user interface properties, such as Display As, Label, and Label Template. Application items are not associated with a page and therefore have no user interface properties. An application item can be used as a global variable.

See Also: "Creating Items" on page 5-51

Computations

Computations are units of logic used to assign session state to items. You can use computations to assign a value to an identified item when a page is submitted or displayed.

See Also: "Creating a Page Computation" on page 4-29 and "Understanding Application Computations" on page 4-47

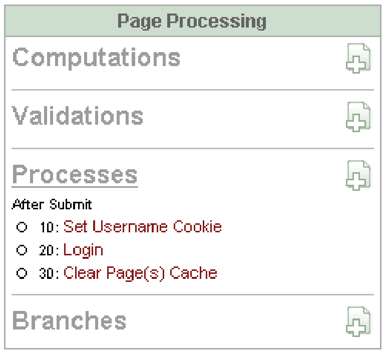
Processes

Processes are logic controls used to execute data manipulation language (DML) or PL/SQL. For example, you can use a process to populate session state at the time a page is rendered, to execute some type of logic (for example, using PL/SQL), or to make a call to the rendering engine. Typically a process performs an action. A process may be hand coded PL/SQL, or the invocation of a predefined process.

See Also: "Understanding Page Processes" on page 4-35 and "Understanding Application Processes" on page 4-44

About Page Processing Controls

Use the Page Processing section of the Page Definition to specify application logic such as computations, validations, processes, and branches. In general, the Application Express engine runs this logic in the order it appears on the Page Definition.



Topics in this section include:

- Computations
- Validations
- Processes
- Branches

See Also: "About the Page Definition" on page 4-16, "Accessing a Summary View of Controls, Components, and Application Logic" on page 4-19, and "Copying or Creating a Control or Component" on page 4-19

Computations

Computations are units of logic used to assign session state to items and are executed at the time the page is processed.

See Also: "Understanding Page Computations" on page 4-29

Validations

Validations enable you to create logic controls to verify whether user input is valid. For example, a validation can check whether or not a value has been entered into a mandatory field

See Also: "Understanding Validations" on page 4-32

Processes

Processes are logic controls used to execute data manipulation language (DML) or PL/SQL. Processes are executed after the page is submitted.

See Also: "Understanding Page Processes" on page 4-35

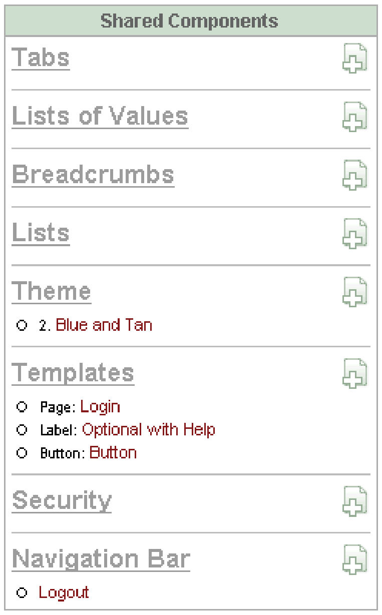
Branches

Branches enable you to create logic controls that determine how the user navigates through the application.

See Also: "Understanding Branches" on page 4-37

About Shared Components

Shared Components section of the Page Definition contains common elements that can display or be applied on any page within an application.


Shared Components

Tabs

Lists of Values

Breadcrumbs

Lists

Theme
- 2. Blue and Tan

Templates
- Page: Login
- Label: Optional with Help
- Button: Button

Security

Navigation Bar
- Logout

See Also: "About the Shared Components Page" on page 4-40

Topics in this section include:

- Tabs
- Lists of Values
- Breadcrumbs
- Lists
- Theme
- Templates
- Security
- Navigation Bar

See Also: "About the Page Definition" on page 4-16, "Accessing a Summary View of Controls, Components, and Application Logic" on page 4-19, and "Copying or Creating a Control or Component" on page 4-19

Tabs

Tabs are an effective way to navigate between pages of an application. Application Builder includes two different types of tabs: standard tabs and parent tabs.

An application having only one level of tabs uses a standard tab set. A standard tab set is associated with a specific page. You can use standard tabs to link users to other pages within your application. A parent tab set functions as a container to hold a group of standard tabs. Parent tabs give users another level of navigation as well as context (or sense of place) within the application.

See Also: "Creating Tabs" on page 6-5

Lists of Values

A list of values (LOV) is a static or dynamic definition used to display a specific type of page item, such as a radio group, check box, or select list. LOVs can be static (that is, based on a set of predefined display and return values) or dynamic (based on SQL queries that select values from tables). Once created, a LOV can then be referenced by one or more page items.

You define LOVs at the application level by running the LOV Wizard and adding them to the List of Values repository.

See Also: "Creating Lists of Values" on page 5-60

Breadcrumbs

A breadcrumb is a hierarchical list of links that is rendered using a template. For example, you can display breadcrumbs as a list of links or as a breadcrumb path.

See Also: "Creating Breadcrumbs" on page 6-9

Lists

A list is a collection of links that is rendered using a template. For each list entry, you specify display text, a target URL, and other attributes that control when and how the list entry displays. You control the display of the list and the appearance of all list entries by linking the list to a template.

See Also: "Creating Lists" on page 6-13

Theme

A theme is a named collection of templates that defines the application user interface. Each theme contains templates for every type of application component and page control, including individual pages, regions, reports, lists, labels, menus, buttons, and list of values.

See Also: "Managing Themes" on page 7-8

Templates

Templates control the look and feel of the pages in your application. As you create your application, you specify templates for pages, regions, reports, lists, labels, menus, buttons, and popup list of values. Groups of templates are organized into named collections called themes.

See Also: "Customizing Templates" on page 7-17

Security

You can provide security for your application by specifying an authorization scheme. Authorization is a broad term for controlling access to resources based on user privileges.

See Also: "Providing Security Through Authorization" on page 12-20

Navigation Bar

Use a navigation bar to link users to various pages within an application. Typically a navigation bar is used to enable users to log in and log out or link to Help text. The location of a navigation bar depends upon the associated page template. A navigation bar icon enables you to display a link from an image or text. When you create a navigation bar icon you can specify an image name, text, a display sequence, and a target location (a URL or page).

> **See Also:** "Creating a Navigation Bar Entry" on page 6-1

Understanding Page Computations

Use page computations to assign a value to an identified item when a page is submitted or displayed. You can also use application-level computations to assign values to items. Most application-level computations are performed for every page in an application. In contrast, computations created at the page-level only execute when that page is rendered or processed.

> **See Also:** "Understanding Application Computations" on page 4-47

Topics in this section include:

- Creating a Page Computation
- Understanding Computation Points and Computation Syntax
- Editing Page Computation Attributes

Creating a Page Computation

You create a page computation by running the Create Page Computation Wizard. For each computation, specify the item for which you are creating the computation as well as a computation type.

> **See Also:** "Computations" on page 4-25

To create a page computation:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Computations, click the **Create** icon.
3. For Item Location, select the where the computation will execute and click **Next**. Location options include:
   - Item on this Page
   - Item on Another Page
   - Application Level Item
4. For Item, select the item and computation point at which you would like to perform the computation:
   a. Compute Item - Select the item the computation will update.
   b. Sequence - Select the order of evaluation.

   c. Computation Point - Select the point at which the computation executes. The computation point **On New Instance** executes the computation when a new session (or instance) is generated.
   d. Computation Type - Select the method of computation you want to create.
   e. Click **Next**.
5. In Computation, enter a computation which corresponds to the selected computation type and click **Next**.
6. On Condition, you can choose to make the computation conditional. To make a computation conditional, make selection from the Condition Type list and enter text in the expression fields.
7. Click **Create**.

Understanding Computation Points and Computation Syntax

A good example of using computations can be illustrated by a page containing form fields for entering phone numbers. In this example, the phone number is stored in one database column, however the data entry form breaks the phone number into three components: area code, prefix, and line number. In this example, the page items are called `P10_AREA_CODE`, `P10_PREFIX`, and `P10_LINE_NUMBER`.

Next, suppose you need to combine the values stored in these items into a single string. You could accomplish this by using an After Submit computation and store the combined values in an item called `P10_PHONE_NUMBER`.

To create a computation to store the combined values of `P10_AREA_CODE`, `P10_PREFIX`, and `P10_LINE_NUMBER` in new items:

1. Navigate to the appropriate Page Definition:
2. Create a new item named `P10_PHONE_NUMBER` to store the combined values of `P10_AREA_CODE`, `P10_PREFIX`, and `P10_LINE_NUMBER`. See "Creating a Page-Level Item" on page 5-51.
3. Under Computations, click the **Create** icon.
4. For Location, select **Item on this Page** and click **Next**.
5. For Compute Item, select **P10_PHONE_NUMBER.**
6. For Sequence, select the order of evaluation.
7. For Computation, you have the option of creating one of the following computation types:
   a. Static Assignment:
      – For Computation Type, select **Static Assignment** and click **Next**.
      – Enter the following computation:

```
(&P10_AREA_CODE.) &P10_PREFIX.-&P10_LINE_NUMBER.
```

      – Click **Next**.
   b. PL/SQL Function Body:
      – For Computation Type, select **PL/SQL Function Body** and click **Next**.
      – Enter the following computation:

```
DECLARE
l_return_value  VARCHAR2(300) DEFAULT NULL;
```

```
BEGIN
    l_return_value :=
'('||:P10_AREA_CODE||')'||:P10_PREFIX||'-'||:P10_LINE_NUMBER;
RETURN l_return_value;
END;
```

   – Click **Next**.

   c. SQL Query:

   – For Computation Type, select **SQL Query** and click **Next**.

   – Enter the following computation:

```
SELECT '('||:P10_AREA_CODE||')'||:P10_PREFIX||'-'||:P10_LINE_NUMBER
FROM DUAL
```

   – Click **Next**.

   d. PLSQL Expression:

   – For Computation Type, select **PLSQL Expression** and click **Next**.

   – Enter the following computation:

```
'('||:P10_AREA_CODE||')'||:P10_PREFIX||'-'||:P10_LINE_NUMBER
```

   – Click **Next**.

8. Click **Create**.

Editing Page Computation Attributes

Once you create a computation, you can edit it on the Edit Page Computation page.

To edit a page computation:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Computations, select the computation name.

   Edit Page Computation page appears.
3. Click **Apply Changes**.

Editing the Computation Point and Source You control when a computation executes under the Computation Point attributes by specifying a sequence and a computation point. The computation point **On New Instance** executes the computation when a new session (or instance) is generated.

Under Source, enter an expression or query to compute an item's value. In the event a computation fails, you can optionally define an error message in the Computation Error Message field.

Creating Conditional Computations You can make a computation conditional by making a selection from the Condition Type list and entering text in the expression fields.

Understanding Validations

You can define a validation declaratively by selecting a validation method. You enter the actual validation edit check in the Validation Messages field. Be aware that if a validation fails, subsequent page processes or computations will not occur. Also remember that the validation you enter must be consistent with the validation type you selected. For more information about validation types, see online Help.

Topics in this section include:

- Creating a Validation
- Defining How Validation Error Messages Display
- Processing Validations Conditionally

Creating a Validation

To create a new validation:

Note: Text entered for validations may not exceed 3,950 characters.

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

   The Page Definition appears.
2. Under Validations, click the Create icon.

   The Create Validations Wizard appears.
3. Select a validation level. **Item level validations** are specific to a single item. **Page level validations** do not apply to any single item, but apply to an entire page.
4. If you selected **Item level validation**, select the item to be validated and click **Next**.
5. Select a validation method as described in Table 4–10.

Table 4–10 Validation Methods

Validation Method	Descriptions

SQL	Compares item values to data in the database.
	For example, you can use a SQL validation to verify whether a last name typed into a field exists in the database. In the following Exists SQL validation, the field is named `P1_LAST_NAME` and the table is named `customers`.
	`SELECT 1 FROM customers`
	`WHERE last_name = :P1_LAST_NAME`

Table 4–10 (Cont.) Validation Methods

Validation Method	Descriptions

PL/SQL	Useful if you need complex logic to validate entered data.
	
	For example, suppose you need to create a validation for an address form that requires the user enter a province if the address is not in the United States. You could create the validation as a Function Returning Boolean, using the following PL/SQL:
	
	`BEGIN`
	`  IF :P1_COUNTRY = 'US' AND :P1_PROVINCE IS NULL THEN`
	`     RETURN FALSE;`
	`  ELSE`
	`    RETURN TRUE;`
	`  END IF;`
	`END;`
	
	You could also create the same validation implemented as a PL/SQL Expression as follows:
	
	`NOT (:P1_COUNTRY='US' AND  :P1_PROVINCE IS NULL);`

Item Level Null	Checks if an item's value in session state is null. For example, you could validate that the user enters a value in a field by creating an item validation and then selecting the validation method **Item Not Null**.

Item String Comparison	Compares the value of an item to a specific string.
	
	There are several string comparison validations that compare the value of an item to a literal string. For example, you select the validation type **Item in Expression 1 is contained in Expression 2** to validate a user entry in a field against a list of values you provide.
	
	In Expression 1, enter the name of item you want to validate without a colon. For example:
	
	`P1_VALUE`
	
	In Expression 2, enter a string a values you want to validate against. For example:
	
	`ABC/DEF/GHI`

Regular Expression	Regular expressions provide a method to describe text patterns. Use a Regular Expression validation to perform data validation.
	
	For example, you could use the following regular expression validation to verify that a string of entered data always consists of groups of six numbers separated by commas and followed by a comma:
	
	`^([[:digit:]]{6},)+$`
	
	This regular expression would find the following entries valid:
	
	`123456,654321,`
	
	`123456,`
	
	`123456,123456,654321,`
	
	However, the following would not be valid:
	
	`123456,12345`
	
	`12345`

6. For SQL, PL/SQL, and Item String Comparison validations, select the type of validation you want to create and click **Next**.
7. Specify the sequence and validation name and click **Next**.
8. Depending upon the validation method, enter the validation or message text that displays if the validation fails. Click **Next**.
9. Define conditions that apply to this validation and click **Create**.

   See Also: "Validating User Input in Forms" on page 5-33

Defining How Validation Error Messages Display

You can choose to have validation error messages display inline (that is, on the page where the validation is performed) or on a separate error page.

To define how a validation error message displays:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Validations, select the appropriate validation.

   The attributes page for the validation appears.
3. Scroll down to Error Message.
4. In Error Message, enter your error message text.
5. From Error message display location, select a display location.

   This attribute identifies where a validation error message displays. **Validation error messages** can display on an error page or inline within the existing page. **Inline error messages** can display in a notification area (defined as part of the page template) or within the field label.

   To create a hard error that stops all processing (including validations), you must display the error on an error page.
6. If you select Inline with Field or Inline with Field and in Notification, you need to associate an item with the error message. To associate an item with the error message, select the item from the Associated Item list.
7. Click **Apply Changes**.

Processing Validations Conditionally

You can control when and if a validation is performed under Conditions.

To create a condition for an existing validation:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.

   d. Select a page.

      The Page Definition appears.

2. Under Validations, select the appropriate validation.

   The attributes page for the validation appears.

3. Scroll down to Conditions.
4. To have a validation performed when a user clicks a particular button, make a selection from the When Button Pressed list.
5. Make a section from the Condition Type list.
6. Depending upon the selected Condition Type, enter values in the Expression attributes. The validation will be rendered or processed if the specified condition is met.
7. Click **Apply Changes**.

Understanding Page Processes

A page process performs an action at a specified point during the rendering or submission of the page. For example, you can create a page process to execute logic or to make a call to the Application Express engine. A page process is a unit of logic that runs when a specific event occurs, such as loading or submitting a page.

From a functional perspective, there is no difference between page-level and application-level processes. The difference between these two process types is where the process is defined, that is at the page level or at the application level.

See Also: "Understanding Application Processes" on page 4-44

Topics in this section include:

- Creating a Page Process
- Editing Process Attributes

Creating a Page Process

You create a process by running the Create Process Wizard. During the wizard, you define a process name, specify a sequence, the point at which process will execute, and select a process category. You can change nearly all of these attributes on the Edit Page Process page.

To create a new process:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Processes, click the **Create** icon.
3. Select a category. Table 4–11 describes available page process categories.

Table 4–11 Process Categories

Process Category	Description
Data Manipulation	Data Manipulation process types are frequently used by wizards to support data manipulation language (DML) actions. Application Builder supports the following declarative data manipulation processes: ■ Select **Automatic Row Fetch** and **Automatic Row Processing (DML)** to create an automatic data manipulation language (DML) process ■ Use **Multi Row Update** and **Multi Row Delete** in conjunction with tabular forms. ■ Use **Add Rows to Tabular Form** in conjunction with a tabular form
Close Popup Window	Upon execution, this process type closes a popup window and refreshes the calling window.
Form Pagination	Implements pagination through the detail records associated with a master detail form. Most often used in master detail forms (such as in the Master Detail Wizard), this process checks the master table to determine which set of detail records you are in and determines what the next detail record should be. **See Also:** "Building a Master Detail Form" on page 5-29
On Demand	Creates an application-level process that can only be executed when called from a specific page. When you create this process type at the page-level, you are creating reference to an existing application-level process. **See Also**: "About On Demand Application Processes" on page 4-44
PL/SQL	Runs the PL/SQL you provide. Use this process type to execute a block of PL/SQL entered directly into the process or to simply call an existing API.
Reset Pagination	In Report regions, resets pagination back to the first result set. The Application Express engine keeps track of where the user is within a given result set. This process category returns the user to the beginning result set. In other words, this category resets the counters associated with the report region to return the first part of the result set the next time the result set displays.
Session State	Sets the values of existing session state items to null. Select this process type to clear the cache for applications, sessions, or items as well as to clear existing user preferences. **See Also:** "Managing Session State Values" on page 3-6 and "Managing User Preferences" on page 8-5
Web Services	Implements a Web Service as a process on a page. Running the process submits a request to the service provider. **See Also:** "Invoking a Web Service as a Process" on page 13-20

4. Follow the on-screen instructions.

Editing Process Attributes

Once you create a process, you can control when the process executes and what the process does by editing attributes on the Edit Page Process page.

To edit an existing page process:

1. Navigate to the appropriate Page Definition:

   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.

2. Select the process name.

   The Edit Page Process page appears.

Changing Processing Points and Source You control when a process executes by specifying a sequence number and a process point under Process Point. You can prevent a process from running during subsequent visits to a page by selecting one of the following options under Run Process:

- Once for each page visit
- Once for each session or when reset

Enter the appropriate code for PL/SQL process types. For PL/SQL anonymous block processes, enter the appropriate code under **Process**. For Clear Cache processes, enter the appropriate code under **Source**. In the event a process fails, you can optionally define an error message in the Process Error Message field.

Creating Conditional Processes You can make a process conditional by selecting a condition type and entering an expression under Conditional Processing.

Additionally, you can also make a selection from the When Button Pressed attribute. When you select a button from this list, the process only executes if a user clicks the selected button.

Understanding Branches

A branch is an instruction to go to a specific page, procedure, or URL. For example, you can branch from page 1 to page 2 after page 1 is submitted.

You create a new branch by running the Create Page Branch Wizard and specifying Branch Point and Branch Type. The Branch Type defines the type of branch you are creating. For more information about Branch Types, see online Help.

Topics in this section include:

- Defining a Branch Point and Action
- Branching Conditionally

Defining a Branch Point and Action

When you click a standard tab in an application, the Application Express engine sets session state, executes computations, and then links you to the target page. It does not run any processes or explicitly defined branches. In cases where the page is submitted without clicking a tab, the Application Express engine explicitly defines branches to direct users to a subsequent page.

You can control when a branch executes by making a selection from the Branch Point list. Available options include:

- **On Submit: Before Computation** - Branching occurs before computations, validations, or processing. Use this option for buttons that do not need to invoke any processing, for example, a Cancel button.

- **On Submit: Before Validation** - Branching occurs after computations, but before validations or processing. If a validation fails, page processing stops, a rollback is issued, and the page displays the error. Because of this default behavior, you do not need to create branches to accommodate validations. However, you may want to branch based on the result of a computation (for example, to a previous branch point).
- **On Submit: Before Processing** - Branching occurs after computations and validations, but before processing. Use this option to branch based on a validated session state, but before performing any page processing.
- **On Submit: After Processing** - Branching occurs after computations, validations, and processing. This option branches to a URL or page after performing computations, validations, and processing. When using this option, remember to sequence your branches if you have multiple branches for a given branch point.
- **On Load: Before Header** - Branching occurs before a page is rendered. This option displays another page instead of the current page or redirects the user to another URL or procedure.

Depending upon the Branch Type you select, you can specify the following additional information in the Action attributes:

- The page ID of the page to which you want to branch
- PL/SQL procedure which ultimately renders a branch target page
- A URL address

Branching Conditionally

Like other controls, branches can be made conditional. To create a conditional branch, make a selection from the Condition Type list, and enter text in the expression fields to implement the condition type you choose.

See Also: "Controlling Navigation Using Branches" on page 6-8

Linking to Other Pages from the Page Definition

You can use the View list on the Page Definition to quickly access the Page Events, Database Object Dependencies, History, Export, Groups, and Referenced Components pages.

Topics in this section include:

- Using the View List on the Page Definition
- About Events
- About Objects
- About History
- About Export
- About Groups
- About Referenced

See Also: "Understanding the Page Definition" on page 4-17

Using the View List on the Page Definition

To access other pages using the View list:

1. Navigate to the Database Home Page.
2. Click the **Application Builder** icon.
3. Select an application.
4. Select a page.

   The Page Definition appears.
5. From the View list, select one of the following and click **Go**:
   - Events. See "About Events".
   - Objects. See "About Objects".
   - History. See "About History"
   - Export. See "About Export".
   - Groups. See "About Groups".
   - Referenced. See "About Referenced".

   See Also: "Understanding the Page Definition" on page 4-17

About Events

Page Events details all currently defined page controls and processes. This page provides a chronological view of how and in what order the Application Express engine renders the page, invokes logic, and runs processes. You can control the amount of information that displays by selecting one of the following buttons:

- **Show All** displays all possible page controls and processes, including those not currently defined.
- **Show Used** displays currently used page controls and processes (Default).

To view details about a specific page control or process, click the appropriate hypertext link. Alternately, you can create new page controls and processes by clicking the small icons to the left of each entry.

To run the current page, click the **Run** icon.

About Objects

The Database Object Dependencies page displays a list of database objects referenced by the current page.

About History

The History page displays a history of recent changes to the currently selected page by developer (or user), application, page ID, modification date, component, and action.

About Export

Use the Export page to export the current page. Remember that some pages may reference shared components. To export all pages within an application, you need to complete an application export.

See Also: "How to Deploy an Application to Another Oracle Application Express Instance" on page 11-4 and "Exporting a Page in an Application" on page 11-6

About Groups

The Groups page displays all pages that are part of the same page group as the current page. Click a page ID to edit the page group. Click a page name to view the page definition.

> **See Also:** "Grouping Pages" on page 5-12

About Referenced

The Referenced Components page lists page components and shared components associated with the current page.

About the Shared Components Page

Shared components are common elements that can display or be applied on any page within an application. Examples of shared components include:

- Logic controls, such as application items, application processes, application computations, Web service references, and build options
- Security controls, such as creating authentication and authorization schemes, enabling session state protection, or configuring Security attributes
- Globalization tasks such as translating an application, translating messages, or configuring globalization attributes
- Navigation controls, such as breadcrumbs, lists, navigation bar entries, tabs, and trees
- User interface elements, such as themes, templates, user interface defaults, lists of values, and shortcuts)
- File management, such as managing cascading style sheets, images, and static files

You can use the tools and wizards on the Shared Components page either at the application-level, or on specific pages.

Topics in this section include:

- Accessing the Shared Components Page
- Understanding the Shared Components Page

Accessing the Shared Components Page

To access the Shared Components page:

1. Navigate to the Database Home Page.
2. Click the **Application Builder** icon.
3. Select an application.
4. On the Application home page, click **Shared Components**.

   The Shared Components page appears.
5. To create a shared component, select the appropriate link.

Understanding the Shared Components Page

The following sections describe each link on the Shared Components page.

Application Items

Application level items do not display, but are used as global variables to the application. Commonly, you set the value of a page-level item using an application or page computations.

> **See Also:** "Creating an Application-Level Item" on page 5-58

Application Processes

Use application processes to run PL/SQL logic:

- At specific points for each page in an application
- As defined by the conditions under which the process is set to execute
- Upon the creation of a new session

Note that **On Demand** processes execute only when called from page-level On Demand process.

> **See Also:** "Understanding Application Processes" on page 4-44

Application Computations

Use application-level computations to assign values to application and page-level items for each page displayed or upon the creation of a new application session. You can also create an application-level computation and execute it conditionally on multiple pages.

> **See Also:** "Understanding Application Computations" on page 4-47

Web Service References

Web service references in Application Builder are based on the Simple Object Access Protocol (SOAP). You can create a reference to a Web service and then incorporate it into an application to process data submitted by a form, or to render output in the form or report.

> **See Also:** "Implementing Web Services" on page 13-14

Build Options

Use build options to conditionally display or process specific functionality within an application. You can use build options to control which features of an application are turned on for each application deployment.

> **See Also:** "Using Build Options to Control Configuration" on page 11-15

Authentication Schemes

Authentication is the process of establishing each user's identity before they can access your application. Authentication may require a user to enter a user name and password or may involve verification of a user’s identity or use of a secure key

> **See Also:** "Establishing User Identity Through Authentication" on page 12-14

Authorization Schemes

Authorization restricts user access to specific controls or components based on predefined user privileges.

> **See Also:** "Providing Security Through Authorization" on page 12-20

Session State Protection

Session State Protection is a built-in functionality that prevents hackers from tampering with the URLs within your application. URL tampering can adversely affect program logic, session state contents, and information privacy.

> **See Also:** "Understanding Session State Protection" on page 12-3

Edit Security Attributes

Use the Edit Security Attributes page to configure general security attributes for all pages within an application.

> **See Also:** "Configuring Security Attributes" on page 4-12

Translate Application

You can develop applications in Oracle Application Express that can run concurrently in different languages. A single Oracle database and Oracle Application Express instance can support an application in multiple languages. Translating an application involves multiple steps. Click this link to access the Translate Application page.

> **See Also:** "About Translating an Application and Globalization Support" on page 14-1 and "Understanding the Translation Process" on page 14-6

Text Message

Text messages are named text strings that can be called from PL/SQL code you write. This PL/SQL can be anonymous blocks within page processes and page regions, or in packages and procedures.

> **See Also:** "Translating Messages Used in PL/SQL Procedures" on page 14-10

Edit Globalization Attributes

You can develop applications that can run concurrently in different languages. Click this link to specify globalization options such as the Application Primary Language and Application Language Derived From attributes.

> **See Also:** "Configuring Globalization Attributes" on page 4-14 and "About Translating an Application and Globalization Support" on page 14-1

Breadcrumbs

Breadcrumbs provide users with hierarchical navigation. A breadcrumb is a hierarchical list of links that display using templates. You can display a breadcrumb as a list of links, or as a breadcrumb path.

> **See Also:** "Creating Breadcrumbs" on page 6-9

Lists

A list is a shared collection of links. You control the appearance of a list through list templates. Each list element has a display condition that enables you to control when it displays.

See Also: "Creating Lists" on page 6-13

Navigation Bar Entries

Navigation bar entries offer users a simple navigation path for moving between pages in an application. The location of a navigation bar depends upon the associated page template. Navigation bar entries can display as a link from an image or text. A navigation bar entry can be an image, an image with text beneath it, or text.

See Also: "Creating a Navigation Bar Entry" on page 6-1

Tabs

Tabs are an effective way to navigate users between pages in an application. You can create two types of tabs: standard tabs or parent tabs. A standard tab set is associated with a specific page and page ID. A parent tab set functions as a container to hold a group of standard tabs.

See Also: "Creating Tabs" on page 6-5

Trees

A tree is an effective way to communicate hierarchical or multiple level data.

See Also: "Creating Trees" on page 6-18

Themes

A theme is a named collection of templates that defines the application user interface.

See Also: "Managing Themes" on page 7-8

Templates

Templates control the look and feel of specific constructs within your application, such as pages, regions, items, and menus.

See Also: "Customizing Templates" on page 7-17

User Interface Defaults

User interface defaults enable you to assign default user interface properties to a table, column, or view within a specified schema. When you create a form or report using a wizard, the wizard uses this information to create default values for region and item properties.

Because user interface defaults are associated with a table, you can use them with applications created using the form and report wizards.

See Also: "Managing User Interface Defaults" on page 9-1

Lists of Values

A list of values (LOV) is a static or dynamic set of values used to display a popup list of values, select list, check box, or radio group.

See Also: "Creating Lists of Values" on page 5-60

Shortcuts

Use shortcuts to avoid repetitive coding of HTML or PL/SQL functions. You can create a shortcut to define a page control such as a button, HTML text, a PL/SQL procedure, or HTML. Once you define a shortcut it is stored in a central repository so you can reference from various locations within your application.

See Also: "Using Shortcuts" on page 5-63

Cascading Style Sheets

Application Builder includes themes that contain templates that reference their own cascading style sheets (CSS). Use the Cascading Style Sheets link to upload cascading style sheets to your database user account.

See Also: "Using Custom Cascading Style Sheets" on page 7-43

Images

Use the Images link to upload images to your database user account.

See Also: "Managing Images" on page 7-45

Static Files

Use the Static Files link to upload static files to your database user account.

See Also: "Managing Static Files" on page 7-48

Understanding Application Processes

Application processes are blocks of PL/SQL logic that are set to run at specific points using the processing of multiple pages of an application. By default, application processes execute at the same point for every page in the application. However, you can apply conditions for specific pages to control when the process executes.

Topics in this section include:

- About On Demand Application Processes
- Application Process Example
- Creating an Application Process
- Accessing Application Processes Reports

About On Demand Application Processes

A special type of application process is the **On Demand** process. An On Demand application process has a Process Point of **On Demand** and executes when called from a page-level On Demand process. On Demand processes are useful when you have PL/SQL logic that you would like to run from different execution points across multiple pages.

See Also: "Creating a Page Process" on page 4-35

Running an On Demand Process from a Page Request

You can have a page request run an On Demand process by using the following syntax:

```
f?p=application_id:page_id:session:APPLICATION_PROCESS=process_id
```

Where:

- `application_id` is the application ID or alphanumeric alias
- `page_id` is the page ID or alphanumeric alias
- `session` is the session ID
- `APPLICATION_PROCESS=process_id` is the keyword `APPLICATION_PROCESS=` followed by either the process ID or an alphanumeric name of an application-level process having a Process Point of On Demand

When you use this syntax, the Application Express engine recognizes the request and processes it using the following rules:

- The page ID in the URL can be any page ID. The page ID is required in the request only as a syntactic placeholder because no specific page is accessed for this type of request.
- The process authorization scheme, the application's authorization scheme, and the process conditions are supported.
- Session state (that is, item names and values) may be set in the URL, but clear cache options are ignored.
- Any failures of authentication, authorization, or process conditions do not result in visible error messages or other indicators of such failures and most often result in a blank page being displayed.
- Specifying the process by name locates the first process with the specified (case-preserved) name.

See Also: "Clearing Session State" on page 3-7

Application Process Example

A shopping cart application is a good example of when you might use an application process. For example, suppose you need to display the contents of a user's shopping cart with each page view. To accomplish this you create a region on page zero of your application that displays the values of the application-level items `TOTAL_CART_ITEMS` and `TOTAL_PURCHASE_PRICE`.

See Also: "Displaying Components on Every Page of an Application" on page 7-2

Instead of writing a process of each page to set the values of `TOTAL_CART_ITEMS` and `TOTAL_PURCHASE_PRICE`, you could write an application process of type **On Load: Before Header** to compute these values. Then, the Application Express engine would execute the process on each page as it renders the application. As a result, each page, would display the most current values for `TOTAL_CART_ITEMS` and `TOTAL_PURCHASE_PRICE`.

Creating an Application Process

To create an application process:

1. Navigate to the Shared Components page:
    a. On the Database Home Page, click **Application Builder**.
    b. Select an application.
    c. On the Application home page, click **Shared Components**.

       The Shared Components page appears.
2. Under Logic, select **Application Processes**.
3. Click **Create**.
4. For Identification:
    a. Name - Enter a name for the application process.
    b. Sequence Number - Specify the sequence number for this process. The sequence number determines the order in which the process will be evaluated relative to other process.
    c. Point - Identify the point at which this process executes.
    d. Click **Next**.
5. For Source:
    a. Process Text - Enter the text that is to be the source of your process.
    b. Error Message - Enter the error message that displays if the process raises an error.
    c. Click **Next**.
6. For Conditionality:
    a. Condition Type - Select a condition type that must be met in order for this process to execute.
    b. Expression 1 and Expression 2 - Uses these attribute to conditionally control whether or not the process executes. Enter values in this attribute based on the specific condition type you select. The process will execute if the specified condition is met.
    c. Click **Create Process**.

About the Application Process Page

Once you create an application process, it appears on the Application Processes page. You control how the page displays by making a selection from the View list. Available options include:

- **Icons** (the default) displays each process as a large icon. To edit a process, click the appropriate icon.
- **Details** displays each application process as a line in a report. To edit a process, click the name.

Accessing Application Processes Reports

After you create an application process, you can access the Utilization and History reports.

To access application processes reports:

1. Navigate to the Database Home Page.

2. Click **Application Builder**.
3. Select an application.
4. When Application Builder appears, click **Shared Components**.
5. Under Logic, select **Application Processes**.
6. Select one of the following tabs at the top of the page:
   - **Utilization**
   - **History**
7. Follow the on-screen instructions.

Utilization

Click **Utilization** to display the Application Process Utilization page. This page displays application processes used in the current application.

History

Click **History** to display the Application Process History page. This page displays a history of recently changed application process by date.

Understanding Application Computations

Application Computations are units of logic that set the value of a single page or application level item and are run at the same point across multiple pages in an application. Like page level computation, application computations can be based on static values, item values, PL/SQL, or SQL.

Topics in this section include:

- About Application Computations
- Creating an Application Computation
- Accessing the Application Computation History Report

About Application Computations

A common use of an application item is to store the value of the last page viewed in the application. By storing the value in an item, you can add a back button and then redirect the user to the page ID captured by the computation. This type of computation works well, for example, when you need to enable users to back out of an error page.

The following is an example of a computation that stores the last page visited. In this example, the computation:

- Stores the last application page visited to an item named `LAST_PAGE`
- Checks that the value of a `CURRENT_PAGE_ITEM` is of type PL/SQL Function Body with a Computation body of:

```
BEGIN
   :LAST_PAGE := nvl(:CURRENT_PAGE,:APP_PAGE_ID);
   :CURRENT_PAGE := :APP_PAGE_ID;
   RETURN :LAST_PAGE;
END;
```

About Application Computations that Execute On New Instance

Typically an application computation runs at the same point across multiple pages in an application. The exception is computations having a Computation Point of **On New Instance**. This type of computation only runs when a user first accesses your application. This type of computation is useful when you need to only retrieve information once within a user's session (for example, to retrieve a user’s job title).

Creating an Application Computation

To create an application computation:

1. Navigate to the Shared Components page:
   a. On the Database Home Page, click **Application Builder**.
   b. Select an application.
   c. On the Application home page, click **Shared Components**.

      The Shared Components page appears.
2. Under Logic, select **Application Computation**.
3. Click **Create**.
4. For Computation Item, select the item this computation will affect.
5. For Computation Point, select a process point at which this computation should be performed.
6. For Computation:
   a. For Computation - Enter the computation logic that corresponds to the computation type.
   b. Computation Error Message - Enter the error message that displays if the computation fails.
7. From Authorization Scheme (optional), select an authorization scheme which must evaluate to True in order for this computation to execute.
8. Under Conditions:
   a. Condition Type - Select a condition type that must be met in order for this computation to execute.
   b. Expression 1 and Expression 2 - Uses these attribute to conditionally control whether or not the computation executes. Enter values in this attribute based on the specific condition type you select. The computation will execute if the specified condition is met.
9. From Build Option (optional), select a build option for this component.
10. Click **Create**.

About the Application Computations Page

Once you create an application computation, it appears on the Application Computations page. You control how the page displays by making a selection from the View list. Available options include:

- **Icons** (the default) displays each computation as a large icon. To edit an computation, click the appropriate icon.

- **Details** displays each application process as a line in a report. To edit a computation process, click the name.

Accessing the Application Computation History Report

Once you create an application computation, you can view the Application Computation History report.

To access the Application Computation History report:

1. Navigate to the Database Home Page.
2. Click the **Application Builder** icon.
3. Select an application.
4. When Application Builder appears, click **Shared Components**.
5. Under Logic, select **Application Processes**.
6. Select **History** tab at the top of the page.

   This Application Computation History report displays a history of recently changed application computations by date.

Viewing Application Reports

Application Builder includes over 65 reports which provide a comprehensive view of your application from various perspectives. You can use application reports to achieve consistency among shared components and page components within your application. For example, you can view details about buttons used on all pages within your application. Additionally, many reports are updatable so you can standardize components, such as item and region labels, without navigating to a specific page.

To view reports specific to the currently selected application:

1. Navigate to the Database Home Page.
2. Click the **Application Builder** icon.
3. Select an application.

   The Applications home page appears.
4. On the Tasks list, click **View Application Reports**.
5. Select a type of report to view:
   - **Shared Components** reports offer information on common elements that can display on every page within an application. Report examples include Application Comments, Breadcrumb Entries, Database Object Dependencies, Lists of Values, Static Entries, and Messages).
   - **Page Components** reports offer detailed information on controls and logic that execute when the page is rendered (for example, branches, buttons, computations, items, and regions).
   - **Activity Reports** offer details about developer activity within the current application. Available reports include Changes by Developer, Changes by Developer by Day, Chart of Changes by Developer, and Recent Changes.
   - **Cross Application Reports** offer information that apply to multiple applications. Available reports include Application Attributes, Application

Comments, Build Options, Page Component Counts, Security Profiles by Application, and Template Defaults by Application.

See Also: "Creating Custom Activity Reports Using HTMLDB_ ACTIVITY_LOG" on page 13-10

About the Database Object Dependencies Report

The Database Object Dependencies report identifies database objects referenced by the current application. Review this report to determine what objects to move when deploying an application.

See Also: "How to Deploy an Application to Another Oracle Application Express Instance" on page 11-4

To view the Database Object Dependencies report:

1. Navigate to the Database Home Page.
2. Click the **Application Builder** icon.
3. Select an application.
4. On the Tasks list, click **View Application Reports**.
5. Click **Shared Components**.
6. Under Shared Component Reports, select **Database Object Dependencies**.
7. Click **Compute Dependencies**.
8. To view the components that reference a specific database object, select the Reference Count number.

About the Region Search Source Report

Use the Region Source report to search through all region source in your application.

To view the Region Source report:

1. Navigate to the Database Home Page.
2. Click the **Application Builder** icon.
3. Select an application.
4. On the Tasks list, click **View Application Reports**.
5. Click **Page Components**.
6. Select **Search Source**.
7. To view the Page Definition for a region, select the page ID.
8. To view region attributes for a region, select the region name.

5 Building an Application

This section describes how to use Application Builder to build an application and application components. It includes instructions for creating an application and adding pages as well as adding components (reports, charts, or forms), page controls (buttons, items, list of values), and shared components (menus, lists, or tabs).

This section contains the following topics:

See Also:

- "Using Application Builder" on page 4-1
- "Controlling Page Layout and User Interface" on page 7-1
- "Adding Navigation" on page 6-1

Creating an Application

An application is a collection of pages which share a common session state and authentication. You create a new application in Application Builder using a wizard. You delete an application from the Application Builder home page.

Topics in this section include:

- About the Create Application from Spreadsheet Wizard
- About Demonstration Applications
- Deleting an Application

About Creating an Application Using a Wizard

When you click **Create** on the Application Builder home page, you must choose one of the following options:

- **Create Application**. Creates an application based on SQL queries or database tables. You can define blank pages or pages that contain reports, forms, tabular forms, or a report with a linked form. See "About the Create Application Wizard" on page 5-2.
- **Create Application from Spreadsheet**. Creates an application based on spreadsheet data. You can upload or paste spreadsheet data to create a table and then add a user interface. In the resulting application, users can create queries, add, insert, or update records, or analyze the data. See "About the Create Application from Spreadsheet Wizard" on page 5-5.
- **Demonstration Application**. Installs or uninstalls demonstration applications. Use demonstration applications to learn how to build applications. See "About Demonstration Applications" on page 5-6.

See Also: "Adding Pages to an Application" on page 5-8 for information about adding reports and forms by creating a new page

About the Create Application Wizard

The Create Application wizard enables you to create a fully functional application based on any number of tables. You can use the Create Application Wizard to create blank pages, or pages based on SQL queries or database tables. You can create SQL queries by manually typing SQL or by using the graphical user interface of Query Builder. Applications based on tables can consist of a simple report, a form and report, or a tabular form. When creating pages on tables, you have the option to generate analysis pages. Analysis pages extend a simple report or a report on a form to include multiple drill-down reports and charts.

Topics in this section include:

- Creating an Application Based on Tables or Queries
- About Application Models and User Interface Defaults
- Leveraging Application Models and User Interface Defaults

Creating an Application Based on Tables or Queries

You can create an application based on a table, query, or drill-down query by selecting **Create Application** in the Create Application Wizard.

To create an application based on a table, query, or drill-down query:

1. On the Database Home Page, click the **Application Builder** icon.
2. Click the **Create** button.
3. Select **Create Application** and click **Next**.
4. Enter the basic application details and click **Next**:

   a. Name - Enter a name to identify the application.
   b. Application - Enter an unique integer value to identify the application.
   c. Create Application - Select a creation method:
      – **From scratch** enables you to add pages manually
      – **Based on existing application design model** enables you to copy page definitions from a previous application model.

        Note that you will still have to define all other application attributes, or you can choose to copy some attributes using by choosing to copy shared components from another application (See step 7 and "About Application Models and User Interface Defaults" on page 5-4).
   d. **Schema** - Your application will obtain its privileges by parsing all SQL as a specific database schema. Identify the database schema owner.

   Next, add pages to your application.

5. Under Add Pages:
   a. Select the type of page you want to add. Options include:
      – **Blank** creates a page with no built-in functionality.
      – **Report** creates a page the contains the formatted result of a SQL query. You can choose to build a report based on a table you select, or based on a custom SQL SELECT statement or a PL/SQL function returning a SQL SELECT statement that you provide.
      – **Form** creates a form to update a single row in a database table.
      – **Tabular Form** creates a form to perform update, insert, and delete operations on multiple rows in a database table.
      – **Report and Form** builds a two page report and form combination. On the first page, users select a row to update. On the second page users can add a new record or update or delete an existing record.

      Action displays the currently selected page type. For each selection, the wizard prompts you for a variety of different types of information.

      Report pages include the **Include Analysis Pages** check box. Select this option and follow the wizard prompts to extend a simple report or a report on a form to include multiple drill-down reports and charts.
   b. Click **Add Page**.

      The page (or pages) appear at the top of the page. To delete a page, click **Delete** icon.
   c. Repeat the previous steps until all pages have been added.
   d. Click **Next**.
6. Determine whether to include tabs in your application and click **Next**.
7. Determine whether to import shared components from another application. Shared components are common elements that can display or be applied on any page within an application.

   To include shared components:
   a. From Copy Shared Components from Another Application, select **Yes**.

   b. From Copy from Application, select the application from which you want to import shared components.
   c. From Select Components to Import, select the components to import.
   d. Click **Next**.
8. Select the following authentication and globalization preferences:
   a. Default Authentication Scheme - Identify an authentication scheme you would like to use by default.

      Authentication is the process of establishing users' identities before they can access an application. See "Establishing User Identity Through Authentication" on page 12-14.
   b. Language - Select the primary language for this application.

      This attribute identifies the language in which an application is developed. This language is the base language from which all translations are made.
   c. User Language Preference Derived From - Specifies how the engine determines the application language. The application primary language can be static (that is, derived from the Web browser language) or determined from a user preference or item. The database language setting determines date display and sorting characteristics.

      You can alter the Language and User Language Preference Derived From attributes later on Edit Globablization attributes page. See "Configuring Globalization Attributes" on page 4-14.
   d. Click **Next**.
9. Select a theme and click **Next**.

   Themes are collections of templates that can be used to define the layout and style of an entire application. See "Managing Themes" on page 7-8.
10. Confirm your selections. To return to a previous wizard page, click **Previous**. To accept your selections, click **Finish**.

About Application Models and User Interface Defaults

The Create Application Wizard is designed with the assumption that the developer may run it multiple times. To facilitate this iterative approach to application development, every time you run the wizard it saves the page definitions to an application model.

Consider the following example. You create a new application by running the Create Application Wizard. After viewing the application, you realize it is not quite what you wanted. Instead of altering it, you can run the wizard again and select an application model. By selecting an existing application model when you rerun the wizard, you can quickly improve your application with minimal time and effort.

See Also: "Managing Application Models" on page 8-10

Another way to increase your productivity when creating an application is to specify user interface defaults. User interface defaults are metadata that enable you to assign default user interface properties to a table, column, or view within a specified schema.

See Also: "Managing User Interface Defaults" on page 9-1

Leveraging Application Models and User Interface Defaults

You can increase your productivity when creating applications by leveraging application models and user interface defaults. Consider the following scenario:

1. Create an application based on tables or views by running the Create Application Wizard.
2. Run the generated application. Note any functional deficiencies.
3. Evaluate whether to create or edit user interface defaults.

   For example, you can use user interface defaults to control how form field or report labels display. You can also utilize user interface defaults to display specific columns or have columns display in an alternate order.
4. Navigate to the Application home page and create a new application by clicking **Create**.
5. Select **Create Application**.
6. When prompted to enter application details, specify the following:
   a. Name - Enter a name to identify the application.
   b. Application - Enter an unique integer value to identify the application, or accept the default.
   c. Create Application - Select **Based on existing application design model**.
7. Select an application model.

   Note the pages you previously created already appear.
8. Add pages, edit pages, or remove pages.
9. Complete the wizard.
10. Repeat steps 2 through 9 until the application meets your functional requirements.

About the Create Application from Spreadsheet Wizard

You can create an application based on spreadsheet data by selecting **Create Application from Spreadsheet** in the Create Application Wizard.

To create an application from spreadsheet data:

1. On the Database Home Page, click the **Application Builder** icon.
2. Click the **Create** button.
3. Select **Create Application from Spreadsheet**.
4. Specify how spreadsheet data will be uploaded. Select one of the following and click **Next**:
   a. **Upload file** (comma-delimited or tab-delimited). Specify the following and click **Next**:
      – File - Click Browse to locate the file to be uploaded.
      – Separator - Specify the column separator character. Use \t for tab separators.
      – Optionally Enclosed By - Enter a delimiter character. You can use this character to delaminate the starting and ending boundary of a data value. If you specify a delimiter character, the wizard ignores white space occurring before the starting and ending boundary of a data value. You

can also use this option to enclose a data value with the specified delimiter character.

- File Character Set - Choose the character set in which the text file is encoded.

b. **Copy and paste** (up to 30KB). Copy and paste the spreadsheet data you wish to import and click **Next**.

5. Review the preview of how your table will display and click **Next**. Specify the table name and column names, modify the data types, or specify which columns to include.
6. Review the displayed Singular Name and enter a Plural Name.

   Column User Interface Defaults display default label names.
7. (Optional) Under Column User Interface Defaults, edit the displayed Label names and click **Next**.
8. For Summary By Column, select the columns for which data will be summarized in reports and charts and click **Next**.
9. This wizard creates several summary reports. Select columns for which values will be aggregated in summary reports.

   a. **Aggregate by Column** - Choose one or more columns for which you want data summarized or averaged.

   b. **Aggregate Function to Use** - Select the aggregate function to use in the report (Sum or Average).

   c. Click **Next**.
10. Select Application Options:

    a. **Application Name** - Enter an alphanumeric name for this application.

    b. Specify a Create Mode:

    - **Read and Write** includes insert and update pages.
    - **Read Only** does not include insert and update pages.

    c. Select a chart type.

    d. Click **Next**.
11. Select a theme and click **Next**.

    Themes are collections of templates that can be used to define the layout and style of an entire application. See "Managing Themes" on page 7-8.
12. Confirm your selections. To return to a previous wizard page, click **Previous**. To accept your selections, click **Create**.

About Demonstration Applications

Oracle Application Express installs with a number of demonstration applications. Use these applications to learn more about the different types of functionality you can include in your applications.

See Also: "Running a Demonstration Application" on page 2-1

Accessing Demonstration Application

To access demonstration applications:

1. On the Database Home Page, click the **Application Builder** icon.
2. Click the **Create** button.
3. Select **Demonstration Application**.

   The Demonstration Applications page appears, displaying links to the following applications:

   - *Sample Application* offers a working demonstration that highlights basic design concepts
   - *Collection Showcase* demonstrates shopping cart concepts
   - *Web Services* serves an example of how you can use Web Services
4. To install a demonstration application, scroll down to the application you want to install, click **Install**.

   Installed appears as the Status.
5. To edit an installed demonstration application, click **Edit**.
6. To run an installed demonstration application, click **Run**.
7. To reinstall a demonstration application, click **Re-Install**.

Deleting an Application

You can delete an application from within Application Builder, or while editing application attributes. If you delete an application you also delete all defined components (reports, charts, or forms), page controls (buttons, items, list of values), and shared components (breadcrumbs, lists, and tabs, but not user interface defaults).

Topics in this section include:

- Deleting an Application from Application Builder
- Deleting an Application from Edit Application Attributes

Deleting an Application from Application Builder

To delete an application from Application Builder:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. When Application Builder appears, verify the application ID and name at the top of the page.
4. On the Tasks list, click **Delete this Application**.
5. Follow the on-screen instructions.

Deleting an Application from Edit Application Attributes

To delete an application from Edit Application Attributes:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click Edit Attributes.

4. Click **Edit Standard Attributes**.
5. Verify the application ID and name.
6. Click **Delete** at the top of the page.

See Also: "Configuring Standard Application Attributes" on page 4-6

Adding Pages to an Application

You can add a new page or add a component to an existing page by running the Create Page Wizard. You can access this wizard by:

- Clicking **Create Page** on the Applications home page
- Clicking **Create** on the Page Definition
- Selecting the Create link on the Developer toolbar

Note: You can also add a component (that is, a report, chart, form, wizard, a calendar, or tree) to an existing page using the Create Page Wizard. When prompted, specify an existing page ID.

Topics in this section include:

- Creating a Page from Application Home Page
- Creating a Page from the Page Definition
- Creating a Page from the Developer Toolbar
- Running a Page or Application
- Grouping Pages
- Locking and Unlocking a Page
- Deleting a Page

See Also: "Creating Reports" on page 5-17, "Creating Charts" on page 5-39, "Creating Forms" on page 5-26, "Creating Calendars" on page 5-34, and "Creating Trees" on page 5-34

Creating a Page from Application Home Page

To create a new page from the Application home page:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.

   The Application home page appears.
3. Click **Create Page**.
4. Select the type of page you want to create:
   - Blank Page
   - Multiple Blank Pages
   - Report
   - Chart

- Form
- Wizard
- Calendar
- Tree
- Login Page

5. Follow the on-screen instructions.

Creating a Page from the Page Definition

To create a new page while viewing a Page Definition:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. From the navigation bar at the top of the page, click the **Create** button.
3. Select the type of page you want to create:
   - New Page
   - Region on this page
   - Page control on this page
   - Shared Control

   Table 5–1 describes the various selections available based on the type of page you select.

Table 5–1 Create Page Options

Create Page Options	Available Selections
New Page	Available page types: ■ Blank age ■ Multiple blank pages ■ Report ■ Chart ■ Form ■ Wizard ■ Calendar ■ Tree ■ Login Page

Table 5–1 (Cont.) Create Page Options

Create Page Options	Available Selections
Region on this page	Regions function as containers for content. Available region types: ■ HTML ■ Report ■ Form ■ Chart ■ Breadcrumb ■ PL/SQL Dynamic Content ■ Tree ■ URL ■ Calendar ■ Multiple HTML ■ Help Text
Page control on this page	Page controls: ■ Item ■ Button ■ Branch ■ Computation ■ Process ■ Validation
Shared control	Shared component options: ■ Navigation Bar icon ■ Parent tab ■ Standard tab ■ List of values ■ List ■ Breadcrumb

4. Follow the on-screen instructions.

See Also: Editing a Page Definition on page 4-19

Creating a Page from the Developer Toolbar

Users who log in to the Database Home Page user interface using a database user account have access to the Developer toolbar. The Developer toolbar displays at the bottom every page and offers a quick way create a new page.

To create a new page from the Developer toolbar:

1. On the Developer toolbar, select **Create**.

   The New Component Wizard appears.

2. Select the type of component you want to create and click Next. Available options include:

   - New Page

- Region on this page
- Page control on this page
- Shared control

  Table 5–1 on page 5-9 describes the various selections available based on the type of page you select.

3. Follow the on-screen instructions.

   See Also: "Using the Developer Toolbar" on page 4-18

Running a Page or Application

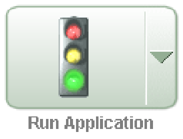
The Application Express engine dynamically renders and processes pages based on data stored in database tables. To view a rendered version of your application, you run or submit it to the Application Express engine. As you create new pages, you can run them individually, or run an entire application. You can run an application by clicking the Run Application icon.

Topics in this section include:

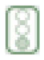
- About the Run Application Icon
- Running an Application from the Application Builder Home Page
- Running an Application from the Application Home Page
- Running a Page from the Pages List on the Application Home Page
- Running a Page from the Page Definition

About the Run Application Icon

The Run Application icon resembles a traffic light. A large colored Run Application icon appears on the Application home page.



Many pages within Application Builder also feature a smaller, light green version of this icon. Clicking this smaller Run icon runs the current application or individual pages depending upon the context. For example, clicking the Run icon on the Application home page runs the entire application. Clicking the icon on the Page Definition runs the current page.



Running an Application from the Application Builder Home Page

To run an entire application from the Application Builder home page:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the View list, select **Details** and click **Go**.
3. Locate the application in the Applications list.
4. Click the **Run** icon in the far right column.

Running an Application from the Application Home Page

To run an entire application from the Application home page:

1. On the Database Home Page, click the **Application Builder** icon.

   The Application Builder home page appears.

2. Select on application.
3. Click the **Run** icon.

Running a Page from the Pages List on the Application Home Page

To run a page from the Pages list:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. From the View list, select **Details** and click **Go**.
4. From the Pages list, locate the page you want to run and click the **Run** icon in the far right column.

Running a Page from the Page Definition

To run a specific page from the Page Definition:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Select a page.

   The Page Definition appears.

4. Click the **Run Page** icon in the upper right corner of the page.

Grouping Pages

Use page groups to organize and manage the pages within an application. To use page groups, you create a group and then assign pages to the group.

Page groups do not have any function other than to help a developer organize their application pages.

You can make the pages within your application easier to access by organizing them into page groups.

Topics in this section include:

- Viewing the Page Group Report
- Creating a Page Group
- Assigning Pages to a Page Group

Viewing the Page Group Report

The Page Group report offers a comprehensive list of which pages in an application are assigned to a group and which pages are unassigned.

Viewing Page Groups from the Application Home Page To view the Page Group report from the Application home page:

1. On the Database Home Page, click the **Application Builder** icon.

2. Select the application.
3. On the Tasks List on the right side of the page, click **Manage Page Groups**.
4. On the Tasks list, click **Report Page Groups**.

Viewing Page Groups from the Page Definition To view page groups from the Page Definition:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Select a page.

   The Page Definition appears.
4. From the View list, select **Groups**.

Creating a Page Group

To create a page group:

1. On the Database Home Page, click the **Application Builder** icon
2. Select an application.
3. On the Tasks List on the right side of the page, click **Manage Page Groups**.
4. On the Page Groups page, click **Create**.
5. Enter a name, a description (optional), and click **Create**.

Assigning Pages to a Page Group

To assign pages to page group:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Tasks List on the right side of the page, click **Manage Page Groups**.
4. On the Tasks list, click **Report Unassigned Pages**.

   The Unassigned Pages page appears.
5. From Page Group, select a group to which you want to assign pages.
6. Select the pages to be assigned.
7. Click **Assigned Checked**.

   Selecting the page ID takes you to the Page Attributes page. Selecting the Page Name links to the Page Definition.

Locking and Unlocking a Page

You can prevent conflicts during application development by locking pages in your application. By locking a page, you prevent other developers from editing it.

Topics in this section include:

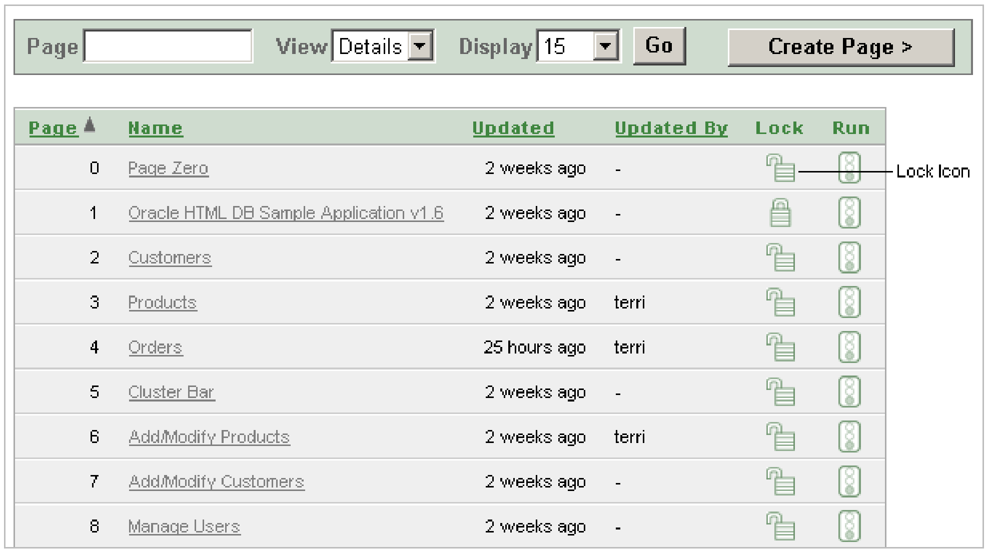
- Determining If a Page Is Locked
- Locking a Page
- Unlocking Pages

- Accessing Alternative Locked Pages Views

Determining If a Page Is Locked

A lock icon indicates whether a page is currently locked. If a page is unlocked, the icon appears as an open padlock. If the page is locked, the icon appears as a locked padlock. A lock icon appears on the following pages:

- **Application home page**. Select **Details** from the Display list. A list of pages appears. The lock icon appears under the Lock column.
- **Page Definition**. The lock icon appears on the far right side of the page across from the breadcrumb menu.



Page	Name	Updated	Updated By	Lock	Run
0	Page Zero	2 weeks ago	-		
1	Oracle HTML DB Sample Application v1.6	2 weeks ago	-		
2	Customers	2 weeks ago	-		
3	Products	2 weeks ago	terri		
4	Orders	25 hours ago	terri		
5	Cluster Bar	2 weeks ago	-		
6	Add/Modify Products	2 weeks ago	terri		
7	Add/Modify Customers	2 weeks ago	-		
8	Manage Users	2 weeks ago	-		

Locking a Page

You can lock pages from the Page Locks page, the Pages list, and from a Page Definition.

Locking a Page from the Page Locks Page To lock a page of your application:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Tasks List on the right side of the page, click **Manage Page Locks**.
4. Select the appropriate pages and click **Lock Checked**.
5. Enter a comment in the Comment field.
6. Click **Lock Page(s).**

Locking a Page from the Pages List To lock a page from the Pages list:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. From the View list, select **Details** and click **Go**.
4. In the Pages list, locate the page you want to lock and click the **Lock** icon.
5. Enter a comment in the Comment field.

6. Click **Lock Page(s)**.

Locking a Page from the Page Definition To lock a page from the Page Definition:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Click the **Lock** icon in the upper right corner above Shared Components.
3. Select the appropriate pages and click **Lock Checked**.
4. Enter a comment in the Comment field.
5. Click **Lock Page(s).**

Unlocking Pages

Only the developer who locked a page can unlock it. However, a developer with administrative privileges can unlock pages locked by other developers.

Unlocking Pages from the Page Locks Page To unlock a page from the Page Locks page:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select the application.
3. On the Tasks List on the right side of the page, click **Manage Page Locks**.
4. Select the appropriate pages.
5. Click **UnLock Checked**.

Unlocking Pages from the from the Pages List To unlock a page from the Pages list:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. From the View list, select **Details** and click **Go**.
4. In the Pages list, locate the page you want to unlock and click the **Lock** icon.

   The Edit Lock Comment page appears.
5. Click **UnLock**.

Unlocking Pages from the Page Definition To unlock pages from the Page Definition:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select the application.
3. Select the Page you want to unlock in the Pages list.
4. Click the **Lock** icon in the upper right corner above Shared Components.

   The Page Locks page appears.
5. Select the Page you want to unlock and click **Unlock Checked**.

Accessing Alternative Locked Pages Views

You can access a number of different views of Locked Pages on the Locked Pages page.

To access different views of locked pages:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select the application.
3. On the Tasks List on the right side of the page, click **Manage Page Locks**.
4. On the Tasks list, click one of the following:
   - **Show Locked Pages** displays only locked pages within the current application.
   - **Show All Pages** displays all pages within the current application.
   - **Show Unlocked Pages** display only unlocked pages within the current application.
   - **Administer Locks** enables an administrators to unlock any pages locked by a developer.

Deleting a Page

You can delete a page from the Page Definition or while editing page attributes.

Topics in this section include:

- Deleting a Page from the Page Definition
- Deleting a Page While Editing Page Attributes

Deleting a Page from the Page Definition

To delete a page from the Page Definition:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Select a page.

   The Page Definition appears.
4. Verify the page name.
5. From the navigation bar at the top of the page, click **Delete**.
6. Follow the on-screen instructions.

   See Also: "Editing a Page Definition" on page 4-19 for information about editing page attributes

Deleting a Page While Editing Page Attributes

To delete a page while editing page attributes:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Select a page.

   The Page Definition appears.
4. Click the **Edit Attributes** icon.

5. Click **Edit Standard Attributes**.
6. Verify the application ID and page name.

   The Edit Application Attributes page appears.
7. Click **Delete**.
8. Follow the on-screen instructions.

Creating Reports

In Oracle Application Express, a report is the formatted result of a SQL query. You can generate reports by selecting and running a built-in query, or by defining a report region based on a SQL query.

Topics in this section include:

- Creating a Report Using a Wizard
- Editing Report Attributes
- Controlling Report Pagination
- Enabling Column Sorting
- Adding a CSV Link to a Report
- Exporting a Report as an XML File or a CSV File
- Creating a Column Link
- Defining an Updatable Column
- Defining a Column as a List of Values
- Controlling When Columns Display
- Controlling Column Breaks

Creating a Report Using a Wizard

Application Builder includes a number of built-in wizards for generating reports.

To create a report using a wizard:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select the application.
3. Click Create Page.
4. Select **Report**.
5. Select one of the following report types:
   - **Wizard Report** - Does not require any SQL knowledge. Select the appropriate schema, table, columns, and result set display.
   - **SQL Report** - Creates a report based on a custom SQL SELECT statement or a PL/SQL function returning a SQL SELECT statement that you provide.
6. Follow the on-screen instructions.

Editing Report Attributes

You can use the Report Attributes and Column Attributes pages to precisely control the definition of report pages. For example, you can use these attributes to alter column heading text, change column positioning, hide a column, create a sum of a column, or select a sort sequence.

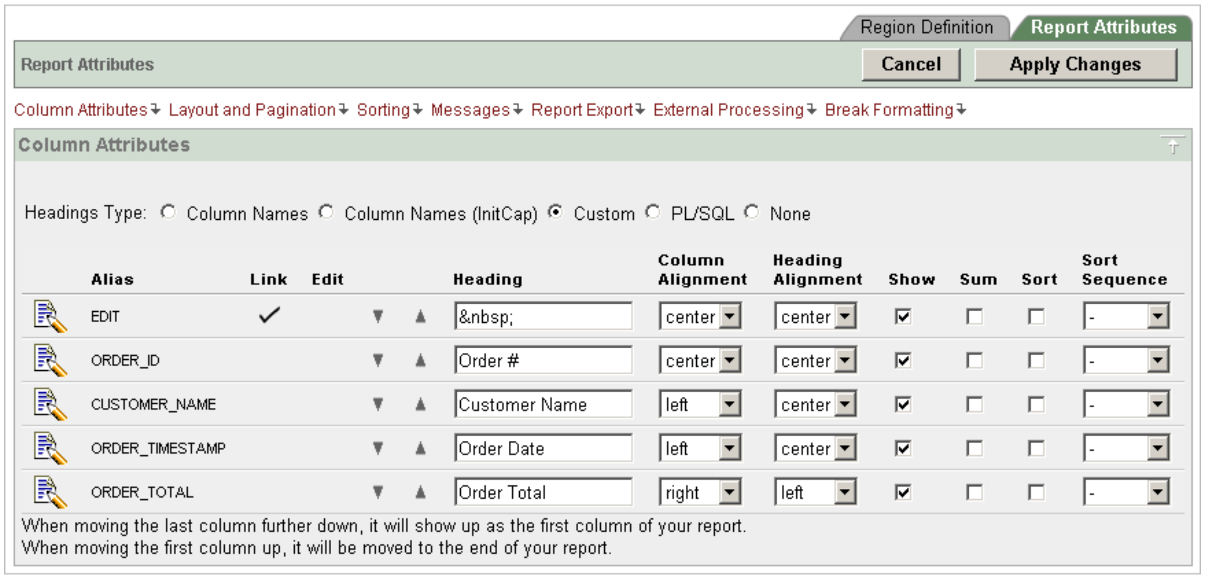
On the Page Definition, you can access the Report Attributes page by clicking either **Report** or **RPT,** adjacent to the report region you want to edit. **Report** indicates the report is a regular report, and **RPT** indicates the report is an wizard report. You can also navigate to the Report Attributes page by clicking the region name and then selecting the Report Attributes tab.

To access the Report Attributes page:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select the application.
3. Select a page.

   The Page Definition appears.
4. Under Regions, click **Report** next to the name of the report region you want to edit.

   The Report Attributes page appears.



Headings Type: Column Names Column Names (InitCap) [✓ Custom] PL/SQL None

Alias	Link	Edit	Heading	Column Alignment	Heading Alignment	Show	Sum	Sort	Sort Sequence
EDIT	✓			center	center	✓			-
ORDER_ID			Order #	center	center	✓			-
CUSTOMER_NAME			Customer Name	left	center	✓			-
ORDER_TIMESTAMP			Order Date	left	center	✓			-
ORDER_TOTAL			Order Total	right	left	✓			-

When moving the last column further down, it will show up as the first column of your report.
When moving the first column up, it will be moved to the end of your report.

Heading Type identifies how the heading was generated for the report. Use the Column Attributes section to control report column appearance and functionality. The Link column indicates if a column link is currently defined. The Edit column indicates whether or not a column is currently updatable.

Table 5–2 describes common report column edits.

Table 5–2 Common Report Column Edits

Description	Developer Action
Alter column display sequence.	Click the up and down arrows.
Alter heading alignment.	Under Column Alignment, select a new column alignment.
Change column heading text.	Under Heading, enter different heading text.

Table 5–2 (Cont.) Common Report Column Edits

Description	Developer Action
Control which columns display.	Click **Show** to indicate a column should display.
Enable an unique sort sequence.	Click **Sort** and select a sequence number from **Sort Sequence**. Any number of columns can be sort enabled. However, at least one column must have a Sort Sequence defined.
Enable the sum of a column.	Click **Sum** to enable the sum of a column.

You can further refine the attributes of a specific column on the Column Attributes page.

5. To access the Column Attributes page, click the **Edit** icon adjacent to the appropriate column Alias.

   See online Help for more information about a specific attribute.

Controlling Report Pagination

You control report pagination by:

- Including a pagination substitution string in the report template
- Making selections from Layout and Pagination on the Report Attributes page

You control how pagination displays by making selections from the Layout and Pagination attributes on the Report Attributes page.

To access the Layout and Pagination section of the Report Attributes page:

1. Create a report. See "Creating a Report Using a Wizard" on page 5-17.
2. Under Regions, click the appropriate report attributes link (**Report** or **RPT**).

   The Report Attributes page appears.
3. Scroll down to Layout and Pagination.

   You use the Layout and Pagination attributes to select a pagination style, determine where pagination displays, and specify the number of rows that display on each page. Table 5–3 describes the most commonly used Layout and Pagination attributes.

Table 5–3 Layout and Pagination Attributes

Attribute	Description
Report Template	Specifies a template to be applied to this report. Report templates provide control over the results of a row from your SQL query. You can choose from a number of default templates, or pick a custom build template.
Pagination Scheme	Specifies a pagination scheme for this report. Pagination provides the user with information about the number of rows and the current position within the result set. Pagination also defines the style of links or buttons used to navigate to the next or previous page. For more information, see the Help for this item.

Table 5–3 (Cont.) Layout and Pagination Attributes

Attribute	Description
Display Position	Defines where pagination displays. If you choose to display pagination above a report, the selected report template needs to support that type of display.
Number of Rows	Defines the maximum number of rows to display on each page.
Strip HTML	Specifies whether or not to remove HTML tags from the original column values for HTML expressions and column links. If you select values from the database that already contain HTML tags, then those tags can cause conflicts with the HTML generated for your columns links or HTML expressions. When this option is enabled, only the actual data portion of your column value is used.

Including Pagination After the Rows in a Report

To include pagination after the rows in a report:

1. Create a report. See "Creating a Report Using a Wizard" on page 5-17.

   Next, select the appropriate Layout and Pagination attributes.

2. Navigate to the Report Attributes page:
   a. Navigate to the Page Definition.
   b. Under Regions, click the appropriate report attributes link (**Report** or **RPT**).

      The Report Attributes page appears.

3. Under Layout and Pagination, select the following:
   a. Report Template - Select a report template (optional).
   b. Pagination Scheme - Select a pagination scheme.
   c. Display Position - Select a display position.
   d. Number of Rows - Specify how many rows display on each page.
   e. Click **Apply Changes**.

4. Edit the report template:
   a. Navigate to the Page Definition.
   b. Under Templates, select the report template name.
   c. Include the `#PAGINATION#` substitution string in the After Rows attribute.
   d. Click **Apply Changes**.

5. Run the page.

Including Pagination Before the Rows in a Report

To include pagination before the rows in a report:

1. Create a report. See "Creating a Report Using a Wizard" on page 5-17.

   Next, select the appropriate Layout and Pagination attributes.

2. Navigate to the Report Attributes page:
   a. Navigate to the Page Definition.

   b. Under Regions, click the appropriate report attributes link (**Report** or **RPT**).

      The Report Attributes page appears.

3. Under Layout and Pagination:
   a. Report Template - Select a report template (optional).
   b. Pagination Scheme - Select a pagination scheme.
   c. Display Position - Select a position that contains the word top.
   d. Number of Rows - Specify how many rows display on each page.
   e. Click **Apply Changes**.
4. Edit the report template.
   a. Navigate to the Page Definition.
   b. Under Templates, select the report template name.
   c. Include the `#TOP_PAGINATION#` substitution string in the Before Rows attribute.
   d. Click **Apply Changes**.
5. Run the page.

Enabling Column Sorting

You enable column sorting on the Report Attributes page.

To enable column sorting:

1. Navigate to the Report Attributes page. See "Editing Report Attributes" on page 5-18.
2. Under Report Column Attributes, select the **Sort** check box adjacent to the columns to be sorted.
3. From Sort Sequence, select a sequence number.

   Sort Sequence is optional. However, if there are one or more sort enabled columns, then at least one column needs a defined Sort Sequence.
4. Scroll down to Sorting.
5. Specify ascending and descending image attributes or click **set defaults**.

Adding a CSV Link to a Report

You can create a link within a report that enables users to export the report as a comma-delimited file (.csv) file. To add a CSV link to a report you need to enable the CSV output option. When using the CSV output option, the report template is not important. You can include a CSV link with any report template that has the CSV export substitution string defined.

See Also: "Automatic CSV Encoding" on page 4-15

Enabling the CSV Output Option

To enable the Enable CSV output option:

1. Navigate to the appropriate Report Attributes page. See "Editing Report Attributes" on page 5-18.

2. Scroll down to Report Export.
3. From Enable CSV output, select **Yes**.
4. (Optional) In the Separator and Enclosed By fields, define the separator and delimiter.

   The default Enclosed By by characters are a double quotation marks (" "). The default delimiter is either a comma or a semicolon depending upon your current NLS settings.
5. In the Link Label field, enter link text. This text will display in your report and enable users to invoke a download.
6. (Optional) To specify a default export file name, enter a name in the Filename field.

   By default, the Application Express engine creates an export file name by taking the region name and adding the appropriate file name extension (`.csv` or `.xml`).
7. Click **Apply Changes**.

Exporting a Report as an XML File or a CSV File

You can export a report as an XML files by selecting a report template.

To export a report as a file:

1. Navigate to the appropriate Report Attributes page. See "Editing Report Attributes" on page 5-18.
2. Scroll down to Layout and Pagination.
3. From the Report Template list, select **export: XML** or **export: CSV**.

   Selecting **export: XML** prevents the Application Express engine from rendering the page and dumps the content to an XML file.
4. Click **Apply Changes**.

Creating a Column Link

Use the Column Link attributes to create a link from a report to another page in your application or to a URL.

To create a column link to another page:

1. Navigate to the appropriate Report Attributes page. See "Editing Report Attributes" on page 5-18.
2. Under Report Column Attributes, locate the column to contain the link.
3. Click the **Edit** icon adjacent to the column name.

   The Column Attributes page appears.
4. Scroll down to Column Link.
5. To create a column link to another page:
   a. From Target, select **Page in this Application**.
   b. (Optional) In Link Attributes, specify additional column link attributes that will be included in the `<a href= >` tag (for example, a link target, classes, or styles).

c. In Link Text, enter the text to be displayed as a link, specify an image tag, or pick from the list of default images.

d. In Page, specify the target page ID. To reset the pagination for this page, select **Reset Pagination**.

e. In Request, specify the request to be used.

f. In Clear Cache, specify the pages (that is, the page IDs) on which to clear cache. You can specify multiple pages by listing the page IDs in a comma-delimited list.

g. Use the Name and Value fields to specify session state for a specific item.

6. Click **Apply Changes**.

To create a column link to a URL:

1. Navigate to the appropriate Report Attributes page. See "Editing Report Attributes" on page 5-18.
2. Access the Column Attributes page by clicking the **Edit** icon adjacent to the appropriate column.

   The Column Attributes page appears.
3. Scroll down to Column Link.
4. Under Column Link, make the following selection:
   a. From Target Type, select **URL**.
   b. In Link Text, enter the text to be displayed as a link and select a substitution string.
   c. (Optional) In Link Attributes, specify additional column link attributes that will be included in the `<a href= >` tag (for example, a link target, classes, or styles).
   d. In URL, enter the appropriate address.
5. Click **Apply Changes**.

Defining an Updatable Column

You can make a column updatable by editing Tabular Form Element attributes on the Column Attributes page. Note that the Application Express engine can only perform updates if:

- A multirow update is defined
- A PL/SQL process is implemented to process updated data
- When using the built-in tabular form elements and display types, then the report has to be defined using the type **SQL Query (updatable report)**

To define updatable column attributes:

1. Navigate to the appropriate Report Attributes page. See "Editing Report Attributes" on page 5-18.
2. Access the Column Attributes page by clicking the **Edit** icon adjacent to the appropriate column.

   The Column Attributes page appears.
3. Scroll down to Tabular Form Element.

4. Under Tabular Form Element, make the following selections:
   a. Display As - Select a type of updatable column.

      Use this option to make a column updatable. Updates can only be performed if a multirow update is defined, or PL/SQL process is implemented to process updated data.
   b. Date Picker Format Mask - Make a selection if you selected the Display As type of **Date Picker**.
   c. Element Width - Specify the width of the form item.
   d. Number of Rows - Specify the height of a form item (applicable to text areas).
   e. Element Attributes - Define a style or standard form element attribute.
   f. Element Option Attributes - Specify form element attributes for items in a radio group or check box.
   g. Primary Key Source Type - Identify the default type.
   h. Primary Key Source - Identify the default source.

      If the current column is part of the primary key defined in an MRU process, only the primary key source type and source appear.

      Otherwise, Default and Default Type appear. Use Default and Default Type to establish a relationship between two master records in a master detail form, or to set the default values for new rows.
   i. Reference Table Owner - Identify the owner of the referenced table. Use this attribute to build User Interface Defaults for reports.
   j. Reference Table Name - Identify the table or view that contains the current report column.
   k. Reference Column Name - Identify the column name that this report column references.
5. Click **Apply Changes**.

Defining a Column as a List of Values

Report columns can be rendered as lists of values. For example, a column can be rendered using a select list or a popup list of values. Or, a column can also be rendered as read-only text based on a list of values.

This last approach is an effective strategy when creating display lookup values and is particularly useful in regular, nonupdatable reports. This approach enables you to display the value of a column without having to write a SQL JOIN statement.

To render a report column as a list of values:

1. Navigate to the appropriate Report Attributes page. See "Editing Report Attributes" on page 5-18.
2. Access the Column Attributes page by clicking the **Edit** icon adjacent to the appropriate column.

   The Column Attributes page appears.
3. Scroll down to List of Values.
4. From Named LOV, make a selection from the List of Values repository.
5. To include a null value in a list of values:

   a. In Display Null, select **Yes**.
   b. In Null Text, specify the value that displays.

   A column can also have a value that does not display in its list of values.

6. To define a value that does not display in the list of values:

   a. From Display Extra Value, select **Yes**.

      The extra value is used if the actual column value is not part of the LOV. In that situation, the actual value is shown. If you do not display extra values, you may end up with the wrong value and unintentionally update your data incorrectly.

   b. In Null Value, specify the value that displays.
   c. If you have not selected a Named LOV, enter the query used to display a select list in the LOV Query field.

7. If you have not selected a Named LOV, enter the query used to display a select list in LOV Query.
8. Click **Apply Changes**.

   See Also: "Creating Lists of Values" on page 5-60

Controlling When Columns Display

You can use the Authorization and Condition column attributes to control when a column displays.

Authorization enables you to control access to resources (such as a report column) based on predefined user privileges. For example, you could create an authorization scheme in which only managers can view a specific report column. Before you can select an authorization scheme, you must first create it.

A condition is a small unit of logic that enables you to control the display of a column based on a predefined condition type. The condition evaluates to true or false based on the values you enter in the Expressions fields.

To specify Authorization and Condition attributes:

1. Navigate to the appropriate Report Attributes page. See "Editing Report Attributes" on page 5-18.
2. Access the Column Attributes page by clicking the **Edit** icon adjacent to the appropriate column.

   The Column Attributes page appears.

3. Under Authorization, make a selection from the Authorization Scheme list.
4. Under Conditions, make a selection from the Condition Type list, and depending upon your selection, enter an expression or value in the appropriate Expression fields.

   If the authorization is successful and the condition type display evaluates to true, the column displays.

See Also:

- "Providing Security Through Authorization" on page 12-20
- "Understanding Conditional Rendering and Processing" on page 3-2
- Appendix A, "Available Conditions" on page A-1

Controlling Column Breaks

You can control if a specific column repeats and how column breaks appear when printed using Break Formatting attributes. For example, suppose your report displays employee information by department number. If multiple employees are members of the same department, you can increase the readability by specifying the department number only appears once.

To create this type of column break:

1. Navigate to the appropriate Report Attributes page. See "Editing Report Attributes" on page 5-18.
2. Scroll down to Break Formatting.
3. Make a selection from the Breaks list.

Creating Forms

You can include a variety of different types of forms in your applications. You can include forms that enable users to update just a single row in a table or multiple rows at once. Application Builder includes a number of wizards you can use to create forms automatically, or you can create forms manually.

Topics in this section include:

- Creating a Form Using a Wizard
- Creating a Tabular Form
- Building a Master Detail Form
- Creating a Form Manually
- Validating User Input in Forms

Creating a Form Using a Wizard

The easiest way to create a form is to use a wizard. For example, the Form on Table or View Wizard creates one item for each column in a table. It also includes the necessary buttons and processes required to insert, update, and delete rows from the table using a primary key. Each region has a defined name and display position; all other attributes are items, buttons, processes, and branches.

To create a form using a wizard:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click Create Page.
4. Select **Form**.
5. Under Forms, select a type of form page as described in Table 5–4.

Table 5–4 Forms Page Types

Form Page Type	Description
Form on a Procedure	Builds a form based on stored procedure arguments. Use this approach when you have implemented logic or data manipulation language (DML) in a stored procedure or package.
Form on a SQL Query	Creates a form based on the columns returned by a SQL query such as an EQUIJOIN.
Form on a Table or View	Creates a form that enables users to update a single row in a database table.
Form on a Table with Report	Creates two pages. One page displays a report. Each row provides a link to the second page to enable users to update each record. **Note:** This wizard does not support tables having more than 127 columns. Selecting more than 127 columns generates an error.
Form on Web Service	Creates a page with items based on a Web service definition. This wizard creates a user input form, a process to call the Web service, and a submit button. **See Also:** "Creating a Form on a Web Service" on page 13-19
Form and Report on Web Service	Creates a page with items based on a Web service definition. This wizard creates a user input form, a process to call the Web service, a submit button, and displays the results returned in a report. **See Also**: "Creating an Input Form and Report on a Web Service" on page 13-18
Master Detail Form	Creates a form that displays a master row and multiple detail rows within a single HTML form. With this form, users can query, insert, update, and delete values from two tables or views. **See Also:** "Building a Master Detail Form" on page 5-29
Summary Page	Creates a read-only version of a form. Typically used to provide a confirmation page at the end of a wizard.
Tabular Form	Creates a form in which users can update multiple rows in a database. **See Also:** "Creating a Tabular Form" on page 5-27

6. Follow the on-screen instructions.

Creating a Tabular Form

A tabular form enables users to update multiple rows in a table. The Tabular Form Wizard creates a form to perform update, insert, and delete operations on multiple rows in a database table.

To create a tabular form:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click Create Page.
4. Select **Form**.

5. Select **Tabular Form**.

   The Tabular Form Wizard appears.

6. On Identify Table/View Owner:

   a. Specify the table or view owner on which you want to base your tabular form.

   b. Select the operations to be performed on the table (for example, **Update, Insert and Delete**).

7. On Identify Table/View Name, select a table.

8. On Identify Columns to Display:

   a. Specify whether or not to use user interface defaults. Select **Yes** or **No**.

      User interface defaults enable you to assign default user interface properties to a table, column, or view within a specified schema.

   b. Select the columns (updatable and nonupdatable) to include in the form.

      You can modify the column order or your SQL query after you create the page.

9. On Identify Primary Key, select the Primary Key column and a secondary Primary Key column (if applicable).

10. On Defaults for Primary and Foreign Keys, select a source type for the primary key column. Valid options include:

    - **Existing trigger** - Select this option if a trigger is already defined for the table. You can also select this option if you plan on specifying the primary key column source later after completing the form.
    - **Custom PL/SQL function** - Select this option if you wish to provide a PL/SQL function to generate returning key value.
    - **Existing sequence** - Select this option is you wish to pick the sequence from a list of sequences available in the selected schema.

11. On Updatable Columns, select which columns should be updatable.

12. On Identify Page and Region Attributes.

    a. Specify page and region information.

    b. Select a region template.

    c. Select a report template.

13. On Identify Tab, specify a tab implementation for this page.

14. On Button Labels, enter the display text to appear for each button.

15. On Identify Branching, specify the pages to branch to after the user clicks the Submit and Cancel buttons.

16. Click **Finish**.

Note: Do not modify the select list of a SQL statement of a tabular form after it has been generated. Doing so can result in a checksum error when you alter data in the form.

See Also: "Managing User Interface Defaults" on page 9-1

Building a Master Detail Form

A master detail form reflects a one-to-many relationship between two tables in a database. Typically, a master detail form displays a master row and multiple detail rows within a single HTML form. With this form, users can insert, update, and delete values from two tables or views.

To create a master detail form:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click Create Page.
4. Select **Form**.
5. Select **Master Detail Form**.

   The Master Detail Wizard appears.
6. On Define Master Table:
   a. Select a table or view owner.
   b. Select a table or view name.
   c. Select the columns to display.
7. On Define Detail Table:
   a. Specify to show only related tables.
   b. Select the table or view owner.
   c. Select the table or view name.
   d. Select the columns to display.
8. On Define Primary Key, select the primary key column for the master table, and then select the primary key column for the detail table.
9. On Define Master and Detail, define the relationships between master and detail tables.
10. Specify the source for the master table and detail table primary key columns.
11. On Define Master Options, specify whether or not to include master row navigation.

    If you include master row navigation, define navigation order columns. If a navigation order column is not defined, the master update form navigates by the primary key column.
12. On Choose Layout, specify the layout of the master detail pages.

    You can include the master detail as a tabular form on the same page, or add the master detail on a separate page.
13. On Page Attributes, review and edit the master page and detail page information.
14. On Identify Tabs, specify whether or not to include an optional tab set.
15. Click **Create**.

Creating a Form Manually

You can also create a form manually by performing the following steps:

- Create an HTML region (to serve as a container for your page items)
- Create items to display in the region
- Create processes and branches

To create a form manually by creating and HTML region:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Create an HTML region:
   a. Under Regions, click the **Create** icon.
   b. Select the region type **HTML**.
   c. Follow the on-screen instructions.
3. Start adding items to the page:
   - Under Items, click the **Create** icon.
   - Follow the on-screen instructions.

Processing a Form

Once you create a form, the next step is to process the data a user types by inserting into or updating the underlying database tables or views. There are three ways to process a form:

- Creating an Automatic Row (DML) Processing Process
- Creating a Process that Contains One or More Insert Statements
- Using a PL/SQL API to Process Form Values

Creating an Automatic Row (DML) Processing Process

One common way to implement a form is to manually create an Automatic Row Processing (DML) process. This approach offers three advantages. First, you are not required to provide any SQL coding. Second, Oracle Application Express performs DML processing for you. Third, this process automatically performs lost update detection. Lost update detection ensures data integrity in applications where data can be accessed concurrently.

To implement this approach you need to:

- Add items, define the Item Source Type as Database Column, and specify a case-sensitive column name.
- Select the option **Always overrides the cache value**.

To create an Automatic Row Processing (DML) process:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.

   c. Select an application.
   d. Select a page.

      The Page Definition appears.

2. Under Processes, click the **Create** icon.
3. Select the process **Data Manipulation**.
4. Select the process category **Automatic Row Processing (DML)**.
5. Specify the following process attributes:
   a. In the Name field, enter a name to identify the process.
   b. In the Sequence field, specify a sequence number.
   c. From the Point list, select the appropriate processing point. In most instances, select **Onload - After Header**.
   d. From the Type list, select **Automated Row Processing (DML)**.
6. Follow the on-screen instructions.

Creating a Process that Contains One or More Insert Statements

In this approach to form handling, you create one or more processes to handle insert, update, and delete actions. Instead of having the Application Express engine handling everything transparently, you are in complete control.

For example, suppose you have a form with three items:

- `P1_ID` - A hidden item to store the primary key of the currently displayed row in a table.
- `P1_FIRST_NAME` - A text field for user input.
- `P1_LAST_NAME` - A text field for user input.

Assume also there are three buttons labeled Insert, Update, and Delete. Also assume you have a table T that contains the columns `id`, `first_name`, and `last_name`. The table has a trigger that automatically populates the `ID` column when there is no value supplied.

To process the insertion of a new row, you create a conditional process of type PL/SQL that executes when the user clicks the Insert button. For example:

```
BEGIN
  INSERT INTO T ( first_name, last_name )
     VALUES  (:P1_FIRST_NAME, :P1_LAST_NAME);
END;
```

To process the updating of a row, you create another conditional process of type PL/SQL. For example:

```
BEGIN
    UPDATE T
       SET first_name = :P1_FIRST_NAME,
           last_name = :P1_LAST_NAME
    WHERE ID = :P1_ID;
END;
```

To process the deletion of a row, you create a conditional process that executes when the user clicks the Delete button. For example:

```
BEGIN
```

```
    DELETE FROM T
    WHERE ID = :P1_ID;
END;
```

Using a PL/SQL API to Process Form Values

For certain types of applications, it is appropriate to centralize all access to tables in a single or a few PL/SQL packages. If you created a package to handle DML operations, you can call procedures and functions within this package from an After Submit PL/SQL process to process insert, updates, and delete requests.

Populating Forms

Oracle Application Express populates a form on load, or when the Application Express engine renders the page. You can populate a form in the following ways:

- Create a process and define the type as Automated Row Fetch.
- Populate the form manually by referencing a hidden session state item.

To create an Automated Row Fetch process:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Processes, click **Create**.
3. Select the process type **Data Manipulation**.
4. Select the process category **Automatic Row Fetch**.
5. Specify the following process attributes:
   a. In the Name field, enter a name to identify the process.
   b. In the Sequence field, specify a sequence number.
   c. From the Point list, select the appropriate processing point.
   d. From the Type list, select **Automated Row Fetch**.
6. Follow the on-screen instructions.

You can also populate a form manually by referencing a hidden session state item. For example, the following code in an Oracle Application Express process of type PL/SQL would set the values of `ename` and `sal`. The example also demonstrates how to manually populate a form by referencing a hidden session state item named `P2_ID`.

```
FOR C1 in (SELECT ename, sal
FROM emp WHERE ID=:P2_ID)
LOOP
     :P2_ENAME := C1.ename;
     :P2_SAL := C1.sal;
END LOOP;
```

In this example:

- `C1` is an implicit cursor.

- The value of `P2_ID` has already been set.
- The process point for this process would be set to execute (or fire) on or before **Onload - Before Regions**.

Validating User Input in Forms

You can use validations to check data a user enters before processing. Once you create a validation and the associated error message, you can associate it with a specific item. You can choose to have validation error messages display inline (that is, on the page where the validation is performed) or on a separate error page.

Creating an inline error message involves these steps:

- Create a new validation and specify error message text.
- Associate the validation with a specific item.

Creating a Validation

To create a new validation:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Validations, click the **Create** icon.
3. When the Create Validations Wizard appears, follow the on-screen instructions.

   Validation Types are divided into two categories:

   - **Item.** These validations start with the term Item and provide common checks you may want to perform on the item with which the validation is associated.
   - **Code.** These validations require you provide either a piece of PL/SQL code or SQL query that defines the validation logic. Use this type of validation to perform custom validations that require verifying values of more than one item or accessing additional database tables.
4. Follow the on-screen instructions.

   Note: Validations cannot contain more than 3,950 characters.

Associating a Validation with a Specific Item

To associate an item with a validation and specify error message text:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

The Page Definition appears.

2. Under Validations, select the validation item you want to associate.

   The attributes page for the validation appears.

3. Scroll down to Error Message:

   - In Error message display location, verify the display location.
   - In Associated Item, select the item you want to associate with this validation.

4. Click **Apply Changes**.

About Error Message

Error message display location identifies where a validation error message displays. Validation error messages can display on an error page or inline within the existing page. Inline error messages can display in a notification area (defined as part of the page template) or within the field label.

To create a hard error that stops processes, including any remaining validations, you must display the error on an error page.

Creating Calendars

Application Builder includes a built-in wizard for generating a calendar. Once you specify the table on which the calendar is based, you can create drill-down links to information stored in specific columns.

Topics in this section include:

- About Creating Calendars
- Creating a New Calendar
- Converting an Easy Calendar to a SQL Calendar
- Editing a Calendar Title
- Editing Calendar Attributes

About Creating Calendars

Application Builder supports two calendar types:

- **Easy Calendar** creates a calendar based on schema, table, and columns you specify. The wizard prompts you to select a date column and display column.
- **SQL Calendar** creates a calendar based on a SQL query you provide. The SQL SELECT statement you provide must include at least two columns: a date column and display column.

See Also: "Calendar Display" on page 5-37

Supported Calendar Substitution Strings

Application Builder supports a number of date format substitution strings. You can view a complete list of supported substitution strings on the Calendar Templates page.

To view a list of supported substitution strings for calendars:

1. Navigate to the appropriate calendar template.
2. Expand the Substitution Stings list on the right side of the page.

See Also: "Viewing Templates" on page 7-19

Creating a New Calendar

How you create a calendar depends upon if you are adding a calendar to an existing page or adding a calendar on a new page. When creating calendars remember:

- You can only create one calendar for each page.
- The **date column** determines which days on the calendar will contain entries.
- The **display column** defines a specific row that will display a calendar date.

See Also: "Editing Calendar Attributes" on page 5-36

Adding a Calendar to an Existing Page

To add a calendar to an existing page:

1. Navigate to the Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Regions, click the **Create** icon.

   The Create Region Wizard appears.
3. Select **Calendar**.
4. Select the type of calendar you want to create:
   - **Easy Calendar** creates a calendar based on the date column and display column you specify.
   - **SQL Calendar** creates a calendar based on a SQL query you provide.
5. Follow the on-screen instructions.

Adding a Calendar to a New Page

To create a calendar on a new page:

1. Navigate to the Database Home Page.
2. Click the **Application Builder** icon.
3. Select an application.
4. Click **Create Page**.
5. Select **Calendar**.
6. Select the type of calendar you want to create:
   - **Easy Calendar** creates a calendar based on the date column and display column you specify.
   - **SQL Calendar** creates a calendar based on a SQL query you provide.
7. Follow the on-screen instructions.

Converting an Easy Calendar to a SQL Calendar

Creating an Easy Calendar is the simplest way to create a calendar. However, if you find the resulting calendar does not meet your needs, you can quickly convert it to a SQL Calendar.

To convert an Easy Calendar to a SQL Calendar:

1. Navigate to the Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Regions, click **CAL** next to the region name.

   The Calendar Attributes page appears.
3. On the Tasks list, click **Convert to SQL Based calendar**.

   Converting an Easy Calendar to a SQL Calendar, adds a Region Source section to the Region Definition. The Region Source contains the original SQL query that creates the calendar. By having accessing the Region Source, you can edit the query to meet your needs.

Editing a Calendar Title

The title that appears at the top of calendar corresponds to the region title.

To alter the region title:

1. Navigate to the Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Regions, select the region name.

   The Region Definition appears.
3. Under Region, enter a new title.
4. Click **Apply Changes**.

Editing Calendar Attributes

Once you have created a calendar, you can alter how it appears by editing its attributes.

Topics in this section include:

- Accessing the Calendar Attributes Page
- Calendar Display
- Calendar Interval

- Column Link
- Day Link

Accessing the Calendar Attributes Page

To access the Calendar Attributes page:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Regions, click **CAL** next to the region name.

   The Calendar Attributes page appears. The topics that follow describe the specific sections of the Calendar Attributes page.

Calendar Display

Use Calendar Display to specify a calendar template, date columns, and general calendar formatting.

Calendar Template determines what template is used when the Application Express engine renders a calendar. **Date Column** defines the column from the table or query containing the dates to be placed on the calendar. **Display Column** defines a specific row that displays on a calendar date.

To select another Display Column:

1. Navigate to the appropriate Calendar Attributes page.
2. Locate the Calendar Display section.
3. To specify another display column, make a selection from the Display Column list.
4. Click **Apply Changes**.

To specify a custom Display Column:

1. Navigate to the appropriate Calendar Attributes page.
2. Locate the Calendar Display section.
3. From Display Type, select **Custom**.
4. In Column Format, enter a custom column format. You can use an HTML expression and supported substitution strings.
5. Click **Apply Changes**.

See Also: "Supported Calendar Substitution Strings" on page 5-34

Calendar Interval

Use Calendar Interval to define the dates that are included in the calendar.

Begin At Start Of Interval determines when the calendar should start. Selecting this option creates a calendar that spans an entire interval (such as a month). For example:

- If **Begin at start of interval** is selected, the date is June 15th, and the display is monthly, the resulting calendar spans from June 1st to June 30th.
- If **Begin at start of interval** is not selected, the date is June 15th, and the display is monthly, the resulting calendar spans from June 15th to June 30th.

Date Item holds the date on which the calendar is based.

The next two attributes define which items hold the calendar start date and end date. You can use these attributes to create calendars that span multiple months at a time. **Item Containing Start Date** points to an item that holds the start date of the calendar. **Item Containing End Date** points to an item that holds the end date of calendar. Note that format of the date of either item must be `YYYYMMDD`.

Start of Week determines the day of the week on which the calendar starts.

Column Link

Use Column link to create a link on the column in the calendar.

To create a column link to another page:

1. Navigate to the appropriate Calendar Attributes page.
2. Scroll down to Column Link.
3. From Target is a, select **Page in this Application**.
4. In Page, specify the target page ID. To reset the pagination for this page, select **reset pagination for this page**.
5. In Request, specify the request to be used.
6. In Clear Cache, specify the pages (that is, the page IDs) on which to clear cache. Specify multiple pages by listing the page IDs in a comma-delimited list.

   You can set session state (that is, give a listed item a value) using the next two attributes: the Set these items attribute and the With these values attribute.
7. To set session state:
   a. Set these items - Enter a comma-delimited list of item names for which you would like to set session state.
   b. With these values - Enter a comma-delimited list of values for the items specified in the previous step.

      You can specify static values or substitution syntax (for example, `&APP_ITEM_NAME.`). Note that item values passed to `f?p=` in the URL cannot contain a colon (:). Additionally, item values cannot contain commas unless you enclose the entire value in backslashes (for example, `\1234,56\`).
8. Click **Apply Changes**.

See Also: "Supported Calendar Substitution Strings" on page 5-34

To create a column link to a URL:

1. Navigate to the appropriate Calendar Attributes page.
2. Scroll down to Column Link.
3. From Target is a, select **URL**.
4. In URL, enter the appropriate address.
5. Click **Apply Changes**.

Day Link

Use Day link to create a link on a day in the calendar. This attribute creates a link on an actual number (or day) on the calendar.

To create a day link to another page:

1. Navigate to the appropriate Calendar Attributes page.
2. Scroll down to Day Link.
3. From Target is a, select **Page in this Application**.
4. In Page, specify the target page ID.

   To reset the pagination for this page, select **reset pagination for this page**.
5. In Request, specify the request to be used.
6. In Clear Cache, specify the pages (that is, the page IDs) on which to clear cache. Specify multiple pages by listing the page IDs in a comma-delimited list.

   You can set session state (that is, give a listed item a value) using the next two attributes: Set these items and With these values.
7. To set session state:
   1. Set these items - Enter a comma-delimited list of item names for which you would like to set session state.
   2. With these values - Enter a comma-delimited list of values for the items specified in the previous step.

      You can specify static values or substitution syntax (for example, `&APP_ITEM_NAME.`). Note that item values passed to `f?p=` in the URL cannot contain a colon (:). Additionally, item values cannot contain commas unless you enclose the entire value in backslashes (for example, `\1234,56\`).
8. Click **Apply Changes**.

To create a day link to a URL:

1. Navigate to the appropriate Calendar Attributes page.
2. Scroll down to Day Link.
3. From Target is a, select **URL**.
4. In URL, enter the appropriate address.
5. Click **Apply Changes**.

Creating Charts

Application Builder includes built-in wizards for generating HTML and Scalable Vector Graphics (SVG) charts. Oracle Application Express supports two types of graphical charts:

- HTML
- SVG

SVG is an XML-based language for Web graphics from the World Wide Web Consortium (W3C). SVG charts are defined using an embed tag. When evaluating whether or not a SVG chart is the appropriate chart type for your application remember that:

- Some Web browsers do not support SVG charts.

- Most Web browsers that support SVG charts require users download an SVG plug-in.

Topics in this section include:

About SVG Plug-in Support

The Adobe SVG plug-in can handle data encoded in UTF-8, UTF-16, ISO-8859-1, and US-ASCII. Encoding of an SVG chart is determined by the database access descriptor (DAD) database character set. If the DAD character set is not UTF8, AL32UTF8, AL16UTF16, WE8ISO8859P1, or US7ASCII, SVG charts may not render properly in the Adobe SVG plug-in.

About Creating Charts

You define a chart in Application Builder using a wizard. For most chart wizards, you select a chart type and provide a SQL query using the following syntax:

```
SELECT link, label, value
FROM   ...
```

Where:

- `link` is a URL.
- `label` is the text that displays in the bar.
- `value` is the numeric column that defines the bar size.

For example:

```
SELECT null, ename, sal
FROM   scott.emp
WHERE  deptno = :P101_DEPTNO
```

To create a dial chart, select a dial chart type and provide a SQL query using the following syntax:

```
SELECT value , maximum_value [ ,low_value [ ,high_value] ]
FROM   ...
```

Where:

- `value` is the starting point on the dial.
- `maximum_value` is the possible highest point on the dial.
- `low_value` and `high_value` are the historical low and high values.

For example:

```
SELECT dbms_random.value(500, 1200), 1300, dbms_random.value(100, 200)
FROM DUAL
```

Table 5–5 describes the chart types available in Application Builder.

Table 5–5 Available Chart Types

Chart Type	Description
Bar (HTML)	Bar chart showing one data series with each data point represented by a bar. HTML-based. Does not require a plug-in.
Bar, Horizontal	Single series-based bar chart oriented horizontally with each data point in the series represented by a bar. SVG-based. Requires a SVG plug-in.
Bar, Vertical	Single series-based bar chart oriented vertically with each data point in series represented by a bar. SVG-based. Requires a SVG plug-in.
Cluster Bar, Horizontal	Multiple series-based bar chart oriented horizontally and clustered by a common variable with each data point in the series represented by a bar (for example, *Department sales total clustered by month of year*). SVG-based. Requires a SVG plug-in.
Cluster Bar, Vertical	Multiple series-based bar chart oriented vertically clustered by a common variable with each data point in series represented by a bar (for example, *Department sales total clustered by month of year*). SVG-based. Requires a SVG plug-in.
Dial - Sweep	Also known as an angular gauge; this chart shows either percentage of maximum value or absolute value compared to a maximum value represented as a solid area. SVG-based. Requires a SVG plug-in.
Dial	Also known as angular gauge; this chart shows either percentage of maximum value or absolute value compared to maximum value represented as a line. SVG-based. Requires a SVG plug-in.
Line	Multiple series-based line chart oriented with each line representing all data points in the series. SVG-based. Requires a SVG plug-in.
Pie	Single series-based pie chart with each slice representing a data point in the series. SVG-based. Requires a SVG plug-in.
Stacked Bar, Horizontal	Multiple series-based bar chart oriented horizontally with each data point being an absolute value in the series representing a segment of a single bar. SVG-based. Requires a SVG plug-in.
Stacked Bar, Vertical	Multiple series-based bar chart oriented vertically with each data point being an absolute value in the series representing a segment of a single bar. SVG-based. Requires a SVG plug-in.

Table 5–5 (Cont.) Available Chart Types

Chart Type	Description
Stacked Percentage Bar, Horizontal	Multiple series-based bar chart oriented horizontally with each data point being an percentage of 100% of the series represented by a segment of a single bar. SVG-based. Requires a SVG plug-in.
Stacked Percentage Bar, Vertical	Multiple series-based bar chart oriented vertically with each data point being an percentage of 100% of the series represented by a segment of a single bar SVG-based. Requires a SVG plug-in.

Note: Do not change the type of an existing chart. Instead, delete the existing chart and then re-create it.

Creating a New Chart

How you create a chart depends upon whether you are adding the chart to an existing page, or adding a chart on a new page.

Adding a Chart to an Existing Page

To add a chart to an existing page:

1. Navigate to the Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Regions, click the **Create** icon.

   The Create Region Wizard appears.
3. Select **Chart**.
4. Select the type of chart you want to create. See Table 5–5 on page 5-41.
5. Follow the on-screen instructions.

See Also: "About Creating Charts" on page 5-40

Adding a Chart to a New Page

To create a chart on a new page:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click **Create Page**.
4. Select **Chart**.
5. Select the type of chart you want to create. See Table 5–5 on page 5-41.
6. Follow the on-screen instructions.

See Also: "About Creating Charts" on page 5-40

Editing Chart Attributes

Once you have created a chart, you can alter its display by editing chart attributes on the Chart Attributes page.

To access the Chart Attributes page:

1. Navigate to the Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Regions, click **Chart** next to the name of the chart region you want to edit.

   The Chart Attributes page appears.

   Note that removing the chart title (that is, the Chart Title attribute) may negatively impact the location and display of the chart legend.

Understanding Chart Cascading Style Sheet Classes

When you create a new chart, Oracle Application Express renders it based on cascading style sheet (CSS) classes associated with the current theme. You can change the appearance of a chart by referencing another CSS or by overriding individual classes in the CSS section of the Edit Attributes page

The following sample contains the CSS classes for the dial chart in *Sample Application*. This example contains all the available CSS classes. Class names appear in boldface.

```
text{font-family:Verdana, Geneva, Arial, Helvetica, sans-serif;fill:#000000;}
tspan{font-family:Verdana, Geneva, Arial, Helvetica, sans-serif;fill:#000000;}
text.title{font-weight:bold;font-size:14;fill:#000000;}
text.moredatafound{font-size:12;}
rect.legend{fill:#EEEEEE;stroke:#000000;stroke-width:1;}
text.legend{font-size:10;}
#background{fill:#FFFFFF;stroke:none;}
rect.chartholderbackground{fill:#ffffff;stroke:#000000;stroke-width:1;}
#timestamp{text-anchor:start;font-size:9;}
text.tic{stroke:none;fill:#000000;font-size:12}
line.tic{stroke:#000000;stroke-width:1px;fill:none;}
#dial{stroke:#336699;stroke-width:2px;fill:#336699;fill-opacity:.5;}
#dial.alert{fill:#FF0000;fill-opacity:.5;}
#dialbackground{stroke:#000000;stroke-width:none;fill:none;filter:url(#MyFilter);}
#dialcenter{stroke:none;fill:#111111;filter:url(#MyFilter);}
#dialbackground-border{stroke:#DDDDDD;stroke-width:2px;fill:none;filter:url
(#MyFilter);}#low{stroke-width:3;stroke:#336699;}
#high{stroke-width:3;stroke:#FF0000;}
#XAxisTitle{letter-spacing:2;kerning:auto;font-size:14;fill:#000000;text-anchor:mi
ddle;}
#YAxisTitle{letter-spacing:2;kerning:auto;font-size:14;fill:#000000;text-anchor:mi
ddle;writing-mode:tb;}
.XAxisValue{font-size:8;fill:#000000;}
.YAxisValue{font-size:8;fill:#000000;text-anchor:end;}
.nodatafound{stroke:#000000;stroke-width:1;font-size:12;}
```

```
.AxisLine{stroke:#000000;stroke-width:2;fill:#FFFFFF;}
.GridLine{stroke:#000000;stroke-width:0.3;stroke-dasharray:2,4;fill:none;}
g.dataholder rect{stroke:#000000;stroke-width:0.5;}
.legenditem rect{stroke:#000000;stroke-width:0.5;}
```

Table 5–6 describes all supported CSS classes. Note that certain classes only apply to specific chart types.

Table 5–6 Available Chart CSS Classes

Class	Description
`text`	Defines the appearance of text that displays in a chart.
`tspan`	Defines the appearance of text that displays in a chart. `tspan` should match the definition of `text`.
`text.title`	Overrides the default chart text. Use this class for title text.
`text.moredatafound`	Defines the appearance of more datafound text.
`rect.legend`	Creates the rectangular box that holds the chart legend. To a remove the legend border, change `rect.legend` to the following: `rect.legend{fill:#CCCC99;stroke:none;}`
`text.legend`	Defines the text that appears in the chart legend.
`#background`	Creates the entire background for the SVG plug-in. For a solid white background with no border, change `#background` to the following: `#background{fill:#FFFFFF;stroke:#FFFFFF;stroke-wid th:2;}`
`rect.chartholderbackground`	Not applicable to pie and dial charts. Creates the background of the rectangle that holds the chart data. For a clear background, change `rect.chartholderbackground` to the following: `rect.chartholderbackground(display:none;)`
`#timestamp`	Only applicable if the Asynchronous Update chart attribute is set to Yes. Controls the appearance of the update timestamp test. To disable the display of the timestamp, use defines `#timestamp` as follows in the Custom CSS, Inline attribute. `"#timestamp{display:none;}"` **See Also:** "Enabling Asynchronous Updates" on page 5-46
`text.tic`	Dial charts only. Defines the numbers on a dial chart.
`line.tic`	Dial charts only. Defines the graduation mark that displays directly beneath the number on a dial chart.
`#dial`	Dial charts only. Defines the value that displays on the dial chart.
`#dial.alert`	Dial charts only. Defines a value (called an alert value) that renders on in a dial chart using a different display.
`#dialbackground`	Dial charts only. Creates the background of a dial chart.

Table 5–6 (Cont.) Available Chart CSS Classes

Class	Description
`#dialcenter`	Dial charts only. Creates the center of the dial on a dial chart.
`#dialbackground-border`	Dial charts only. Works in conjunction with `#dialbackground` to create specific graphic effect.
`#low`	Dial charts only. Defines a historical low watermark of the data being displayed on a chart.
`#high`	Dial charts only. Defines historical high watermark of the data being displayed on a chart.
`#XAxisTitle`	Defines the title that appears on the x-axis
`#YAxisTitle`	Defines the title that appears on the y-axis.
`.XAxisValue`	Defines the value that appears on the x-axis.
`.YAxisValue`	Defines the value that appears on the y-axis.
`.AxisLabel`	Similar to the axis value.
`.nodatafound`	Defines the text element that displays if no information is available.
`.AxisLine`	Indicates zero on charts that have negative values.
`.GridLine`	Creates the horizontal and vertical lines on the chart.
`g.dataholder rect`	Applies a blanket style to all data that displays in the chart.
`.legenditem rect`	Applies a blanket style to all rectangular items in the legend.

Referencing a Custom Cascading Style Sheet

You can reference a custom cascading style sheet for a chart using the CSS section of the Chart Attributes page. When you reference an external CSS, you can reference it entirely or simply override specific styles.

To reference a custom chart CSS:

1. Upload the CSS to Application Builder. See "Uploading Cascading Style Sheets" on page 7-44.
2. Create a chart. See "Creating a New Chart" on page 5-42.
3. Navigate to the Page Definition:
    a. Navigate to the Database Home Page.
    b. Click the **Application Builder** icon.
    c. Select an application.
    d. Select a page.

       The Page Definition appears.
4. Under Regions, click **Chart** next to the region name.

   The Chart Attributes page appears.
5. Scroll down to the CSS section.
6. From Use Custom CSS, select **Yes**.

7. To reference an external CSS exclusively:

   a. In Custom CSS, Link, enter a link to a custom CSS. For example:

      ```
      #IMAGE_PREFIX#themes/theme_4/svg.css
      ```

   b. Specify that the CSS should be used exclusively. In Custom CSS, Inline enter the following:

      ```
      /**/
      ```

8. To reference a custom CSS and override specific styles:

   a. In Custom CSS, Link, enter a link to a custom style sheet. For example:

      ```
      #IMAGE_PREFIX#themes/theme_4/svg.css
      ```

   b. In Custom CSS, Inline, enter the custom CSS styles you want to override.

Specifying Custom CSS Styles Inline

You can override specific styles within the default CSS, using the Custom CSS, Inline attribute on the Chart Attributes page.

To override specific styles within the default CSS:

1. Create a chart. See "Creating a New Chart" on page 5-42.
2. Navigate to the Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
3. Under Regions, click **Chart** next to the region name.

   The Chart Attributes page appears.
4. Scroll down to CSS.
5. From Use Custom CSS, select **Yes**.
6. In Custom CSS, Inline, enter the custom CSS styles you want to override.

Enabling Asynchronous Updates

You can create charts that monitor information by enabling the Asynchronous Update attribute on the Chart attributes page. Enabling this attribute updates the chart to reflect changes in the underlying data within a specified time interval.

To enable asynchronous updates:

1. Create a chart. See Creating a New Chart on page 5-42.
2. Navigate to the Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.

   d. Select a page.

      The Page Definition appears.

3. Under Regions, click **Chart** next to the region name.

   The Chart Attributes page appears.

4. Scroll down to Refresh.
5. From Asynchronous Update, select **Yes**.
6. In Update Interval (Seconds), enter the interval in seconds between chart updates. For optimal performance, select an interval that is greater than 2 seconds.

When Asynchronous Update is enabled, a timestamp displays on the chart indicating that last update.

To disable the Asynchronous Update timestamp:

1. Navigate to the Chart Attributes page.
2. Locate the CSS section.
3. From Use Custom CSS, select **Yes**.
4. In Custom CSS, Inline edit `#timestamp` as follows:

   ```
   #timestamp{display:none;}
   ```

Displaying Charts in Other Languages

To display a chart in another language, you edit the `text` and `tspan` classes to reflect the correct language.

To display a chart in another language:

1. Navigate to the Chart Attributes page. See "Editing Chart Attributes" on page 5-43.
2. Scroll down to CSS.
3. From Use Custom CSS, select **Yes**.
4. In Custom CSS, Inline, edit the `text` and `tspan` classes to reflect the correct language. The following example demonstrates how to change a chart to Korean:

   ```
   text{font-family:Batang;fill:#000000;}
   tspan{font-family:Batang;fill:#000000;}
   ```

Creating Buttons

As you design your application, you can use buttons to direct users to a specific page or URL, or to post or process information (for example, by creating Create, Cancel, Next, Previous, or Delete buttons).

Buttons can perform two different types of actions. A button can submit a page and then redirect to a URL. Alternately, a button can branch to a URL without submitting the page.

Topics in this section include:

- Creating a Button Using a Wizard
- Creating Multiple Buttons
- Understanding the Relationship Between Button Names and REQUEST

- About Branching with Buttons
- Displaying Buttons Conditionally

Creating a Button Using a Wizard

You create a button by running the Create Button Wizard from the Page Definition. Each button resides in a region. A region is an area on a page that serves as a container for content.

To create a new button:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. If necessary, create an HTML region. See "Customizing Regions" on page 7-2.
3. Under Buttons, click the **Create** icon.

   The Create Button Wizard appears.
4. Select a region to contain the button.
5. Select a position for the button:
   - **Create a button displayed among this region's items**
   - **Create a button in a region position**

   Select **Create a button displayed among this region's items** to add a button to a region as if it was an item (for example, to add a button directly to the right of a form field).
6. If you select **Create a button in a region position**:
   a. Specify the Button Name and Label.
   b. Select a Button Type: **HTML Button (Default)**, **Image**, or **Template Driven**

      Select **Button is Reset** to create an Undo button. When enabled, this type of button resets the page values to the state they were in when the page was initially rendered.
   c. Select an Action.

      Selecting **Submit page and redirect to URL** submits the current page to the Application Express engine whenever a user clicks the button.

      Selecting **Redirect to URL without submitting page** avoids submitting the page. Choose this action when submitting the page for processing is not necessary (for example, a Cancel button). This action avoids processing in the database and therefore reduces the load.
7. If you select **Create a button displayed among this region's items**:
   a. Specify the Button Name and Sequence.
   b. Specify if the button displays at the beginning of a new line or new field.
   c. Specify a Label.

   d. Enter the value of Request.
   e. Select the Button Style.
8. Follow the on-screen instructions

   See Also: "Understanding the Relationship Between Button Names and REQUEST" on page 5-49

Creating an HTML Button

Buttons can be placed in a predefined region template position or among items in a form. To create an HTML button, select one of the following while running the Create Button Wizard:

- Under Task, select Create a button in a region position.
- Under Button Type, select a button type and then HTML Button (default).

Creating Multiple Buttons

You can create multiple buttons within the same region at once using the Create Multiple Buttons Wizard.

To create multiple buttons at once:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. If necessary, create an HTML region. See "Customizing Regions" on page 7-2.
3. Under Buttons, click the **Create** icon.

   The Create Button Wizard appears.
4. Select **Create Multiple Buttons** at the bottom of the page.

   The Create Multiple Button Wizard appears.
5. From Place Buttons in Region, select the region to contain the buttons.
6. From Template, select a template.
7. In HTML Attributes, specify HTML attributes for these buttons. This text will be added to the HTML element definition. For example, you could set the class of a text button as follows:

   ```
   class="myclass"
   ```
8. To quickly populate the remaining fields, make a selection from the Quick Button list on the right side of the page.
9. Click **Create Buttons**.

Understanding the Relationship Between Button Names and REQUEST

The name you give a button determines the value of the built-in attribute `REQUEST`. You can reference the value of `REQUEST` from within PL/SQL using the bind variable

:REQUEST. By using this bind variable, you can conditionally process, validate, or branch based on which button the user clicks. You can also create processes that execute when the user clicks a button. You can also use a more complex condition as demonstrated in the following examples:

```
If :REQUEST in ('EDIT','DELETE') then ...
If :REQUEST != 'DELETE' then ...
```

These examples assume the existence of buttons named EDIT and DELETE. You can also use this syntax in PL/SQL Expression conditions. Be aware, however, that the button name case is preserved. In other words, if you name a button LOGIN, then a request looking for the name *Login* will fail. For example:

```
<input type="BUTTON" value="Finish" onClick="javascript:doSubmit('Finish');">
```

Note that in this example *Finish* is the name of the `REQUEST` and this example is case-sensitive.

About Branching with Buttons

Each page can include any number of branches. A branch links to another page in your application or to a URL. The Application Express engine considers branching at different times during page processing. You can choose to branch before processing, before computation, before validation, and after processing. Like any other control in Application Builder, branching can be conditional. For example, you can branch when a user clicks a button. When you create a branch, you associate it with a specific button. The branch will only be considered if a user clicks the button.

See Also: "Controlling Navigation Using Branches" on page 6-8

Displaying Buttons Conditionally

You can choose to have a button display conditionally by editing attributes on the Edit Pages Button page.

To have a button display conditionally:

1. Create the button. See "Creating a Button Using a Wizard" on page 5-48.
2. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
3. Under Buttons, select the button name.

   The attributes page for the button appears.
4. Scroll down to Conditional Button Display.
5. Make a selection from the Condition Type list.
6. Enter an expression in the fields provided.
7. Click **Apply Changes**.

See Also: "About Bind Variables" on page 3-9

Creating Items

An item is part of an HTML form. An item can be a text field, text area, password, select list, check box, and so on. Item attributes affect the display of items on a page. For example, these attributes can impact where a label displays, how large an item will be, and if the item will display next to or below the previous item.

There are two types of items: page items and application items. Page items are placed on a page and have associated user interface properties, such as Display As, Label and Label Template. Application items are not associated with a page and therefore have no user interface properties. You can use an application item as a global variable.

Topics in this section include:

- Creating a Page-Level Item
- Referencing Item Values
- Editing Page Item Attributes
- Displaying Conditional or Read-Only Page Items
- Working with a Multiple Select List Item
- Creating an Application-Level Item
- Populating an Alternative Date Picker Format

See Also: "How Item Attributes Affect Page Layout" on page 7-6 and "Using Substitution Strings" on page 3-13

Creating a Page-Level Item

You create a page-level item by running the Create Item Wizard from the Page Definition.

To create a new page-level item:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. If necessary, create an HTML region. See "Customizing Regions" on page 7-2.
3. Under Items, click the **Create** icon.
4. Select an item type. See "About Item Types" on page 5-52.
5. Follow the on-screen instructions

About Item Naming Conventions

When specifying an item name, remember the following rules. Item names must:

- Not have quotation marks
- Begin with a letter or a number, and subsequent characters can be letters, numbers, or underscore characters,
- Be case-insensitive.

- Should not exceed 30 characters.
- Cannot contain letters outside the base ASCII character set.

About Item Types

When you create an item, you specify an item type. Once you create an item, these types appear on the Display As list on the Edit Page Item page. Table 5–7 describes available item types.

Table 5–7 Available Item Types

Item Type	Description

Check box	Displayed using a list of values. A list of values is required for items displayed as check boxes. The value corresponding to a checked box is returned in a single colon-delimited string.
	
	The following example demonstrates how to create a single check box that returns `YES`. This example would display both a check box and a field label.
	
	`SELECT NULL display_text, 'YES' return_value FROM DUAL;`
	
	This example includes the additional text *Click to select*.
	
	`SELECT 'Click to select' display_text, 'YES' return_value FROM DUAL;`
	
	See Also: "HTMLDB_UTIL" on page 15-1 for information about breaking up returned values

Date Picker	Displays a text field with a Calendar icon next to it. When clicked, this icon displays a small calendar from which the user can select a date and a time (optional). If the format you need is not included in the Display As list, select **Date Picker (use application format mask)**. When using a format mask, your application looks for the format in an item called `PICK_DATE_FORMAT_MASK`. Note that you need to populate this item before this item type will work. **See Also:** "Populating an Alternative Date Picker Format" on page 5-59
Display As Text	Available Display As Text subtypes include: ■ **Display as Text (based on LOV, does not save state)** - Displays the display value from an LOV by matching the item's source value with the LOV's return value. ■ **Display as Text (based on LOV, saves state)** - Same as the previous option, but also generates a form item that gets submitted with the page to pass the return value into session state. ■ **Display as Text (based on PL/SQL, does not save state)** - Displays the output of an anonymous PL/SQL block. ■ **Display as Text (does not save state)** - Displays the item's source value on the page without creating a form item. ■ **Display as Text (escape special characters, does not save state)** - Displays the item's source value with special characters ('<','>','&') escaped. ■ **Display as Text (saves state)** - Displays the item's source value and creates a form items which gets submitted with the page to pass the value into session state.
File Browse	Displays a text field with a Browse... button. This enables the user to locate a file on a local file system and upload it. Oracle Application Express provides a table for these files to be uploaded to as well as an API to retrieve the files. **See Also:** "Understanding the Security Risks of File Upload Tables" on page 12-14

Table 5–7 (Cont.) Available Item Types

Item Type	Description
Hidden	Renders an HTML hidden form element. Session state can be assigned and referenced just like a text field.
List Managers	Based on a list of values. This item enables you to manage a list of items by selecting and adding to a list. The list of values display as a popup.
Multiple Select	Renders as a multiselect HTML form element. When submitted, selected values are returned in a single colon-delimited string. You can break up the values using the `HTMLDB_UTIL API`. **See Also:** "Working with a Multiple Select List Item" on page 5-57 and "HTMLDB_UTIL" on page 15-1
Password	Renders as an HTML password form element.

Popup List of Values	Renders as a text field with an icon. When the user clicks the icon, a popup window appears with a list of values represented as a series of links. When the user makes a selection from this list, the selected value will be placed in the text field. You control popup LOVs through templates. You can only specify one popup LOV template for each application
	
	Using a popup LOV is a good choice for lists of values that are too large to return on a single page.
	
	There are two types of Popup LOVs: one that fetches a set of rows when the window pops up and one that does not.
	
	Popup LOVs must be based on a query that selects two columns with different column aliases. For example:
	
	`SELECT ename name, empno id`
	`   FROM emp`
	
	If one of the columns is an expression, remember to use an alias. For example:
	
	`SELECT ename||' '||job display_value, empno FROM emp`

Radio	Renders as an HTML radio group form element, based on a list of values. Choose **Radiogroup with Submit** to have the page submitted when the radio button is selected. The following example displays employee names (`ename`), but returns employee numbers (`empno`): `SELECT ename, empno FROM emp`

Table 5–7 (Cont.) Available Item Types

Item Type	Description
Select List	Displays using a list of values. A list of values is required for items displayed as a select list. Select lists are rendered using the HTML form element `<select>`. The values in a select list are determined using a named list of values or a list of values defined at the item level. You can specify the NULL display value and NULL return value. The following example would return employee names (`ename`) and employee numbers (`empno`) from the `emp` table. Note that column aliases are not required and are included in this example for clarity. `SELECT ename display_text, empno return_value FROM emp` Oracle Application Express provides additional enhancements to a standard HTML select list: ■ **Select List with Submit** - Submits the page when the user changes its selected value. Upon submit, the REQUEST will be set to the name of the item that represents the select list, allowing you to execute conditional computations, validations, processes, and branches. ■ **Select List with Redirect** - Redirects the user back to the same page, setting ONLY the newly selected value of the select list in session state. ■ **Select List Returning URL Redirect** - Based on a list of values with URLs as the return values. Changing the value of the select list causes the browser to redirect to the corresponding URL. ■ **Select List with Branch to Page** - Based on list of values with page IDs as return values. Changing the selected value in the select list causes the Application Express engine to branch to the corresponding page. **Note**: Long select lists can result in error. If you have a long select list that generates an error try using a Popup List of Values instead.

Table 5–7 (Cont.) Available Item Types

Item Type	Description
Stop and Start Table	Forces the close of an HTML table using the `</table>` tag and starts a new HTML table. You can use this item type of reset the column width in the middle of the region. Note that a Stop and Start Table item only displays its label. You can prevent the label from displaying at all by setting it to null. To do this, you simply remove the default label.
Text	Displays as an HTML text field containing a maximum of 30,000 bytes of text. You control the maximum length and display width by editing the Height and Width item attribute. Available Text display options include: ■ **Text Field** - Renders as a text field. ■ **Text Field (Disabled, does not save state)** - Displays a read-only version of a display value from a list of values by using the item's value in session state to look up the corresponding display value in the associated list of values. The value displayed on the screen is not saved in session state upon submit. ■ **Text Field (Disabled, saves state)** - Displays a read-only version of a display value from a list of values by using the item's value in session state to look up the corresponding display value in the associated list of values. ■ **Text Field (always submits page when Enter pressed)** - Displays a read-only version of the value in session state. Upon submit, the value displayed is saved in session state. ■ **Text Field with Calculator Popup** - Renders as a text field with an icon next to it. When clicked, the icon displays a small window containing a calculator. Calculations are placed back in the text field.
Text Area	Renders as an HTML text area. There is no maximum length for an item displayed as a text area. You control the height and width by editing the Height and Width item attribute. Additional available Text Area Display As options include: ■ **Text Area (auto height)** - Varies the height based on the amount of text. Use this option to have a large text area if you have a lot of data and a small text area if you have little or no data. ■ **Text Area with Counter** - Includes a counter that displays the number of bytes entered in the field. ■ **Text Area with Spell Checker** - Provides a popup English language spell checker. ■ **Text Area with HTML Editor** - Provides basic text formatting controls. Note that these controls may not work in all Web browsers.

Referencing Item Values

You can reference item values stored in session state in regions, computations, processes, validation, and branches. Table 5–8 describes the supported syntax for referencing item values.

See Also: "Managing Session State Values" on page 3-6

Table 5–8 Syntax for Referencing Item Values

Type	Syntax	Description
SQL	`:MY_ITEM`	Standard bind variable syntax for items whose names are no longer than 30 bytes. Use this syntax for references within a SQL query and within PL/SQL.
PL/SQL	`V('MY_ITEM')`	PL/SQL syntax referencing the item value using the `V` function. **See Also:** "Oracle Application Express APIs" on page 15-1
PL/SQL	`NV('MY_NUMERIC_ITEM')`	Standard PL/SQL syntax referencing the numeric item value using the `NV` function. **See Also:** "Oracle Application Express APIs" on page 15-1
Static Text	`&MY_ITEM`	Static text.
Static Text (exact)	`&MY_ITEM.`	Static text. Exact Substitution.

You can set the value of an item in your application using any of the following methods:

- For page items, use the Source Attribute to set the item value.

  From the Page Definition, select the item name to view the Edit Page Item page. Scroll down to Source and edit the appropriate fields.

  You can also set the value of an item in any region based on PL/SQL or a process using the following syntax:

```
BEGIN
 :MY_ITEM :=  'new value';
END;
```

- Pass the value on a URL reference using `f?p` syntax. For example:

```
f?p=100:101:10636547268728380919::NO::MY_ITEM:ABC
```

- Set the value using a computation. Computations are designed to set item values. For example:

```
TO_CHAR(SYSDATE,'Day DD Month, YYYY');
```

- Use the PL/SQL API to set an item value within a PL/SQL context. For example:

```
HTMLDB_UTIL.SET_SESSION_STATE('MY_ITEM',SYSDATE);
```

See Also: "Clearing Session State" on page 3-7, "Oracle Application Express APIs" on page 15-1, and "About Cross-Site Scripting Protection" on page 12-1

Editing Page Item Attributes

Once you create an item, you can edit it on the Edit Page Item page.

To edit item attributes:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.

   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Items, select the item name.

   The attributes page for the item appears.
3. Edit the appropriate item attributes. For information on a specific attribute, see item Help.
4. Click **Apply Changes**.

See Also: "About Cross-Site Scripting Protection" on page 12-1

Displaying Conditional or Read-Only Page Items

You can choose to have an item display conditionally or as read-only by editing attributes on the Edit Pages Item page.

To display a conditional or read-only item:

1. Create the item. See "Creating a Page-Level Item" on page 5-51.
2. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
3. Under Items, select the item name.

   The attributes page for the item appears.
4. To display an item conditionally:
   a. Scroll down to Conditions.
   b. Make a selection from the Condition Type list.
   c. Enter an expression in the fields provided.
5. To make an item read-only:
   a. Scroll down to Read Only Display Settings.
   b. Make a selection from the Read Only Condition Type list.
   c. Enter an expression in the fields provided.
6. Click **Apply Changes**.

Working with a Multiple Select List Item

A multiple select item renders as a multiple select list form element. When submitted, selected values are returned in a single colon-delimited string. You can handle values in this format in two ways:

- Using the `INSTR` function
- Using the `HTMLDB_UTIL.STRING_TO_TABLE` function

Using HTMLDB_UTIL.STRING_TO_TABLE to Convert Selected Values

Suppose you had a report on the `EMP` and `DEPT` tables that is limited by the departments selected from a Department multiple select list. First, you create the multiple select item, P1_DEPTNO, using the following query:

```
SELECT dname, deptno
FROM dept
```

Second, you return only those employees within the selected departments as follows:

```
SELECT ename, job, sal, comm, dname
FROM emp e, dept d
WHERE d.deptno = e.deptno
AND instr(':'||:P1_DEPTNO||':',':'||e.deptno||':') > 0
```

Next, assume you want to programmatically step through the values selected in the multiple select item, P1_DEPTNO. To accomplish this, you would convert the colon-delimited string into a PL/SQL array using the `HTMLDB_UTIL.STRING_TO_TABLE` function. The following example demonstrates how to insert the selected departments into an audit table containing the date of the query.

```
DECLARE
    l_selected HTMLDB_APPLICATION_GLOBAL.VC_ARR2;
BEGIN
  --
  -- Convert the colon separated string of values into
  -- a PL/SQL array

  l_selected := HTMLDB_UTIL.STRING_TO_TABLE(:P1_DEPTNO);

  --
  -- Loop over array to insert department numbers and sysdate
  --

  FOR i IN 1..l_selected.count
  LOOP
    INSERT INTO report_audit_table (report_date, selected_department)
        VALUES (sysdate, l_selected(i));
  END LOOP;
END;
```

See Also: "STRING_TO_TABLE Function" on page 15-30

Creating an Application-Level Item

Application level items do not display, but are used as global variables to the application.

To create a new application-level item:

1. Navigate to the Database Home Page.
2. Click the **Application Builder** icon.
3. Select an application.
4. When Application Builder appears, click **Shared Components**.
5. Under Logic, select **Items**.

   The Application Items page appears.

6. To create a new application item, click Create.
7. Follow the on-screen instructions.

About the Application Items Page

Once you create a application item, it appears on the Application Items page. You control how the Application Items page displays by making a selection from the View list. Available options include:

- **Icons** (the default) displays each application item as a large icon. To edit an application item, click the appropriate icon.
- **Details** displays each application item as a line in a report. To edit an application item, click the name.

Accessing Application Item History

You can view a history of changes to application items by clicking **History** at the top of the Application Items page.

Populating an Alternative Date Picker Format

If you need to create a Date Picker item, but the format you need does not appear in the Display As list, select **Date Picker (use application format mask)**. When an application uses this type of date picker, the Application Express engine derives the date format from an item named `PICK_DATE_FORMAT_MASK`. You can populate this item in two ways:

- By defining an application substitution string named `PICK_DATE_FORMAT_MASK`
- By creating an application-level item named `PICK_DATE_FORMAT_MASK`

Defining PICK_DATE_FORMAT_MASK as an Application Substitution String

One approach to populating `PICK_DATE_FORMAT_MASK` is to create an application substitution string. You define application-level substitution strings on the Edit Application Attributes page. Remember that an application-level substitution string is a static value and cannot be altered at run time.

To define a new application substitution string named `PICK_DATE_FORMAT_MASK`:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.

   Application Builder appears.
3. Click the Edit Attributes icon.
4. Click **Edit Standard Attributes**.
5. Scroll down to Static Substitution Strings.
6. Create a new static substitution string named `PICK_DATE_FORMAT_MASK`:
   a. In Substitution String, enter the name `PICK_DATE_FORMAT_MASK`.
   b. In Substitution Value, enter a value for your date format (for example, `Month DD, YYYY`).

Defining an Application-Level Item Named PICK_DATE_FORMAT_MASK

Another approach to populating `PICK_DATE_FORMAT_MASK` is to create an application-level item named `PICK_DATE_FORMAT_MASK`. This approach enables you to control any items rendered as **Date Picker (use application format mask)** by simply setting the value of this item. Plus, you can set the value of `PICK_DATE_FORMAT_MASK` using a computation from anywhere within your application.

If you want to provide the user with a list of date formats as preferences, you will need to create an application-level item named `PICK_DATE_FORMAT_MASK` and then use a computation to set the value of this item based upon the user's selection.

See Also: "Creating an Application-Level Item" on page 5-58

Creating Lists of Values

A list of values (LOV) is a static or dynamic set of values used to display a specific type of page item, such as popup lists of values, a select list, a check box, a radio group, or multiple select lists.

Topics in this section include:

- Creating Named LOVs at the Application Level
- About Static LOVs
- Editing an Existing LOV
- Referencing Session State Within a LOV
- Referencing a Null Value in an Item Based on a LOV
- Accessing LOV Reports

See Also: "Creating Items" on page 5-51

Creating Named LOVs at the Application Level

You define named (or shared) LOVs at the application level by running the Create LOV Wizard and adding them to the List of Values repository. All LOVs can be defined as static or dynamic. Static lists are based on predefined pairs of display values and return values. Dynamic lists are based on a SQL query you write that selects values from a table.

To create a named LOV:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.

   The Application home page appears.
3. Click the **Shared Components** icon.
4. Under User Interface, select **Lists of Values**.
5. To create a new LOV, click Create.
6. Follow the on-screen instructions.

   New named LOVs are added to the List of Values repository.

About the List of Values Page

Once you create a LOV, it appears on the List of Values page. You control how the page displays by making a selection from the View list. Available options include:

- **Icons** (the default) displays each LOV as a large icon. To edit a LOV, click the appropriate icon.

- **Details** displays each LOV as a line in a report. To edit a LOV, click the name.

About Static LOVs

Static LOVs are based on a static list of display values and return values you specify when you run the Create LOV Wizard. To create a static LOV, you run the Create LOV Wizard and select the LOV type **Static**. Oracle Application Express stores the display values, return values, and sort sequence you specify in the List of Values repository. Once you add a static LOV to the repository, you can create an item and display it as a check box, radio group, select list, or popup list based on this definition.

Editing an Existing LOV

To edit an existing LOV, select the LOV on the Lists of Values page.

To edit a LOV:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click **Shared Components**.
4. Under User Interface, select **Lists of Values**.
5. Select an LOV.

   The Edit List of Values page appears.
6. Edit the appropriate attributes and click **Apply Changes**.

Bulk Edit of Static LOVs

You can edit the display values of all static LOVs by clicking the Grid Edit button on the Edit List of Values page.

To perform a bulk edit of static LOVs:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click **Shared Components**.
4. Under User Interface, select **Lists of Values**.

   By default, LOVs display as icons.
5. Change the default display. Select **Details** from the View list and click **Go**.
6. Locate Static LOV and select the LOV name.
7. Click the **Grid Edit** button located under Subscription.
8. Edit the appropriate display values and click **Apply Changes**.

Referencing Session State Within a LOV

You can reference session state by using bind variables. In the following example, this LOV only works if the item called *my_deptno* contains a valid department number.

```
SELECT ename, empno FROM emp WHERE deptno = :my_deptno
```

Referencing a Null Value in an Item Based on a LOV

LOVs have a null display value option and a null return value option. The null display value is the value the end user sees in the list indicating the no selection from the proper (non-null) values of the list will be made. When a user selects a null display value, the LOV's null return value is sent to the application when the page is submitted. If the developer has left the null return value unspecified (or empty), the actual value transmitted is not an empty string or an Oracle null, but the literal `%null%`. The application must be prepared to deal with this literal and treat is as the null selection.

Be aware of this behavior when writing code to evaluate submitted values. For example, suppose a page evaluates the submitted item `P1_X` and you need to use the PL/SQL expression `replace(:P1_X,'%'||'null%',null)` to prepare the item for permanent storage in session state or for passing to DML or other APIs.

To avoid problems, be aware of the appropriate way to code `%null%` in expressions that occur in page computations, processes, and validations. You must break up the string so that the application does not replace `%null%` with a null value in the page metadata when it is saved. Consider the following example:

```
'%'||'null%'
```

Accessing LOV Reports

Application Builder includes a number of reports designed to help you better manage LOVs.

To access LOV reports:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. When Application Builder appears, click **Shared Components**.
4. Under User Interface, select **Lists of Values**.
5. Select one of the following tabs at the top of the page:
   - **Search**
   - **Subscription**
   - **Utilization**
   - **History**
6. Follow the on-screen instructions.

Search

Click **Search** to display the Search Dynamic Lists of Values page. Use this page to search the queries that makes up dynamic LOVs. Enter a query in the Query Contains field and click **Go**.

Subscription

Click **Subscription** to display the List of Values Subscription page. This page displays all subscribed LOVs in your application.

Utilization
Click **Utilization** to display the List of Values Utilization page. This page displays LOVs used in the current application. To edit an LOV, click the LOV name.

History
Click **History** to display the List of Values History page. This page displays a history of recently changed LOVs by date.

Using Shortcuts
By using shortcuts you can avoid repetitive coding of HTML or PL/SQL functions. You can use a shortcut to define a page control such as a button, HTML text, a PL/SQL procedure, or HTML. Once defined, you can invoke a shortcut using specific syntax unique to the location in which the shortcut is used. Shortcuts can be referenced many times, thus reducing code redundancy.

This section contains the following topics:

- About Shortcut Types
- Defining Shortcuts
- Accessing Shortcut Reports

About Shortcut Types
When you create a new shortcut, you must specify the type of shortcut you want to create. Oracle Application Express supports the following shortcut types:

- PL/SQL Function Body
- HTML Text
- HTML Text with Escaped Special Characters
- Image
- Text with JavaScript Escaped Single Quotes
- Message
- Message with JavaScript Escaped Special Quotes

Text with JavaScript Escaped Single Quotes
Use this type of shortcut to reference a shortcut inside of a JavaScript literal string. This shortcut defines a text string. When the shortcut is referenced, it escapes the single quotation marks required for JavaScript.

Message
Use this type of shortcut to reference a translatable message at run time. Note that since this shortcut does not have a shortcut body, the name of the shortcut must match the corresponding message name. At run time, the name of the shortcut expands to the text of the translatable message for the current language.

Message with JavaScript Escaped Single Quotes
Use this type of shortcut to reference a shortcut inside of JavaScript literal string and reference a translatable message at run time.

See Also: "About Translating an Application and Globalization Support" on page 14-1

Defining Shortcuts

Before you can incorporate a shortcut in your application, you must define it and add it to the Shortcuts repository. You reference new shortcuts using the following syntax:

```
"MY_SHORTCUT"
```

Note that the shortcut name must be capitalized and enclosed in quotation marks.

To define a new shortcut:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. When Application Builder appears, click **Shared Components**.
4. Under User Interface, select **Shortcuts**.
5. Click **Create**.
6. Select one of the following creation methods:
   - **From Scratch**
   - **As a Copy of an Existing Shortcut**
7. Follow the on-screen instructions.

New shortcuts are added to the Shortcut repository and are available for use within the following locations:

- The Region Source attribute of regions defined as HTML Text (with shortcuts). See "Customizing Regions" on page 7-2.
- Region Header and Footer Text attribute. See "Specifying a Region Header and Footer" on page 7-5.
- Item Label attributes and Default Value attribute. See "Items" on page 4-25.
- Region Templates attributes. See "Editing Templates" on page 7-20.

About the Shortcuts Page

Once you create a shortcut, it appears on the Shortcuts page. You control how the page displays by making a selection from the View list. Available options include:

- **Icons** (the default) displays each shortcut as a large icon. To edit a shortcut, click the appropriate icon.
- **Details** displays each shortcut as a line in a report. To edit a shortcut, click the name.

Accessing Shortcut Reports

Application Builder includes a number of reports designed to help you better manage LOVs.

To access shortcut reports:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.

3. When Application Builder appears, click **Shared Components**.
4. Under User Interface, select **Shortcuts**.
5. Click one of the following buttons:
   - **Subscription**
   - **History**

Note: The Subscription and History buttons only appear after you create a shortcut.

Subscribed Shortcuts

Click **Subscription** to display the Subscribed Shortcuts page. This page displays all subscribed shortcuts in your application.

Shortcut History

Click **History** to display the Shortcut History page. This page displays a history of recently changed shortcuts by date.

Incorporating JavaScript into an Application

Adding JavaScript to a Web applications is a great way to add features that mimic those found client/server applications without sacrificing all of the benefits of Web deployment. Oracle Application Express includes multiple built-in interfaces especially designed for adding JavaScript.

Remember that JavaScript is not appropriate for data intensive validations. For example, to verify that a name is contained within a large database table, you would need to pull down every record to the client, creating a huge HTML document. In general, complex operations are much better suited for server-side Application Express validations instead of JavaScript.

This section contains the following topics:

- Referencing Items Using JavaScript
- Incorporating JavaScript Functions
- Calling JavaScript from a Button

See Also: "Understanding Validations" on page 4-32

Referencing Items Using JavaScript

When you reference an item, the best approach is to reference by ID. If you view the HTML source of an Oracle Application Express page in a Web browser, you would notice that all items have an id attribute. This id corresponds to the name of the item, not the item label. For example, if you create an item with the name `P1_FIRST_NAME` and a label of `First Name`, the ID will be P1_FIRST_NAME.

Knowing the item ID enables you to use the JavaScript method `getElementById()` to get and set item attributes and values. The following example demonstrates how to reference an item by ID and display its value in an alert box.

```
<script language="JavaScript1.1" type="text/javascript">
  function firstName(){
    alert('First Name is ' + document.getElementById('P1_FIRST_NAME').value );
  }
 // or a more generic version would be
  function displayValue(id){
    alert('The Value is ' + document.getElementById(id).value );
  }
</script>

  // Then add the following to the "Form Element Attributes" Attribute of the
item:
  onchange="displayValue('P1_FIRST_NAME');"
```

Incorporating JavaScript Functions

There are two primary places to include JavaScript functions:

- In the HTML Header attribute of the page
- In a .js file in the page template

See Also: "Text with JavaScript Escaped Single Quotes" on page 5-63 for information about referencing a shortcut inside of a JavaScript literal string

Incorporating JavaScript in the HTML Header Attribute

One way to include JavaScript into your application is to add it to the HTML Header attribute of the page. This is a good approach for functions that are very specific to a page as well as a convenient way to test a function before you include it in the `.js` file.

You can add JavaScript functions to a page by simply entering the code into the HTML Header attribute of the Page Attributes page. For example, adding the following would make the `test` function accessible from anywhere on the current page.

To add JavaScript code in the HTML Header attribute:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Select a page.
4. Click **Edit Attributes**.
5. Scroll down to HTML Header.
6. Enter code into HTML Header and click **Apply Changes**.

For example, adding the following would test a function accessible from anywhere on the current page.

```
<script type="text/javascript">
  function test(){
    window.alert('This is a test.');
  }
</script>
```

See Also: "HTML Header" on page 4-21

Including JavaScript in a .js File Referenced by the Page Template

In Oracle Application Express you can reference a `.js` file in the page template. This approach makes all the JavaScript in that file accessible to the application. This is the most efficient approach since a `.js` file loads on the first page view of your application and is then cached by the browser.

The following demonstrates how to include a .js file in the header section of a page template. Note the line `script src=` that appears in bold.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <title>#TITLE#</title>
    #HEAD#
    <script src="http://myserver.myport/my_images/custom.js"
type="text/javascript"></script>
</head>
<body #ONLOAD#>#FORM_OPEN#
```

See Also: "Page Templates" on page 7-28

Calling JavaScript from a Button

Calling a JavaScript from a button is a great way to confirm a request. Oracle Application Express uses this technique for the delete operation of most objects. For example, when you delete a button, a JavaScript message appears asking you to confirm your request. Consider the following example:

```
<script type="text/javascript">
  function deleteConfirm(msg)
  {
var confDel = msg;
if(confDel ==null)
  confDel= confirm("Would you like to perform this delete action?");
else
  confDel= confirm(msg);

if (confDel== true)
  doSubmit('Delete');
  }
</script>
```

This example creates a function to confirm a delete action and then calls that function from a button. Note that the function optionally submits the page and sets the value of the internal variable `:REQUEST` to `Delete`, thus performing the delete using a process that conditionally executes based on the value of request.

Note that when you create the button you would need to select **Action Redirect to URL without submitting page**. Then, you would specify a URL target such as the following:

```
confirmDelete('Would you like to perform this delete action?');
```

See Also: "Creating a Button Using a Wizard" on page 5-48

Creating Dependent Select Lists

You can use a select list to determine the range of values of another select list on the same page. You can achieve this functionality by having a driving select list submit values to a subsequent select list. You incorporate these values in the subsequent select list as a bind variable in the WHERE clause of its query.

You can have one LOV drive another LOV by:

- Creating a basic form.
- Defining two lists of values. Note that the driving LOV must submit the page after a value is chosen.
- Defining a branch that branches back to the current page.

Consider the following example. The first LOV enables the user to pick a state.

```
SELECT state_name d, state_id v
FROM states
```

The second LOV selects the country name and country ID based on the state selected in the first LOV.

```
SELECT county_name d, county_id v
  FROM counties
WHERE state_id = :Px_STATE_ID
```

See Also:

- "Creating Forms" on page 5-26
- "Creating Lists of Values" on page 5-26
- "Controlling Navigation Using Branches" on page 6-8

Creating a Help Page

Application Builder includes built-in attributes to create Help for your application. Creating Help for your application involves the following steps:

- Create a dedicated Help page and Help region
- Define page Help text
- Define item Help text
- Create a navigation bar icon to link to your Help page

Help created in Application Builder displays on a dedicated Help page. To access Help, users click a link that takes them to a dedicated Help page. This Help page displays page and item Help topics specific to the page they are viewing.

Topics in this section include:

- Creating a Help Page and Region
- Defining Help Text
- Creating a Help Navigation Bar Entry

Creating a Help Page and Region

The first step in creating Help for your application is to create a dedicated page and Help Text region.

To create a new Help Text region:

1. Create new page for your Help. See "Adding Pages to an Application" on page 5-8.
2. Navigate to the Page Definition of your Help page. See "Accessing a Page Definition" on page 4-16.
3. Under Regions, the **Create** icon.

4. When prompted to select a region type, select **Help Text**.
5. Follow the on-screen instructions.

Defining Help Text

You define Help text for a page or single item by editing attributes. Ideally, you would define these attributes as you create your application. For simplicity, however, the following procedures describe how to define this text after the fact.

To define page Help text:

1. Navigate to the Page Definition for the page for which you want to add page Help.
2. Click **Edit Attributes** to view the existing page attributes.
3. Scroll down to **Page Help Text**.
4. Enter your Help text in the field provided.
5. Click **Apply Changes**.

Repeat the previous procedure for each page requiring page Help text.

To define item Help text for each page:

1. Navigate to the Page Definition for the page for which you want to add item Help.
2. Under Items, click name of the item you want to edit.
3. Scroll down to **Help Text**.
4. Enter your Help text in the field provided.
5. Click **Apply Change**.

Repeat the previous procedure for each item requiring Help text.

Editing Multiple Item Help Topics at Once

If you are including item Help in your application, you can edit multiple item Help topics at once using the Bulk Edit Item Help report.

Accessing the Bulk Edit Item Help Report To view the Bulk Edit Item Help report:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Tasks list, click **View Application Reports**.
4. Click **Page Components**.
5. Select **Item Help Text**.

   A report displays at the bottom on the page.
6. In Bulk Item Help Report you can:
   - Update existing Help topics. Edit the Help text that appears and click **Apply Changes**.
   - Link to the Page Definition containing the item by clicking the page ID.
   - Link to the Page Item by clicking the item name.

Seeding Item Help Topics If you application does not yet contain item Help, you perform a mass update of default Help text.

To seed item Help topics:

1. Access the Bulk Edit Item Help report.
2. Click **Seed Item Help Text**.
3. In Default Help Text, enter the default text to appear in all Help topics.
4. Click **Apply Changes**.

Searching for Existing Item Help Topics You can search for existing Help text, or for an item label.

To search for existing item Help topic:

1. In Help Contains, enter keywords.
2. Click **Go**.

Searching for an Item Label To search for an item label:

1. In Help Contains, enter keywords.
2. Click **Go**.

> **See Also:** "Viewing Application Reports" on page 4-49

Creating a Help Navigation Bar Entry

Once you have created your Help, the next step is to create a navigation bar entry so users can link to it.

To create a navigation bar entry:

1. Navigate to the Page Definition:
    a. Navigate to the Database Home Page.
    b. Click the **Application Builder** icon.
    c. Select an application.
    d. Select a page.

       The Page Definition appears.
2. Under Navigation Bar, click the **Create** icon.
3. Specify the appropriate navigation bar entry attributes:
    - Sequence
    - Alt Tag Text
    - Icon Image Name
    - Image Height and Image Width
    - Text

   Specify the target location.
4. To specify the target location:
    - From Target type, select **Page in this application**.
    - In Page, specify the page ID.

- In Request, type:

  ```
  &APP_PAGE_ID.
  ```

By specifying substitution string *`&APP_PAGE_ID`* as the Request, you are instructing the Application Express engine to display Help text for the current page when the user clicks this icon.

6 Adding Navigation

When you build an application, you can include different types of navigation controls, including navigation bar entries, tabs, breadcrumbs, lists, and trees. This section describes how to implement navigation in your application.

Navigation controls are shared components. Once you create them, you can add them to any page within your application. You add a specific type of navigation control at the application level on the Shared Components page.

This section contains the following topics:

See Also:

- "About the Shared Components Page" on page 4-40
- "About the Page Definition" on page 4-16
- "Controlling Page Layout and User Interface" on page 7-1

Creating a Navigation Bar Entry

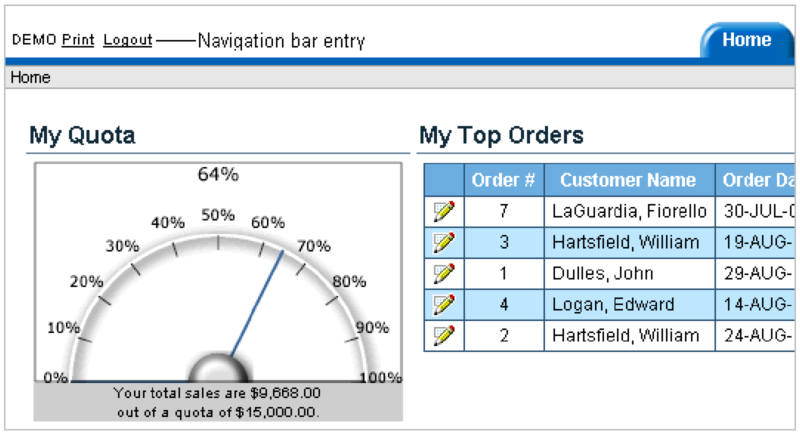
Navigation bar entries offer an easy way to move users between pages in an application. The location of a navigation bar depends upon the associated page template. A navigation bar entry enables you to display a link from an image or text.

Topics in this section include:

See Also: Customizing Templates on page 7-17

About Navigation Bars

A navigation bar entry can be an image, an image with text beneath it, or text. You must supply navigation bar entry images and text. When you create a navigation bar entry, you can specify an image, text, a display sequence, or a URL.



Your total sales are $9,668.00 out of a quota of $15,000.00.

	Order #	Customer Name	Order Da
✎	7	LaGuardia, Fiorello	30-JUL-
✎	3	Hartsfield, William	19-AUG-
✎	1	Dulles, John	29-AUG-
✎	4	Logan, Edward	14-AUG-
✎	2	Hartsfield, William	24-AUG-

Navigation bars are different from other shared components in that you do not need to reference them on a page by page basis. If your page template includes the `#NAVIGATION_BAR#` substitution string, the Application Express engine automatically includes any defined navigation bars when it renders the page.

> **See Also:** "Supported Page Template Substitution Strings" on page 7-28 on using the `#NAVIGATION_BAR#` substitution string

How to Create a Navigation Bar Entry

Before you can add a navigation bar, you must create a navigation bar entry on the Navigation Bar page. You can access the Navigation Bar page from either the Page Definition or from the Shared Components page.

> **See Also:** "About the Shared Components Page" on page 4-40

Creating a Navigation Bar Entry Referencing an Icon

To create a navigation bar entry referencing an icon:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Shared Components, scroll down to **Navigation Bar**.
3. Under Navigation Bar, click the **Create** icon.

   The Create Navigation Bar Entry Wizard appears.
4. Specify the following Navigation Bar Entry attributes:
   a. Sequence - Specify the order of evaluation for this component.

   b. Alt Tag Text - Enter ALT text for navigation icons that are images. If you do not specify an image name, then this text displays.
   c. Icon Image Name - Defines the name of the image that displays.
   d. Image Height - Defines the height of the image.
   e. Image Width - Defines the width of the image.
   f. Text - Enter additional text to display with the image. You can include text or use icons with no text. This attribute is optional and can be translated.
5. Specify the target location.
   a. If the target location is a URL:
      – From Target is a, select **URL**.
      – In URL Target, enter a URL.
   b. If the target location is a page:
      – From Target is a, select **Page in this Application**.
      – In Page, specify the page ID.
6. If the navigation bar entry will display conditionally, specify the appropriate conditional information and click **Create**.

Creating a Navigation Bar Entry without an Icon

To create a navigation bar entry without icons:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Shared Components, scroll down to **Navigation Bar**.
3. Under Navigation Bar, click the **Create** icon.

   The Create Navigation Bar Entry Wizard appears.
4. Specify the following icon attributes:
   a. Sequence - Specify the order of evaluation for this component.
   b. Text - Enter additional text to display with the image. You can include text or use icons with no text. This attribute is optional and can be translated.
5. Specify the target location.
   a. If the target location is a URL:
      – From Target is a, select **URL**.
      – In URL Target, type a URL. For example:

        ```
        http://www.yahoo.com
        ```
   b. If the target location is a page:
      – From Target is a, select **Page in this Application**.

   - In Page, specify the page ID.
6. If the navigation bar entry will display conditionally, specify the appropriate conditional information and click Create.

Editing a Navigation Bar Entry

Once you create a navigation bar entry you can edit it on the Navigation Bar Entries page.

To edit a navigation bar entry:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Applications home page, click **Shared Components**.
4. Under Navigation, select **Navigation Bar Entries**.

   The Navigation Bar Entries page appears.
5. You can change the appearance of the page by making a selection from the View list and clicking **Go**. Available options include:
   - **Icons** (the default) displays each navigation bar entry as a large icon. To edit a navigation bar entry, click the icon.
   - **Details** displays each navigation bar as a line in a report. To edit a navigation bar, click the appropriate sequence number.

     The Edit Navigation Bar Entry page appears.
6. Make the appropriate edits and click **Apply Changes**.

Editing Multiple Navigation Bar Entries Simultaneously

To edit multiple navigation bar entries at once:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Applications home page, click **Shared Components**.
4. Under Navigation, select **Navigation Bar Entries**.

   The Navigation Bar Entries page appears.
5. C lick **Grid Edit**.
6. Edit the appropriate attributes and click **Apply Changes**.

Accessing Navigation Bar Entry Reports

You can view the Navigation Bar Entry Subscription and Navigation Bar Entry History reports by clicking the appropriate tab at the top of the Navigation Bar Entries page.

Note: The Subscription and History buttons only appear after you create a navigation bar.

Navigation Bar Entry Subscription Report

Click **Subscription** to access the Subscribed NavBars report. This report displays subscribed navigation bar entries in your application.

Navigation Bar Entry History

Click **History** to view the Navigation Bar History report. This report lists recent changes to navigation bars.

Creating Tabs

Tabs are an effective way to navigate users between pages of an application. You can create a tabbed application look by using parent tabs, standard tabs, and lists.

Application Builder includes two different types of tabs:

- Standard tabs
- Parent tabs

An application having only one level of tabs uses a standard tab set. A standard tab set is associated with a specific page and page ID. You can use standard tabs to link users to a specific page. A parent tab set functions as a container to hold a group of standard tabs. Parent tabs give users another level of navigation as well as a context (or sense of place) within the application. You can use parent tabs to link users to a specific URL associated with a specific page.

Topics in this section include:

- About Template Support
- About the Tabs Page
- Editing Multiple Tabs at Simultaneously
- Accessing Tab Reports

Note: When running the Create Application Wizard, you have the option of creating an application with tabs. The following procedures assume you have already created an application that does not have any tabs.

See Also: "Creating an Application" on page 5-1

About Template Support

Before you can create parent and standard tabs, you need to check that your default template has positions defined for both standard and parent tabs using the appropriate substitution strings. You also need to make sure you do not override this template at the page-level.

See Also:

- "Template Defaults" on page 4-10 for information about setting a default page template at the application level
- "Display Attributes" on page 4-21 for information about setting a template at the page-level

About the Tabs Page

The Tabs page displays a graphical representation of the tabs defined in your application. You access the Tabs page from the Shared Components page, or by clicking the heading Tabs on the Page Definition.

Topics in this section include:

- Accessing the Tabs Page from Shared Components
- Accessing the Tabs Page from a Page Definition
- Creating a New Tab from the Page Definition
- Using the Standard Tab Task List

Accessing the Tabs Page from Shared Components

To access the Tabs page from the Shared Components page:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application Builder home page, click **Shared Components**.
4. Under Navigation, select Tabs.

   The Tabs page appears displaying a graphical representation of the tabs defined in your application.
5. To make another tab current, click the tab.

   Notice the two Add buttons. Use the Add button on the upper right side of the image to add Parent tabs. Use the Add button at the lower left side of the image to add standard tabs.
6. To add a new tab, click Add adjacent to the appropriate tab type.

   Think of parent tabs as a container to hold standard tabs. For example, to add two levels of tabs, you first create a parent tab and then add standard tabs to it.

Accessing the Tabs Page from a Page Definition

To access Tab Manager from the Page Definition:

1. Navigate to the appropriate Page Definition:

   a. Navigate to the Database Home Page.

   b. Click the **Application Builder** icon.

   c. Select an application.

   d. Select a page.

      The Page Definition appears.
2. Under Shared Components, select the heading **Tabs**.

   The Tabs page appears displaying a graphical representation of the tabs defined in your application. The currently selected standard or parent tab is highlighted.
3. To make another tab current, click the tab.

   Notice the two Add buttons. Use the Add button on the upper right side of the image to add Parent tabs. Use the Add button at the lower left side of the image to add standard tabs.

4. To add a new tab, click Add adjacent to the appropriate tab type.

   Think of parent tabs as a container to hold standard tabs. For example, in order to add two levels of tabs you first create a parent tab and then add standard tabs to it.

Creating a New Tab from the Page Definition

To create a new tab from the Page Definition:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Tabs, click the **Create** icon.

   The Create Tab Wizard appears.
3. Follow the on-screen instructions.

Using the Standard Tab Task List

The Standard Tab Task list displays on the right side of the Tabs page. You can access the links on this list to rename a standard tab set, resequence the display order, associate pages with a tab set, create a new standard tab, or create a new standard tab set.

To access the Standard Tab Task list:

1. Navigate to the Tabs page:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under Navigation, select Tabs.
2. Make a selection from the Standard Tab Task list. Available options include:
   - **Rename Standard Tab Set**
   - **Resequence Display Order**
   - **Associate Page(s) with Selected Standard Tab**
   - **Create New Standard Tab**
   - **Create New Standard Tab Set**

Editing Multiple Tabs at Simultaneously

You can edit multiple tabs at once by clicking **Edit Standard Tabs** and **Edit Parent Tabs** on the Tabs page.

To edit multiple tabs at once:

1. Navigate to Tab Manager. See "About the Tabs Page" on page 6-6.
2. Click one of the following buttons:

- **Edit Standard Tabs**
- **Edit Parent Tabs**

Accessing Tab Reports

You can view the Tab Utilization and Tab History reports by clicking the appropriate tab at the top of the Tab Manager page.

Standard Tab Utilization

Click **Utilization** to access the Standard Tab Utilization report. This report lists the standard tabs used in the current application.

Standard and Parent Tab History

Click **History** to view the Standard Tab History and Parent Tab History reports. These reports display a history of changes to tab attributes for the current application.

Controlling Navigation Using Branches

A branch is an instruction to link to a specific page, procedure, or URL. For example, you can branch from page 1 to page 2 after page 1 is submitted.

Note: If a page has a select list and a submit button, it can submit itself. However, you must create a branch to call the page or the submit will fail.

You create a new branch by running the Create Branch Wizard and specifying a the branch point and branch type. The branch type defines the type of branch you are creating.

To create a branch:

1. Navigate to the appropriate Page Definition:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Select a page.

      The Page Definition appears.
2. Under Branches, click the **Create** icon.
3. Select a branch point:
   - On Submit: Before Computation - Occurs before computations, validations, or processing. Use this option for a Cancel button.
   - On Submit: Before Validation - Occurs after computations, but before validations or processing. Typically not used. If a validation fails, page processing stops, a rollback is issued, and the page displays the error. Because of this default behavior, you do not need to create branches to accommodate validations. However, you might want to branch based on the result of a computation (for example, to the previous branch point).
   - On Submit: Before Processing - Occurs after computations and validations, but before processing. Use this option to branch based on a validated session state, but before performing any page processing.

- On Submit: After Processing - Occurs after computations, validations, and processing. This option branches to a URL or page after performing computations, validations, and processing. When using this option, remember to sequence your branches if you have multiple branches for a given branch point.
- On Load: Before Header - Occurs before a page is rendered. This option displays another page, instead of the current page, or redirects the user to another URL or procedure.

4. Select a branch type.

   Depending upon the branch type, specify the following types of information on the pages that follow:

   - The page ID of the page to which you want to branch
   - PL/SQL code
   - A URL address

5. Follow the on-screen instructions.

Creating Breadcrumbs

Breadcrumbs provide users with hierarchical navigation. A breadcrumb is a hierarchical list of links that display using templates. You can display a breadcrumb as a list of links or as a breadcrumb path.

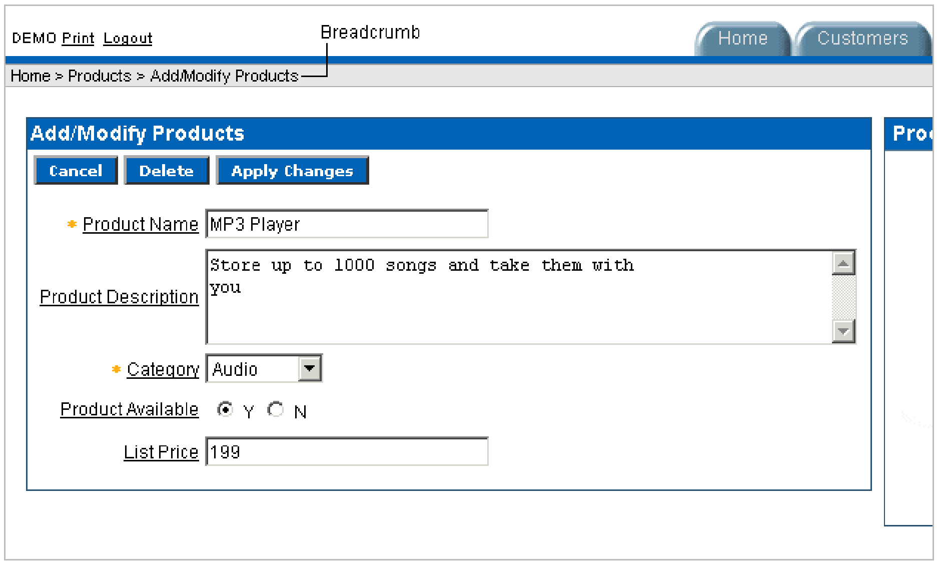
Topics in this section include:

- About Breadcrumbs
- How to Create Breadcrumbs
- Editing Breadcrumbs
- Accessing Breadcrumb Reports

See Also: "Creating a New Template" on page 7-19 and "Breadcrumb Templates" on page 7-21

About Breadcrumbs

A breadcrumb trail indicates where the user is within the application from a hierarchical perspective. In addition, users can click a specific page to instantly view it. You can include a breadcrumb that functions as second level of navigation and displays beneath the standard tabs at the top of each page.



* Product Name	MP3 Player
Product Description	Store up to 1000 songs and take them with you
* Category	Audio
Product Available	[✓ Y] N
List Price	199

How to Create Breadcrumbs

To add a breadcrumb to a page in your application you must:

- Create the breadcrumb by running the Create Breadcrumb Wizard.
- Add entries to it.
- Add the breadcrumb to a page by creating a region.

Creating Breadcrumbs from the Shared Components Page

To create breadcrumbs from the Shared Components page:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application Builder home page, click **Shared Components**.
4. Under Navigation, select **Breadcrumbs**.

   The Breadcrumbs page appears.
5. Click Create.
6. Enter a name and click Create.

Creating Breadcrumbs from a Page Definition

To create breadcrumbs from a Page Definition:

1. Navigate to the appropriate Page Definition:

   a. Navigate to the Database Home Page.

   b. Click the **Application Builder** icon.

   c. Select an application.

   d. Select a page.

   The Page Definition appears.
2. Under Shared Components, scroll down to **Breadcrumbs** and click the **Create** icon.

3. Enter a name and click Create.

After you create a breadcrumb, you add entries to it.

Adding Breadcrumb Entries

To add a breadcrumb entry:

1. Navigate to the Breadcrumbs page:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. On the Application Builder home page, click **Shared Components**.
   e. Under Navigation, select **Breadcrumbs**.

      The Breadcrumbs page appears.
2. Select a breadcrumb to add entries to.
3. Click **Create Breadcrumb Entry**.
4. Under Breadcrumb Identification, specify the page on which this menu will be current
5. Under Entry:
   a. Sequence - Indicate the order in which breadcrumb entries appear.
   b. Parent Bread Entry - Identify the parent of this entry.
   c. Short Name - Specify the short name of this entry (referenced in the breadcrumb template).
   d. Long Name - Specify the long name of this entry (referenced in the breadcrumb template).
6. Under Target:

   If the target location is a URL:
   a. From Target is a, select **URL**.
   b. In URL Target, enter a URL.

   If the target location is a page:
   a. From Target is a, select **Page in this Application**
   b. In Page, specify the page ID
7. You can make a breadcrumb conditional by making selections under Conditions.

   To make the breadcrumb conditional:
   a. Make a selection from the Condition Type list.
   b. Enter an expression in the fields provided.
8. When you are finished defining menu attributes, click **Create** at the top of the page.

Repeat these procedures for each breadcrumb entry you need to create.

Adding a Breadcrumb Region

A region is a area on a page that serves as a container for content. Once you create a breadcrumb and a breadcrumb template, the next step is to create a region. Once you create a region you can add a breadcrumb to page.

> **See Also:** "Creating a New Template" on page 7-19 and "Breadcrumb Templates" on page 7-21 for information about changing menu display

To create a breadcrumb region:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Regions, click the **Create** icon.

   The Create Region Wizard appears.
3. For the region type, select Breadcrumb.
4. Enter a title and select a region template.
5. Select a breadcrumb and a breadcrumb template.
6. Optional. Identify the breadcrumb entry used to identify this page:
   a. Breadcrumb Entry Label - Enter a label for the breadcrumb entry.
   b. Parent Breadcrumb Entry - Select the appropriate hierarchical parent.
7. Click Finish.

Repeat these procedures for each page where you would like to add breadcrumb navigation.

About Creating a Dynamic Breadcrumbs

To give users more exact context, you can include session state in breadcrumbs, making your breadcrumbs dynamic. For example, suppose a page in your application displays a list of orders for a particular company and you want to include the following breadcrumb:

```
Home > Orders > Orders for ACME Inc
```

In this example, `ACME Inc` not only indicates the page a user is on and the navigation path. The Application Express engine stores the value of ACME Inc. in session state.

To create this type of dynamic menu, you must include a reference to a session state item in the breadcrumb's short name or long name, for example:

```
&COMPANY_NAME.
```

Editing Breadcrumbs

Once you create a breadcrumb you can edit it on the Breadcrumbs page.

To edit a breadcrumb:

1. Navigate to the Database Home Page.
2. Click the **Application Builder** icon.
3. Select an application.
4. On the Application Builder home page, click **Shared Components**.
5. Under Navigation, select **Breadcrumbs**.

   The Breadcrumbs page appears.

   You can change the appearance of the page by making a selection from the View list. Available options include:

   - **Icons** (the default) displays each breadcrumb as a large icon. To edit a breadcrumb, click the icon.
   - **Details** displays each breadcrumb as a line in a report. To edit a breadcrumb, select the appropriate name.

Accessing Breadcrumb Reports

You can view the Breadcrumb Utilization and Breadcrumb History reports by clicking the appropriate tab at the top of the Breadcrumbs page.

Note: The Utilization and History buttons only appear after you create a breadcrumb.

Breadcrumb Utilization Report

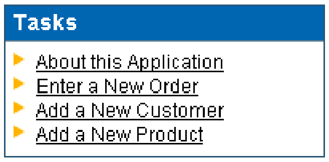
Click **Utilization** to access the Breadcrumb Utilization report. This report lists breadcrumbs by page. Click the page ID to link to a specific page.

Breadcrumb History Report

Click **History** to view the Breadcrumb History report. This report lists recent changes to breadcrumbs.

Creating Lists

A list is a shared collection of links. You control the appearance of a list through list templates. Each list element has a display condition which enables you to control when it displays. You can define a list element to be either current or non current for a specific page. You further specify what current looks like using template attributes. You add a list to a page by creating a region and specifying the region type as List.



Topics in this section include:

- Creating a List
- Adding a List to a Page
- Editing Multiple List Entries Simultaneously

- Accessing List Reports

  See Also: "Creating a New Template" on page 7-19 and "List Templates" on page 7-26 for information about altering list display

Creating a List

To add a list to a page in your application you must:

1. Create the list by running the Create Lists Wizard.
2. Add items to the list.
3. Add the list to a page by creating a List region.

Creating a List from the Shared Components Page

To create a list from the Shared Components page:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Applications home page, click **Shared Components**.
4. Under Navigation, select **Lists**.

   The Lists page appears.
5. To create a new list, click **Create**.
6. In the fields provided:
   a. Enter a name for the list.
   b. Select a list template.
   c. If applicable, select a build option for this component. Build options are predefined settings that determine whether or not components within an application are enabled.
7. Click **Create**.

Creating a List from a Page Definition

To create a list from a Page Definition:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Shared Components, scroll down to Lists and click the **Create** icon.
3. In the fields provided:
   a. Enter a name for the list.
   b. Select a list template.

   c. If applicable, select a build option for this component. Build options are predefined settings that determine whether or not components within an application are enabled.

4. Click **Create**.

Once your list is created, you need to add entries to it.

Adding List Entries

You can create hierarchical lists that contain sublists. To create a hierarchical list, you must:

- Select a list template that supports hierarchical lists. To determine which list templates support hierarchical lists, look for templates having the naming convention "with Sublist."
- Select a Parent List Item when you create each list entry.

See Also: "Viewing Templates" on page 7-19 and "List Templates" on page 7-26

To add an entry to a list:

1. Navigate to the Lists page.
2. Select a list.
3. Click **Create List Entry**.

   The Create / Edit List Entry page appears.
4. Under Label and Sequence:

   a. Parent List Item - Identify the parent for this list entry. Use this attribute if you are creating a hierarchical list that will contain a sublist.

   b. Sequence - Indicate the order in which list entries appear.

   c. Image - Identify the file name for the image used to display this list entry. Control over this attribute is provided by list templates.

   d. Image Attributes - Identify the image attributes (such as width="12" height="12") for the list element image.

   Use the `#LIST_LABEL#` substitution string to reference the list label text. This substitution string allows for the title image attribute to be automatically set based on the value of the list label text. For example:

   ```
   title="#LIST_LABEL#"
   ```

   e. List Entry Label (required) - Enter the label text for this link (required).
5. Specify a target location.

   If the target location is a page:

   a. From Target Type, select **Page in this Application**.

   b. In Page, specify the target page ID.

   To reset pagination for this page, select **reset pagination for this page**.

   c. In Request, specify the request to be used.

   d. In Clear Cache, specify the pages (that is, the page IDs) on which to clear cache. Specify multiple pages by listing the page IDs in a comma-delimited list.

      You can set session state (that is, give a listed item a value) using the next two attributes:

   e. To set session state:

      – In Set these items, enter a comma-delimited list of item names for which you would like to set session state.

      – In With these values, enter a comma-delimited list of values for the items specified in the previous step.

        You can specify static values, or substitution syntax (for example, `&APP_ITEM_NAME.`). Note that item values passed to `f?p=` in the URL cannot contain a colon. Additionally, item values cannot contain commas unless you enclose the entire value in backslashes (for example, `\1234,56\`).

   If the target location is a URL:

   a. From Target type, select **URL**.

   b. In URL Target, enter a URL.

6. Under Current List Entry Identification:

   a. List Entry Current for Pages Type - Specify when this list entry should be current based on the page type.

      List items can be current or non-current. Current list items use the current template, non current list items use the non current list item template. The actual condition and templates are defined in subsequent attributes.

   b. List Entry Current for Condition - Based on the selection above, define a condition to evaluate. When this condition is true then the list item becomes current.

7. To make the list entry conditional:

   a. Make a selection from the Condition Type list.

   b. Enter an expression in the fields provided.

8. When you are finished defining list attributes, click Create or Create and Create Another.

Adding a List to a Page

Once you create a list, the next step is to add it a page by creating a region and specifying the region type as List.

> **See Also:** "Creating a New Template" on page 7-19 and "List Templates" on page 7-26 for information about altering list display

To add a list to a page:

1. Navigate to the appropriate Page Definition:

   a. Navigate to the Database Home Page.

   b. Click the **Application Builder** icon.

   c. Select an application.

   d. Select a page.

      The Page Definition appears.

2. Under Regions, click the Create icon.
3. Select List as the region type.
4. Specify the following display attributes:
   - Title - Enter a title for the region. This title will display if the region template you choose displays the region title.
   - Region Template - Choose a template to control the look of the region.
   - Display Point - Identify a display point for this region.

     Two types of display points exist: page template positions and page body positions. **Page template positions** are controlled by page template substitution strings (#REGION_POSITION_01#..#REGION_POSITION_08#). **Page template positions** allow for exact placement of a region within a page template. Page body positions are displayed where the #BODY# substitution string in the page template indicates.
   - Sequence - Specify the sequence for this component. The sequence determines the order of evaluation.
   - Column - Indicate the column in which this region is to be displayed. A page can have multiple regions, these regions can be displayed in different columns. Please note that this attribute only applies to regions that are displayed in a Page Template Body position.
5. From List, select the list you want to add.
6. Click Create List Region.

Repeat these procedures for each page on which you would like to add a list.

Editing Lists

Once you create a list you can edit it on the Lists page.

To edit a list:

1. Navigate to the Lists page.

   From the Page Definition:

   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.
   e. Under Shared Components, select the title **Lists**.

   From the Shared Components page:

   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. On the Applications home page, click **Shared Components**.
   e. Under Navigation, select **Lists**.

The Lists page appears.

2. You can change the appearance of the page by making a selection from the View list. Available options include:
   - **Icons** (the default) displays each list as a large icon. To edit a list, click the appropriate icon.
   - **Details** displays each list as a line in a report. To edit a list, click the list name.

Editing Multiple List Entries Simultaneously

You can edit multiple list entries at once by clicking **Grid Edit** on the List Entries page.

To edit multiple list entries at once:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application Builder home page, click **Shared Components**.
4. Under Navigation, select **Lists**.

   The Lists page appears.
5. Select a list name.

   The List Entries page appears.
6. Click **Grid Edit**.
7. Edit the appropriate items and click **Apply Changes**.

Accessing List Reports

You can view the List Utilization by Page, Unused Lists, and List History reports by clicking the appropriate tab at the top of the Lists page.

Note: The List Utilization, Unused Lists, and History buttons only appear after you create a list.

Utilization

Click **List Utilization** to access the Lists Utilization report. This report displays all lists included in the current application. To edit list entries, select the list name. To view the pages on which the list appears, click the number in the Pages column.

Unused

Click **Unused** to identify lists that are not used in the current application.

History

Click **History** to view changes to list definitions and list entries by developer and date.

Creating Trees

You can use a tree in your application to effectively communicate hierarchical or multiple level data.

Topics in this section include:

Creating a Tree

To create a tree:

1. Navigate to the Shared Components page:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. On the Application Builder home page, click **Shared Components**.
2. Under Navigation, select **Trees**.

   The Trees page appears.
3. To create a new tree, click **Create**.
4. Enter basic page information.
5. Specify how tabs should be implemented.
6. Enter a Tree Name and specify number of default expanded levels.
7. Select a tree template.
8. To display a tree, you need to specify a starting point. Depending on your Start Tree selection, enter either a query or a single value.
9. Identify whether to include Collapse All, Expand All, or Reset Tree buttons.
10. Specify the owner and name of the table on which the tree will be based.
11. A tree is based on a query and returns data that can be represented in a hierarchy. Identify the column you want to use as the ID, the Parent ID, and specify the text that should appear on the leaf nodes.
    a. I D - Enter the column you want to use as the ID.
    b. Parent ID - Enter the Parent ID.
    c. Leaf Node Text - Specify the text that should appear on the leaf nodes.
    d. Link Option - Select **Existing Application Item** to make the leaf node text a link. If you select this option, specify a page to link to.
12. Optional. Identify an optional where and order by clause to add to your query.
13. Specify the display text for the Go button.
14. Identify the page you want to branch to when the user clicks a button.
15. Click **Finish**.

Editing a Tree

Once you create a tree you can edit it on the Trees page.

To edit a tree:

1. On the Database Home Page, click the **Application Builder** icon.

2. Select an application.
3. On the Applications home page, click **Shared Components**.
4. Under Navigation, select **Trees**.

   The Trees page appears.
5. You can change the appearance of the page by making a selection from the View list. Available options include:
   - **Icons** (the default) displays each tree as a large icon. To edit a tree, click the icon.
   - **Details** displays each tree as a line in a report. To edit a tree, click the **appropriate name**.

Accessing Tree Reports

You can view the Trees Utilization and Tree History reports by clicking the appropriate tab at the top of the Trees page.

Note: The Utilization and History buttons only appear after you have created a tree.

Tree Utilization

Click **Utilization** to access the Tree Utilization report. This report displays all trees included in the current application. To edit a tree, select the tree name.

Tree History

Click **History** to view changes to trees by developer and date.

7 Controlling Page Layout and User Interface

This section describes different ways you can customize your application’s user interface and page layout including customizing regions, editing item attributes, customizing templates, and incorporating cascading style sheets and images.

This section contains the following topics:

See Also: "Adding Navigation" on page 6-1 for information about creating navigation bars, tabs, breadcrumbs, lists, and trees

Understanding Page Layout

The Application Express engine renders pages by combining templates with application components defined by the developer and data in the database.

The overall framework (or structure of a page) is determined by the page template. For example, the page templates controls if a page uses tabs and a navigation bar. It can also define if a page includes a bar on the left side that serves as a placeholder for navigation or secondary content. Finally, a page template can include definitions of region positions, which enable precise control over placement of regions using HTML tables or style sheet definitions. The page template itself is composed of HTML combined with substitution strings which will be substituted with the appropriate components at run time.

As a developer, you add content on a page by creating a region. A region is an area of a page that serves as a container for content. Each region contains a different type of content such as HTML, a report, a form, a chart, a list, a breadcumb, PL/SQL, a tree, a URL, or a calendar. You position a region either relative to other regions (that is, based on its sequence number and column), or by using a region position defined in the page template. The style of the region is also controlled by the region template. Like the page template, the region template defines the structure of the area that the region takes up on a page. It defines if the region title is displayed and where it is displayed relative to the main content, or the body. A region can also define absolute positions for buttons.

See Also: "Creating a Region" on page 7-2

Displaying Components on Every Page of an Application

Page zero of your application functions as a master page. The Application Express engine renders all components you add to page zero on every page within your application. You can further control whether or not the Application Express engine renders a component or runs a computation, validation, or process by defining conditions.

To create a page zero:

1. Create a new page.
2. Specify the page ID as zero (0).

See Also: "Adding Pages to an Application" on page 5-8, "Understanding Conditional Rendering and Processing" on page 3-2, and "Available Conditions" on page A-1

Customizing Regions

A region is a area on a page that serves as a container for content. Each page can have any number of regions. You control the appearance of a region through a specific region template. The region template controls the look of the region, the size, determines whether or not there will be a border or a background color, and what type of fonts display. A region template also determines the standard placement for any buttons placed in region positions.

You can use regions to group page controls (such as items or buttons). You can create simple regions that do not generate additional HTML, or create elaborate regions that frame content within HTML tables or images.

Regions display in sequence within HTML table columns. You can also explicitly place regions in positions defined in the page template. You can also choose to display regions conditionally.

Topics in this section include:

- Creating a Region
- How Region Attributes Affect Page Layout

Creating a Region

You create new regions by running the Create Region Wizard.

To create a new region:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Regions, click the **Create** icon.

   The Create Region Wizard appears.
3. Select a region type and follow the on-screen instructions.

When you create a region you select a region type. The Application Express engine interprets a region differently based on the type you select. Table 7–1 describes the available region types.

Table 7–1 Region Types

Region Type	Description
HTML	When you select HTML, the wizard prompts you to select one of the following: ▪ **HTML** - Functions as containers for items and contain the HTML you provide. Any HTML you type may contain substitution strings. ▪ **HTML Text (escape special characters)** - Same as HTML region, but the Application Express engine escapes special characters before they are rendered. ▪ **HTML Text (with shortcuts)** - Same as HTML region, but with support for shortcuts. **See Also:** "Using Shortcuts" on page 5-63
Report	Report regions can be defined by a SQL query you write, or by using a wizard to guide you through the steps needed to write a query. See Also: "Creating Reports" on page 5-17
Form	Form regions are used to contain a form. See Also: "Creating Forms" on page 5-26
Chart	Chart regions contain line, bar, or pie charts based on SQL queries. See Also: "Creating Charts" on page 5-39
List	List regions contain a shared collection of links called list. See Also: "Creating Lists" on page 6-13
Breadcrumb	Breadcrumb regions contain a hierarchical list of links called a breadcrumb. See Also: "Creating Breadcrumbs" on page 6-9
PL/SQL Dynamic Content	Regions based on PL/SQL enable you to render any HTML or text using the PL/SQL Web Toolkit.
Tree	Trees are a hierarchical navigational control based on a SQL query executed at run time. It enables the user to expand an collapse nodes. **See Also:** "Creating Trees" on page 6-18

Table 7–1 (Cont.) Region Types

Region Type	Description
URL	URL based regions obtain their content by calling a Web server using a predefined URL. **See Also:** "Incorporating Content from Other Web Sites" on page 7-8
Calendar	Calendar regions are used to contain a monthly calendar. See Also: "Creating Calendars" on page 5-34
Multiple HTML	Use this option to create multiple HTML regions at once. In the fields provided, specify the Sequence, Title, Display Point, Report Template, and Column for each region.
Help Text	Help Text regions enable you to provide page-level help. **See Also:** "Creating a Help Page" on page 5-68

See Also:

- *Oracle Database Application Developer's Guide - Fundamentals* for information about developing Web applications with PL/SQL
- *Oracle Database PL/SQL Packages and Types Reference* for information about htp packages

How Region Attributes Affect Page Layout

You can alter the appearance of a page by changing the region attributes.

To edit region attributes:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Regions, select the region name.

   The Region Definition appears.

Table 7–2 describes region attributes that affect the layout of a page.

Table 7–2 Region Attributes Affecting Page Layout

Attribute	Description
Conditions	Defines conditions and appropriate expressions that determine if the region displays. Conditions can reference session state, the currently logged in user, or environment preferences (such as whether or not a page is in Print View mode). **See Also:** "Understanding Conditional Rendering and Processing" on page 3-2 and "Optimizing a Page for Printing" on page 7-42
Header and Footer	Specifies HTML text to be displayed at the top of the region (just before the `#BODY#` content).
Customization	Enables end user customization. To utilize this feature, you must include the `#CUSTOMIZE#` substitution string in the Header, Body, or Footer section of the page template. **See Also:** "Enabling Users to Customize a Page" on page 7-17
User Interface, Column	Determines the column in which the region displays. If two regions are in the same display point, you can place them next to one another by setting the second region to display in column 2. Many regions can display in each column and the display order of the regions within the region display point and column is controlled by the region display sequence number.
User Interface, Template	Determines the look of the region. Select from the region templates defined in the application. To view template attributes, click the template name on the Page Definition. **See Also:** "Customizing Templates" on page 7-17 and "Region Templates" on page 7-34
User Interface, Sequence	Specifies the display order of the regions within the page.
User Interface, Display Point	Identifies where within the page the region displays. Regions are rendered in order of sequence number within a Display Point. Click the View icon to see the page layout and select a position. The possible display points for a region are determined by the page-level template (which is a page attribute). If no page-level template is selected, the default page-level template, defined in the Application Definition is used.
User Interface, Region HTML table cell attributes	Defines additional attributes to be used in the HTML table cells when regions display in multiple columns. The attributes control the cells in the table used to lay out a region in multiple columns.

Controlling Region Positioning

When you create a region, you must specify its position (or Display Point) on the page. You can choose either a default position (such as Page Template Body) or a user-defined position in the template (such as Page Template Region Position 1.)

In addition to Display Point, you can specify the column in which the region will be placed. When you place regions in multiple columns, Oracle Application Express automatically renders the necessary HTML to produce a multiple column layout.

Specifying a Region Header and Footer

In addition to the body content of a region (which can be a report, a chart, or HTML with form elements), you can specify additional HTML to be placed above and below a region or in its header and footer. The region footer supports the following substitution strings:

- `#TIMING#` shows the elapsed time in seconds used when rendering a region. You cab use this substitution string for debugging purposes.

- #ROWS_FETCHED# shows the number of rows fetched by the Oracle Application Express reporting engine (the page size). You can use these substitution strings to display customized messages to the user. For example:

  ```
  Fetched #ROWS_FETCHED# rows in #TIMING# seconds.
  ```

- #TOTAL_ROWS# displays the total number of rows that satisfy a SQL query used for a report.
- #FIRST_ROW_FETCHED# and #LAST_ROW_FETCHED# displays the range of rows displayed. For example:

  ```
  Row(s) #FIRST_ROW_FETCHED# through #LAST_ROW_FETCHED# of #ROWS_FETCHED#
  displayed
  ```

Enabling Users to Customize a Page

You can use the Customization attribute to enable users to turn regions on and off in a running application.

To enable end user customization:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Regions, click the region name.

   The Region Definition appears.
3. Scroll down to Customization and select one of the following:
   - Customizable and Not Shown By Default
   - Customizable and Shown By Default
4. In Customized Option Name, enter the label that represents this region on the page to the user.
5. Include the #CUSTOMIZE# substitution string in the Header, Body, or Footer section of the page template.

To utilize this feature, you must include the #CUSTOMIZE# substitution string in the Header, Body, or Footer section of the page template.

If at least one region supports end user customization, a link called Customize appears wherever you include the #CUSTOMIZE# substitution string in the page template. When users click this link, a window displays enabling them to turn on and off regions on the page.

See Also: "Customizing Templates" on page 7-17

How Item Attributes Affect Page Layout

An item is part of an HTML form and can be a text field, text area, password, select list, check box, and so on. You can alter the appearance of a page by changing the item attributes. For example, these attributes can effect where a label displays, how large an item will be, if the item will display next to or below the previous item.

See Also: "Creating Items" on page 5-51

To edit item attributes:

1. Navigate to the Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Items, click the item name.

   The Item Definition appears.

Table 7–3 describes how item attributes affect the layout of a page.

Table 7–3 Item Attributes Effecting Page Layout

Heading	Attribute	Description
Displayed	Sequence	Determines the order in which items are rendered within a region.
Displayed	Region	Defines the region in which the item displays. All items must be in a region.
Displayed	Begin On New Line	Determines if this item displays on the same line as the previous item or if it displays on the next line.
Displayed	Begin On New Field	Determines if this item displays in the next column or in the same column as the previous item.
Displayed	ColSpan	Items are laid out in HTML tables. Defines the value to be used for the COLSPAN attribute of the table cell containing an item.
Displayed	RowSpan	Items are laid out in HTML tables. Defines the value to be used for the ROWSPAN attribute in the table cell in which that the item displays.
Label	Label	Enter the label for this item. You can include HTML, JavaScript, and shortcuts. You can also use the substitution string `#CURRENT_ITEM_NAME#` to obtain the name of the item associated with this label.
Label	Horizontal/Vertical Alignment	Controls the placement as well as the horizontal and vertical alignment of the label. Labels can be displayed above, below, or to the left of the item.
Label	Template	Specifies the label template. Use label templates to apply a consistent appearance to labels in your application.
Label	HTML Table Cell Attributes	Defines additional attributes for the cell containing this item's label (for example, `nowrap="nowrap"`).
Label	Post Element Texts	Specifies additional attributes for the HTML table cell used to display each individual option in a radio group or set of check boxes. Can include HTML, JavaScript, and shortcuts. You can reference the following substitution strings: ▪ `#CURRENT_FORM_ELEMENT#` obtains the name of the HTML form element with which this post element text is associated. ▪ `#CURRENT_ITEM_NAME#` obtains the name of the item with which this post element text is associated.

Table 7–3 (Cont.) Item Attributes Effecting Page Layout

Heading	Attribute	Description
List of Values	Columns	Applies to radio groups and check boxes. Defines the number of columns to use to display the values defined in the List of Values. By default, all values display in one column.
Conditions	Condition Type and Expressions	Defines conditions and appropriate expressions that determine if an item displays. **See Also:** "Understanding Conditional Rendering and Processing" on page 3-2
Read Only Display Settings	Read Only Condition Type	Defines conditions and expressions that determine if the item will display as read-only. Use this attribute to display certain items to a set of users as updatable, while displaying that same set of items to others users as nonupdatable. Reduces the need to code duplicate interfaces for different users.

Incorporating Content from Other Web Sites

Typically, pages in an application are based on data stored in an Oracle database. To incorporate content from other servers, you can create a region based on a URL to display content. For example, suppose you wanted to reference the current Oracle stock price. You could create a region of type URL based on a URL such as the following:

```
http://quote.yahoo.com/q?d=b&s=ORCL
```

You could then create a item called STOCK_SYMBOL and base your region on a stock price entered by the user. For example:

```
http://quote.yahoo.com/q?d=b&s=&STOCK_SYMBOL.
```

Sometimes (as is the case with the previous example) the HTML returned to the region is more than is needed. To restrict the HTML displayed you can use the following region attributes:

- URL (discard until but not including this text)
- URL (discard after and including this text)

Note that the previous example may require that you set the Proxy Server application attribute. If you do not set the Proxy Server application attribute, you will get an error message. Oracle Application Express uses the Oracle `utl_http.request_pieces` function to obtain the HTML generated from the given URL.

> **See Also:** "Configuring Standard Application Attributes" on page 4-6 for information about setting the Proxy Server application attribute

Managing Themes

Themes are collections of templates that can be used to define the layout and style of an entire application. The idea behind a theme is to provide a complete set of templates that accommodate every UI pattern that may be needed in an application. Templates are organized first by type (button, calendar, label, list, breadcrumb, page, popup LOV, region, and report) and then by template classes, identifying the purpose of the each template within that type. Each template type provides a group of standard classes and eight custom classes. These classifications enable Oracle Application

Express to map templates among themes, making it easy to quickly change the entire look and feel of an application.

Topics in this section include:

- Accessing the Themes Page
- Changing Default Templates in a Theme
- Creating a New Theme
- Switching an Active Theme
- Copying a Theme
- Deleting a Theme
- About Exporting and Importing Themes
- Changing a Theme Identification Number
- Viewing Theme Reports

See Also: "Customizing Templates" on page 7-17

Accessing the Themes Page

You manage themes on the Themes page.

To access the Themes page from Shared Components:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click **Shared Components**.
4. Under User Interface, select **Themes**.

   The Themes page appears.
5. To access the Details view, select **Details** from the View list.

   The currently selected theme displays a check mark in the current column.

To access the Themes page from the Page Definition:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Select a page.

   The Page Definition appears.
4. Select the theme name.

About the Themes Page

Once you create a theme, it appears on the Themes page. You control how the page displays by selecting the following options from the View list:

- **Icons** (the default) displays each theme as a large icon. To edit a theme, click the appropriate icon.
- **Details** displays each theme as a line in a report. To change the theme name or default templates, click the theme name.

  In Details view, you can select the following options from the Display list:

- **Summary View** displays the theme ID, name, and current status.
- **Detailed View** displays the theme ID, name, current status, and the number of templates in each template type.

Changing Default Templates in a Theme

Standard theme contains templates for every type of application component and region type. You can change the selected default templates for a theme on the Define Theme page.

You can override these defaults, by either selecting another template when you create new components or regions, or by changing the template on the component or region attributes page.

To review or change the default templates in a theme:

1. Navigate to the Themes page:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under User Interface, select **Themes**.
2. Access the Details view. From the View list, select **Details**.
3. Select the appropriate theme name.

   Create/Edit Theme page appears.

   The top of the page displays the associated application ID and the Theme Identification Number.
4. To change the theme name, enter a new name in the Name field.
5. To change a default template, make a new selection from the appropriate list.

Table 7–4 describes the default templates available under the section Default Templates by Component.

Table 7–4 Default Templates by Component

Attribute	Description
Page	Identifies the default template for displaying pages. If a developer does not explicitly choose a template, then the Application Express engine uses the template specified here. Once defined, this default template appears on the Edit Application Attributes page under the heading Application Template Defaults. **See Also:** "Display Attributes" on page 4-21 for information about overriding the page template on the Page Attributes page
Error Page	Optional. Specifies a page template to use for errors that display on a separate page as opposed to those that display inline. Leave this attribute blank if you do not want to use a template designed to display errors. Once defined, this default template appears on the Edit Application Attributes page under the heading Application Template Defaults.

Table 7–4 (Cont.) Default Templates by Component

Attribute	Description
Printer Friendly Page	Identifies the template to be used when the Application Express engine is in printer friendly mode. When calling Application Express to render a page, you have the option to identify a printer friendly attribute with values of YES or NO. If you select `YES`, then the page displays using a printer friendly template. The Application Express engine displays all text within HTML form fields as text. The printer friendly template does not need to have the `#FORM_OPEN#` or `#FORM_CLOSE#` tags. The objective is to be able to display information with few tables and in a format suitable for printing. Once defined, this default template appears on the Edit Application Attributes page under the heading Application Template Defaults. **See Also:** "Optimizing a Page for Printing" on page 7-42
Breadcrumb	Identifies the default breadcrumb template used when you create new breadcrumb.
Button	Identifies the default button template used when creating a new button.
Calendar	Specifies the default calendar template used when you create new calendar.
Label	Identifies the default label template used when you create new label.
List	Specifies the default list template used when you create new list.
Region	Specifies the default region template used when you create new region.
Report	Identifies the default region template used when you create a report.

Table 7–5 describes the default templates available under the section Default Templates by Region Type.

Table 7–5 Region Templates by Region Type

Attribute	Description
Breadcrumbs	Default region template used when creating a breadcrumb.
Charts	Default chart template used when creating a chart.
Forms	Default form template used when creating a form.
Lists	Default region template used when creating a list.
Reports	Default region template used when creating a report.
Tabular Forms	Default region template used when creating a tabular form.
Wizards	Default region template used when creating a new wizard component.

Creating a New Theme

You can create a new theme from scratch or select an existing theme from the repository.

To create a new theme:

1. Navigate to the Themes page:
   a. On the Database Home Page, click the **Application Builder** icon.

   b. Select an application.
   c. Click **Shared Components**.
   d. Under User Interface, select **Themes**.
2. Click **Create**.
3. Specify whether to select a theme from the repository, or create a theme from scratch.
4. If you select **From Repository**:
   a. Select a theme from the repository.
   b. Click **Create**.
5. If you select **From Scratch**:
   a. Specify a name.
   b. Click **Create**.

      Themes page appears.
   c. Define the default templates for the new theme:
      – Click the **Edit** icon adjacent to the new theme name.
      – To change the theme name, enter a new name in the Name field.
      – When the Define Theme page appears, select default templates for the new theme.

Switching an Active Theme

When you switch to a new theme, all components that are assigned a template are assigned to a corresponding template in the new theme. Application Builder accomplishes template mapping through the assignment of template class identifiers.

Note: You can only switch to a new theme if another theme already exists.

To apply a theme to an application:

1. Navigate to the Themes page:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under User Interface, select **Themes**.
2. Click **Switch Theme**.

   The Switch Theme page appears.
3. From the Switch to Theme list, select a new theme and click **Next**.
4. Review the Status column to identify problematic mappings:
   - **Check** indicates the mapping was successful.

- **Warning** indicates there are more than one template in the theme you are switching to with the identified class. The warning provides a select list from which to choose the appropriate template.
- **Error** indicates that Application Builder was unable to map the class among the themes. Ensure that a class is identified for the templates in both themes.

5. Click **Next** to continue.
6. Click **Switch Theme**.

See Also: "Creating a New Theme" on page 7-11

Copying a Theme

Each theme is identified by a numeric identification number (ID). When you copy a theme you specify a new theme ID. Copying a theme is useful if you want to experiment with editing a theme or to export a theme with a different ID.

To copy a theme:

1. Navigate to the Themes page:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under User Interface, select **Themes**.
2. On the Tasks list, click **Copy Theme**.
3. On Copy Theme:
   a. Copy From Theme - Select the theme you want to copy.
   b. Copy to this Theme Identification Number - Enter a new ID for the theme.
   c. Click **Next**.
4. Click **Copy Theme ID**.

Deleting a Theme

You can only delete inactive themes. When you delete a theme, Application Builder only removes inactive templates.

To delete a theme:

1. Navigate to the Themes page:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under User Interface, select **Themes**.
2. On the Tasks list, click **Delete Theme**.
3. From Remove Theme, select the theme you want to delete and click **Next**.
4. Click **Remove Theme**.

About Exporting and Importing Themes

You export a theme in the same way you export any related application file. Exporting a theme from one development instance to another involves the following steps:

1. Export the theme using the Export Theme utility.
2. Import the exported file into the target Oracle Application Express instance.
3. Install the exported file from the Export Repository.

See Also: "Exporting Themes" on page 11-9 and "Importing Export Files" on page 11-11

Changing a Theme Identification Number

Each theme has an identification number (ID). You can use the Change Theme ID utility to change a theme ID to another number. Changing a theme ID is useful when you want to export a theme with a different number and then import it into another application.

To change a theme identification number:

1. Navigate to the Themes page:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under User Interface, select **Themes**.
2. On the Tasks list, click **Change Identification Number**.
3. On the Change Theme ID page:
   a. Select a theme.
   b. Specify a new identification number.
   c. Click **Next**.
   d. Confirm your changes and click **Change Theme ID**.

Viewing Theme Reports

Application Builder includes a number of reports designed to help you manage themes and templates.

Topics in this section include:

- Viewing All Templates in a Theme
- Viewing Theme Template Counts
- Viewing File References
- Viewing Class References
- Viewing Template Substitution Strings

Viewing All Templates in a Theme

To view all templates that comprise a theme:

1. Navigate to the Themes page:

   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under User Interface, select **Themes**.
2. Click **Reports**.
3. On the Theme Reports page:
   a. From Report, select **Application Templates**.
   b. From Theme, select a theme.
   c. Click **Go**.

   A listing of templates displays listing the template type, template name, the associated theme, and template class.
4. To edit a template, select the template name.

Viewing Theme Template Counts

The Theme Template Count report lists which template classes currently have templates created for them.

To view the Theme Template Count report:

1. Navigate to the Themes page:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under User Interface, select **Themes**.
2. Click **Reports**.
3. On the Theme Reports page:
   a. From Report, select **Theme Template Counts**.
   b. From Theme, select a theme.
   c. Click **Go**.
4. If you are using custom classifications, select **Show Custom** and click **Go**.

Viewing File References

The File References report displays a listing of all files associated with templates, shared components, or page components in the current application.

To view the File References report:

1. Navigate to the Themes page:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under User Interface, select **Themes**.
2. On the Themes page, click **Reports**.

3. On the Theme Reports page:
   a. From Report, select **File References**.
   b. From Theme, select a theme.
   c. Click **Go**.
4. On the File References page:
   a. From Show, select the type of component to include in the report. If you do not make a selection, no results are returned.
   b. From Show Files, select one of the following:
      – **With context** displays the component, the theme identification number, the component name, the image (if applicable), and the page ID. Select the page ID to link to a Page Definition.
      – **Without context** displays only the file name and the image (if applicable).
   c. From File Extensions, select the type of extensions for which to search.
   d. Click **Go**.
5. To download a comma-delimited file (.csv) version of this report, click **Download CSV** at the bottom of the page.

Viewing Class References

Accessing the Class References report displays a listing of classes associated with templates, shared components, or page components in the current application.

To view the Class References report:

1. Navigate to the Themes page:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under User Interface, select **Themes**.
2. Click **Reports**.
3. On the Theme Reports page:
   a. From Report, select **Class References**.
   b. From Theme, select a theme.
   c. Click **Go**.
4. On the Class References page:
   a. From Show, select the components to check for a class reference. If you do not make a selection, no results are returned.
   b. From Show Class Names, select one of the following:
      – **With context** displays the component, the theme identification number, the component name, the image (if applicable), and the page ID.
      – **Without context** displays only the referenced class.
   c. Click **Go**.

5. To download a comma-delimited file (.csv) version of this report, click **Download CSV** at the bottom of the page.

Viewing Template Substitution Strings

Use the Template Substitution Strings report to view all supported substitution strings by component.

To view the Substitution String report:

1. Navigate to the Themes page:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under User Interface, select **Themes**.
2. Click **Reports**.
3. On the Theme Reports page:
   a. From the Report list, select **Template Substitution Strings**.
   b. From the Theme list, select which themes to include in the report.
   c. Click **Go**.
4. To link to a template definition, select the component name.

   See Also: "Using Substitution Strings" on page 3-13

Customizing Templates

The Application Express engine creates the user interface of an application based on a named collection of templates called a theme. Templates control the look and feel of the components in your application. If you need to create a custom template, it is generally simplest to start with an existing template and then modify it. Once you have created one or more default templates, you can modify those templates to fit your specific needs.

Topics in this section include:

- About Cascading Style Sheets
- Selecting a Default Page Template
- Viewing Templates
- Creating a New Template
- Viewing Template Reports
- Editing Templates
- Breadcrumb Templates
- Button Templates
- Calendar Templates
- Label Templates
- List Templates
- Page Templates

- Popup LOV Templates
- Region Templates
- Report Templates

See Also: Managing Themes on page 7-8

About Cascading Style Sheets

A cascading style sheet (CSS) provides a way to control the style of a Web page without changing its structure. When used properly, a CSS separates visual attributes such as color, margins, and fonts from the structure of the HTML document. Oracle Application Express includes themes that contain templates that reference their own CSS. The style rules defined in each CSS for a particular theme also determine the way reports and regions display.

See Also: "Using Custom Cascading Style Sheets" on page 7-43

Selecting a Default Page Template

You can specify a default page template in two ways:

- Select a default page template within a specific theme.
- Select a specific page template on a page by page basis.

By default, the Application Express engine uses the Page template specified on the Themes page.

Selecting a Page-level Template Within a Theme

To specify a default page template within a theme:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click **Shared Components**.
4. Under User Interface, select **Themes**.

   The Themes page appears.
5. Change the View. From View, select **Details**.
6. In the Themes list, select the theme name.
7. Under Default Templates by Component, make a selection from the Page list.

See Also: "Changing Default Templates in a Theme" on page 7-10

Selecting a Page-level Template for a Specific Page

To specify an page-level template for a specific page:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Select a page.
4. Click **Edit Attributes**.
5. Locate the section Primary Display Attributes.
6. Make a selection from the Page Template list.

Viewing Templates

You can view all available templates on the Templates page. Alternatively, you can view the templates used on a specific page on the Page Definition.

See Also: "Viewing All Templates in a Theme" on page 7-14

Viewing Templates on the Templates Page

To view existing templates:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click **Shared Components**.
4. Under User Interface, select **Templates**.

   The Templates page appears.
5. You can narrow the display by making a selections from the following lists and clicking **Go**.
   - Theme - View only templates in a specific theme.
   - Show - View a specific type of template.
   - View - View all templates, those currently referenced, or those not referenced.
6. To view a template definition, click the template name.
7. To see a preview of a template, click the **Run** icon in the Preview column.

Note: Not all template types have the preview capability.

Viewing Templates from the Page Definition

To view existing templates from the Page Definition:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Select a page.

   The Page Definition appears. Templates associated with the current page display under the Templates heading in the far right column.
4. To view attributes of an existing template, click the template name.

Creating a New Template

If you need to create a custom template, it is generally simplest to start with an existing template and then modify it. Once you have created one or more default templates, you can modify those templates to fit your specific needs.

To create a custom template:

1. Navigate to the Templates page.
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.

   d. Under User Interface, select **Templates**.
2. Click Create.
3. Select the type of template you want to create.
4. Select a creation method:
   - **From Scratch**
   - **As a Copy of an Existing Template**
5. Follow the on-screen instructions. Be careful to associate your template with the correct theme.

Viewing Template Reports

Application Builder includes reports describing template utilization, subscription, and edit history.

To view template reports for the current application:

1. Navigate to the Themes page:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under User Interface, select **Templates**.
2. You can narrow the display by making a selections from the following lists and clicking **Go**.
   - Theme - View only templates in a specific theme.
   - Show - View a specific type of template.
   - View - View all templates, those currently referenced, or those not referenced.
3. To view template reports, click the following buttons:
   - **Utilization** displays template utilization in the current application for all template types (page, report, region, label and list).
   - **Subscription** displays subscribed templates in your application.
   - **History** details recent changes to templates by developer and last update date.

Editing Templates

Once you create a custom template, you can quickly edit it from either the Templates page or from the Page Definition.

To edit an existing template from the Templates page:

1. Navigate to the Templates page.
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under User Interface, select **Templates**.
2. You can narrow the display by making a selections from the following lists and clicking **Go**.

- Theme - View only templates in a specific theme.
- Show - View a specific type of template.
- View - View all templates, those currently referenced, or those not referenced.

3. Locate the template you want to edit and select the template name.
4. Follow the on-screen instructions.

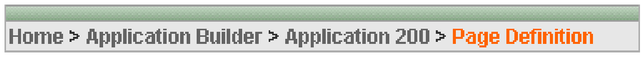
As you edit templates, you can make changes in one window and run your application in another by selecting **Return to Page**. Selecting this check box, keeps the page you are editing current after you click **Apply Changes**.

Breadcrumb Templates

A breadcrumb template controls the display of breadcrumb entries. You select a breadcrumb template when you create a region.

About Breadcrumb Style Navigation

Breadcrumbs usually indicate where the current page is relative to other pages in the application. In addition, users can click a specific page to instantly view it. The Database Home Page user interface includes breadcrumb paths beneath the standard tabs (or second level navigation tabs) at the top of each page.


Home > Application Builder > Application 200 > Page Definition

See Also:

- Online help for information about using specific sections of the Edit Breadcrumb Template page
- "Creating Breadcrumbs" on page 6-9

Breadcrumb Template Attributes

This section describes specific sections of the Breadcrumb Template page.

Identification **Name** identifies the name of the template. Use the **Translatable** check box to indicate that the template contains text strings that require translation. **Theme** indicates the theme to which the template is a member.

Template Class identifies a specific use for the template. When you switch to a new theme, all templates in one theme are mapped to corresponding templates in another theme. Application Builder accomplishes this template mapping through the assignment of a template class.

Breadcrumb Template Subscription Use Template Subscription to apply an existing template to the current application. When you select an existing template, you become a subscriber to that template.

To load a new copy of a master template, click Refresh.

Breadcrumb Template Style) Select one of the following template styles:

- **Child Breadcrumb Entries** displays all breadcrumb entries that are children of the current page parent breadcrumb (that is, peers of the current breadcrumb).
- **Current Breadcrumb** displays all breadcrumb entries in sequence with a common parent.

- **Parent Breadcrumb Entries** displays all breadcrumb entries for the current pages parent breadcrumb (that is, one level up from current breadcrumb entry).
- **Parent to Leaf (breadcrumb style)** displays the current page breadcrumb entry, its parent to the left, and so on until the root node is reached.

Breadcrumb Entry Control Table 7–6 describes available breadcrumb Entry attributes.

Table 7–6 Breadcrumb Entry Control attributes

Attribute	Description
Before First	Defines text that displays before the first breadcrumb entry.
Current Page Breadcrumb Option	Defines the look of a breadcrumb entry that corresponds to the current page. This attribute supports the following substitution strings: ▪ `#NAME#` specifies the short name of the breadcrumb entry. ▪ `#LINK#` specifies the anchor target of the breadcrumb entry. ▪ `#LONG_NAME#` specifies the long name of the breadcrumb entry.
Non Current Page Breadcrumb Option	Defines the look of a breadcrumb entry that does not correspond to the current page. This attribute supports the following substitution strings: ▪ `#NAME#` specifies the short name of the breadcrumb entry ▪ `#LINK#` specifies the anchor target of the breadcrumb entry ▪ `#LONG_NAME#` specifies the long name of the breadcrumb entry
After Last	Defines text that displays after the last breadcrumb entry.

Template Attributes Use **Breadcrumb Link Attributes** to specify hypertext link attributes for a breadcrumb entry.

Use **Between Levels** to specify text that displays between each level of a breadcrumb breadcrumb. For example, if your breadcrumb has three levels, this text would display at the "X" in the example that follows:

```
main X cars X porsche X 911
```

In **Max Levels** specify the number of levels when displaying breadcrumbs.

Comments Use this attribute to record developer comments.

Button Templates

Button templates enable application developers to customize the look and feel of a button. To build a button, you can use multiple images or HTML tags. Using button templates is optional.

Button Template Attributes

This section describes specific sections of the Button Template page.

Button Template **Template Name** identifies the name of the template. Use the **Translatable** check box to indicate if the template contains text strings which require translation. **Theme** indicates the theme to which the template is a member.

Template Class identifies a specific use for the template. When you switch to a new theme, all templates in one theme are mapped to corresponding templates in another theme. Application Builder accomplishes this template mapping through the assignment of a template class.

Template Subscription Use Template Subscription to apply an existing template to the current application. When you select an existing template, you become a subscriber to that template.

To load a new copy of a master template, click Refresh.

Template Text Defines the button template that displays. You have the option of including standard application substitutions. For example, `&ITEM_NAME` values can be substituted at rendering time. Button templates support the following substitution strings:

- `#LABEL#` is replaced with a button label.
- `#LINK#` is replaced with a URL. The URL then calls a `#doSubmit#` or a redirect JavaScript which submits the page (that is, setting the request value), or simply redirects it to the supplied URL.

Comments Use this attribute to record developer comments.

Calendar Templates

Calendar templates control the appearance and placement of a calendar. Calendar templates frequently use HTML tables to arrange dates. You place calendar attributes using substitution strings such as `#DD#` and `#MONTH#`. A list of supported substitution strings appears on the right side of the Calendar Template Attributes page. Note that template substitution strings must be in uppercase letters and begin and end with a number sign (#).

See Also: "Creating Calendars" on page 5-34

Calendar Template Attributes

This section describes specific sections of the Calendar Template page.

Calendar Template Identification **Name** identifies the name of the template. **Theme** indicates the theme to which the template is a member.

Template Class identifies a specific use for the template. When you switch to a new theme, all templates in one theme are mapped to corresponding templates in another theme. Application Builder accomplishes this template mapping through the assignment of a template class.

Template Subscription Use Template Subscription to apply an existing template to the current application. When you select an existing template, you become a subscriber to that template.

To load a new copy of a master template, click Refresh.

Month Attributes In **Month Title Format** enter the format for the monthly title that appears at the top of each month. This is the first part of a calendar which prints on the page. For example:

```
<table>
 <tr>
```

```
   <td>#MONTH#</td>
 </tr>
```

In **Day of Week Format** enter the format for the week day names which displays as the column header for that day of the week. For example:

```
<th width="14%">#IDAY#</th>
```

In **Month Open Format** enter HTML to be used to open a month. This displays immediately after the Month Title Format. Typically this attribute contains an HTML tag that functions as a container (such as a table). For example:

```
<table  border="0" cellpadding="0" cellspacing="0" class="htmldbRowWithBorders"
width="100%"><tr>
```

In **Close Month Format** enter HTML to be used to close a month. Since this is the last part printed, this attribute should contain HTML that closes the HTML tags used in the Month Open Format. For example:

```
</table>
```

Week Attributes Enter HTML to open and close a week.

In **Week Open Format** enter HTML to be used to open a week. This is printed for each week. Typically this attribute contains an HTML tag which functions as a container. For example:

```
<tr>
```

In **Week Close Format** enter HTML to be used to close the week. Since this is the last part printed, this attribute should contain HTML that closes HTML tags used in Week Open Format.

```
</tr>
```

Weekday Attributes Enter HTML to format the days that occur during the work week (that is, Monday through Friday).

In **Day Title Format** enter HTML to be used the title of each day. This title displays after the Day Open Format. For example:

```
#DD#
```

In **Day Open Format** enter HTML to used to open a day. This displays on each day in the calendar. Typically this attribute contains an HTML tag that functions as a container. For example:

```
<td>
```

In **Day Close Format** enter HTML used to close a day. Since this is the last part printed, this attribute should close any HTML tags used in Day Open Format. For example:

```
</td>
```

In **Today Open Format** enter HTML used to open today. Typically this attribute contains an HTML tag which functions as a container (such as `<td>`) and would be different from the Day Open Format. For example:

```
<td style="background:#c5d5c5">
```

Non-Day Attributes A non-day is not part of the current month. For example, suppose the first of a month is a Monday, but the week starts on a Sunday. Because Sunday is not part of the current month, Sunday would be a non-day. Use these attributes to format non-days.

In **Non-Day Title Format** enter a non-day title. For example:

```
#DD#
```

In **Non-Day Open Format** enter HTML to open a non-day. Typically this attribute would contain an HTML tag that functions as a container. For example:

```
<td>
```

In **Non-Day Close Format** enter HTML to close a non-day. Typically this attribute would contain an HTML tag that closes the tag used in Non-Day Open Format. For example:

```
</td>
```

Weekend Attributes Enter HTML used to format days that occur on the weekend. Include substitution strings to include dynamic content. To view a list of supported substitution strings, see the Substitution Strings list on the right side of the Calendar Template page.

In **Weekend Title Format,** enter HTML to be used for a day occurring on a weekend. For example:

```
#DD#
```

In **Weekend Open Format**, enter HTML to open a day which is on a weekend. Typically this attribute would contain an HTML tag that functions as a container. For example:

```
<td>
```

In **Weekend Close Format,** enter HTML to close a day which is in a weekend. Since this is the last part printed, this attribute should close any HTML tags used in Weekend Open Format.For example:

```
</td>
```

Comments Use this attribute to record developer comments.

Label Templates

Label templates are designed to centrally manage HTML markup of page item labels. Each item can have an optional label. You can control how these labels display using label templates. For example, you could create a label template called Required Field that references an image (such as an asterisk) to indicate to the user that the field is required.

Label templates enable you to define a before-and-after text string that gets prepended and appended to the item.

Label Template Attributes

This section describes specific sections of the Label Template page.

Label Template Attributes **Template Name** identifies the name of the template. Use the **Translatable** check box to indicate that the template contains text strings that require translation. **Theme** indicates the theme to which the template is a member.

Template Class identifies a specific use for the template. When you switch to a new theme, all templates in one theme are mapped to corresponding templates in another theme. Application Builder accomplishes this template mapping through the assignment of a template class.

Template Subscription Use **Template Subscription** to apply an existing template to the current application. When you select an existing template, you become a subscriber to that template.

To load a new copy of a master template, click Refresh.

Template Definition (normal display) In **Before Label**, enter HTML to display before the item label. Before Label supports the substitution strings `#CURRENT_FORM_ELEMENT#`; `#CURRENT_FORM_ID#`, and `#CURRENT_ITEM_NAME#`. For example:

```
<label for="#CURRENT_ITEM_NAME#">
<a href="javascript:popupFieldHelp('#CURRENT_ITEM_ID#',
  '&SESSION.','&CLOSE.')" >
```

In **After Label**, enter HTML to display after the item label. Since the label will be automatically display before the HTML in this region, any open HTML tags in the Before Label region should be closed here. For example:

```
</a></label>
```

Label Template Attributes (error display) In **On Error Before Label**, enter HTML to precede the item label when an application displays an inline validation error message for the item. For example:

```
<font class="fieldtitleleft">#ERROR_MESSAGE#</font>
```

In **On Error After Label** enter HTML to be appended to the item label when a application displays an inline validation error message for the item. This attribute supports the substitution strings `#CURRENT_FORM_ELEMENT#`, `#CURRENT_FORM_ID#`, and `#CURRENT_ITEM_NAME#`. The following example would append a space and a closing bracket to the displayed item label with the error.

```
&nbsp;]</font>
```

Template Comments Use this attribute to record developer comments.

List Templates

A list is a shared collection of links. You control the appearance of a list through list templates. Using template attributes, you can also define a list element to be either current or non current for a specific page.

About Hierarchical Lists

Oracle Application Express supports hierarchical lists. To create a hierarchical list, you must:

- Select a list template that supports hierarchical lists. To determine which list templates support hierarchical lists, look for templates having the naming convention "with Sublist."
- Select a Parent List Entry when you create each list entry.

See Also:

- Online Help for information about using specific sections of the Edit List Template page
- "Creating Lists" on page 6-13

List Template Attributes

This section describes specific sections of the List Template page.

Template Identification **Name** identifies the name of the template. Use the **Translatable** check box to indicate that the template contains text strings that require translation. **Theme** indicates the theme to which the template is a member.

Template Class identifies a specific use for the template. When you switch to a new theme, all templates in one theme are mapped to corresponding templates in another theme. Application Builder accomplishes this template mapping through the assignment of a template class.

Template Subscription Use **Template Subscription** to apply an existing template to the current application. When you select an existing template, you become a subscriber to that template.

To load a new copy of a master template, click Refresh.

Before List Elements Enter HTML that displays before any list elements. You can use this attribute to open an HTML table or HTML table row.

List Element Display Defines current and noncurrent list templates. Supported substitution strings include `#LINK#`, `#TEXT#`, `#IMAGE_PREFIX#`, `#IMAGE#`, `#IMAGE_ATTR#`, and `#A01#` to `#A10#`.

- **List Template Current.** Enter HTML or text to be substituted for the selected (or current) list template.
- **List Template Current with Sub List Items.** Enter HTML or text to be substituted for the selected (or current) list template when an item has sublist items. If not specified, the current list item template will be used.
- **List Template Noncurrent.** Enter HTML or text to be substituted for the unselected (or noncurrent) list template.
- **List Template Noncurrent with Sub List Items.** Enter HTML or text to be substituted for the unselected (or noncurrent) list template used when an item has sublist items. If not specified, the current list item template will be used.
- **Between List Elements.** Enter HTML that displays between list elements. This attribute will be ignored if no HTML is specified.

Before Sub List Elements Enter HTML that displays before any sublist elements.

Sub List Elements Defines current and noncurrent list templates. Supported substitution strings include `#LINK#`, `#TEXT#`, `#IMAGE_PREFIX#`, `#IMAGE#`, `#IMAGE_ATTR#`, and `#A01#` to `#A10#`.

- **Sub List Template Current.** Enter HTML or text to be substituted for the selected (or current) list template.
- **Sub List Template Current with Sub List Items.** Enter HTML or text to be substituted for the selected (or current) list template when an item has sublist items. If not specified, the current list item template will be used.
- **Sub List Template Noncurrent.** Enter HTML or text to be substituted for the unselected (or noncurrent) list template.
- **Sub List Template Noncurrent with Sub List Items.** Enter HTML or text to be substituted for the unselected (or noncurrent) list template used when an item has sublist items. If not specified, the current list item template will be used.
- **Between Sub List Elements.** Enter HTML that displays between list elements. This attribute will be ignored if no HTML is specified.

After Sub List Elements Enter HTML that displays after displaying sublist elements.

After List Elements Enter HTML that displays after displaying all list elements. You can use this attribute to close a HTML table opened in the Before List Elements attribute.

Comments Use this attribute to record developer comments.

Page Templates

Page templates define the appearance of a page. Each template consists of a header template, a body template, a footer template, and a number of subtemplates. If you do not specify a page template as a page-level attribute, then the Application Express engine uses the default page template defined on the Define Theme page.

Page templates combine static HTML with substitution strings that are replaced at run time. You use substitution strings to indicate the existence and placement of a component within a page template. You can further specify how a component should display using subtemplates.

Topics in this section include:

- Supported Page Template Substitution Strings
- Page Template Attributes

Supported Page Template Substitution Strings

Table 7–7 describes the available page template substitution strings. Note that all template substitution strings must be in uppercase letters and begin and end with a number sign (#).

Table 7–7 Page Template Substitution Strings

Substitution String	Description
`#APP_VERSION#`	Can be used in the Header or Footer sections of the page template. You define the value of `#APP_VERSION#` in the Version attribute on the Edit Application Attributes page **See Also:** "Name" on page 4-7
`#BOX_BODY#`	Identifies where the Body displays. If the Body is null, then `#BOX_BODY#` will be used instead.

Table 7–7 (Cont.) Page Template Substitution Strings

Substitution String	Description
`#CUSTOMIZE#`	Can be used in the Header, Body, or Footer sections of the page template. The Customization section of the Region Definition enables you to turn on end user customization. To utilize this feature, you must also include the `#CUSTOMIZE#` substitution string in the page template. If at least one region supports end user customization, a link called **Customize** appears wherever the `#CUSTOMIZE#` substitution string appears in the page template. When users click this link, a window displays enabling them to turn on and off regions on the page. **See Also:** "How Region Attributes Affect Page Layout" on page 7-4
`#FORM_CLOSE#`	If a `#FORM_OPEN#` is included, then you must include a `#FORM_CLOSE#` in the header, body, or footer template. `#FORM_OPEN#` must appear before the `#BOX_BODY#` and `#FORM_CLOSE#` must appear after the `#BOX_BODY#` substitution string.
`#FORM_OPEN#`	Specifies where the HTML open form tag `<form>` is placed. You must include this substitution string in order to submit a form. You do not need to code your own form open, the Application Express engine does it for you.
`#GLOBAL_NOTIFICATION#`	Displays the Global Notification attribute. Global notifications are intended to communicate system status, such as pending system downtime. You can also use `HTMLDB_APPLICATION.G_GLOBAL_NOTIFICATION` to set this value if you want to set it programmatically. **See Also:** "Global Notifications" on page 4-9 for information about the Global Notification attribute
`#HEAD#`	Used after the `<head>` open tag, but before the `</head>` close tag. You can optionally define the contents of `#HEAD#` for each page (for example, to reference additional style sheets or JavaScript libraries).
`#LOGO#`	Identifies an application logo. In the Logo section of the Edit Application Attributes page, you can identify an image and image attributes for an application logo. To utilize this feature, you must also include the `#LOGO#` substitution string in the Header or Body page template. **See Also:** "Logo" on page 4-10
`#NAVIGATION_BAR#`	Defines the existence of navigation bar entries. A navigation bar will appear on every page in your application that uses a template that includes this substitution string.You can expand this substitution string using the Navigation bar subtemplate. See Also: "Subtemplate" on page 7-31 for information about Navigation Bar subtemplate
`#NOTIFICATION_MESSAGE#`	Enables developers to communicate messages to the user. Defines where a summary of inline error messages is displayed. Inline error messages can be displayed next to a field, inline in the notification area, or both.

Table 7–7 (Cont.) Page Template Substitution Strings

Substitution String	Description
`#ONLOAD#`	Can be used in the Header and Footer section of the page template and should be placed inside the <body> html tag. For example: `<body #ONLOAD#>` Use this string as a substitute in a JavaScript call to be executed when a page is loaded by the Web browser. The JavaScript to be called can vary for each page.
`#PARENT_TAB_CELLS#`	Identifies the display of parent tabs. Parent tabs require standard tabs. If your application only has one of level tabs, you do not need this substitution string. **See Also:** "Standard Tab Attributes" on page 7-31 for information about defining Parent Tab Attributes
`#REGION_POSITION_NN#`	Identifies the exact placement of regions within a page. If no region is specified (for example, `#REGION_POSITION_01#`), then `#REGION_POSITION_01#` will be replaced with nothing.
`#SUCCESS_MESSAGE#`	Defines where in the page success and error messages appear. If the page process runs without raising errors, then this text displays. You can customize the display of the success message for each template by adding HTML to be displayed before and after the success message.
`#TAB_CELLS#`	Identifies the display of standard tabs. **See Also:** "Standard Tab Attributes" on page 7-31
`#TITLE#`	Defines the page title. Typically included within HTML title tags.

See Also:

- "Using Substitution Strings" on page 3-13
- "Adding Pages to an Application" on page 5-8

Page Template Attributes

This section describes specific sections of the Page Template page.

Name **Name** identifies the name of the template. Use the **Translatable** check box to indicate that the template contains text strings that require translation. **Theme** indicates the theme to which the template is a member.

Template Class identifies a specific use for the template. When you switch to a new theme, all templates in one theme are mapped to corresponding templates in another theme. Application Builder accomplishes this template mapping through the assignment of a template class.

Subscription Use **Subscription** to apply an existing template to the current application. When you select an existing template, you become a subscriber to that template.

To load a new copy of a master template, select Refresh.

Definitions Each template consists of a header, a body, a footer, and subtemplates. Use substitution strings to include dynamic content. All template substitution strings must be in uppercase letters and begin and end with a number sign (#). See item Help for information about supported substitution strings.

Header is the first section of the page template. Enter HTML that defines the `<Head>` section of the HTML document. Regions that display or processes and computations that execute AFTER HEADER will display or execute immediately after this section in the template is rendered. For example:

```
<html>
 <head>
   <title>#TITLE#</title>
   #HEAD#
 </head>
```

Body is the second section in the page template and is rendered after the header section, but before the footer section. Enter HTML that defines the `<Body>` section of the HTML document. At a minimum, you must include the `#BOX_BODY#` substitution string. It is recommended that you also include the `#FORM_OPEN#` and `#FORM_CLOSE#` substitution strings. For example:

```
   <body #ONLOAD#>
      #FORM_OPEN#
      #BOX_BODY#
      #FORM_CLOSE#
   </body>
```

Footer is the third section in the page template that displays after the body.

Display Points **Breadcrumb Display Point** applies to generated components that use breadcrumbs and defines where the breadcrumbs are placed on the page. **Sidebar Display Point** applies to generated components that use Sidebars and defines where sidebars are placed on the page.

Subtemplate Use **Subtemplate** to specify how a component should display. Available subtemplates include:

- **Success Message.** Expands the `#SUCCESS_MESSAGE#` substitution string. You can define a success message either programmatically or as an attribute of a process. If a success message exists and if the page template includes the `#SUCCESS_MESSAGE#` substitution string, then this subtemplate is used to render the message.
- **Navigation Bar.** Controls the display of navigation bar entries. Enter HTML or text to be substituted when the `#NAVIGATION_BAR#` substitution string is referenced in the template header, body, or footer. Use the `#BAR_BODY#` substitution string to identify where each navigation bar icon should display.
- **Navigation Bar Entry.** Enter HTML or text that to be substituted into the navigation bar `#BAR_BODY#` substitution string for each navigation bar entry. Use the following substitution strings to create the navigation bar entry subtemplate.
- **Notification.** Enter HTML or text to be substituted when the `#NOTIFICATION_MESSAGE#` substitution string is referenced in the template header, body or footer. Use the substitution string `#MESSAGE#` to indicate where in the Notification Message the body of the message will appear.

Standard Tab Attributes You must populate this attribute if your application includes standard tabs. Standard tabs can be placed in the header, body, or footer sections of the page template using the `#TAB_CELLS#` substitution string. The page template Header/Body/Footer defines the HTML table and rows. This subtemplate defines how these tabs display by defining the specific cell. Available attributes include:

- **Current Tab.** Enter HTML or text to be substituted for the currently selected standard tab. Whether or not a tab is current is determined by standard tab attributes. For example:

  ```
  <td>#TAB_LABEL#</td>
  ```

- **Non Current Standard Tab.** Enter HTML or text that will be substituted for the unselected standard tabs. Use the `#TAB_TEXT#` substitution string to position a tab's label and link within the template. For example:

  ```
  <td><a href="#TAB_LINK#">#TAB_LABEL#</a></td>
  ```

See Also: "Creating Tabs" on page 6-5

Parent Tab Attributes You must populate this attribute if your application includes two levels of tabs. Enter HTML or text that will be substituted for the selected parent tabs. Parent tabs can be placed in the header, body, or footer section of the page template using the `#PARENT_TAB_CELLS#` substitution string. Parent tabs only display in conjunction with standard tabs. Available attributes include:

- **Current Parent Tab.** Enter HTML or text that will be substituted for the selected parent tabs. Whether or not a tab is current is determined by the page that displays and the standard tab set the page uses. Use the `#TAB_TEXT#` substitution string to position a tab's label and link within the template. For example:

  ```
  <td><a href="#TAB_LINK#">#TAB_LABEL#</a></td>
  ```

- **Non Current Parent Tab.** Enter HTML or text that will be substituted for the unselected parent tabs. Use the `#TAB_TEXT#` substitution string to position a tab's label and link within the template. For example

  ```
  <td><a href="#TAB_LINK#">#TAB_LABEL#</a></td>
  ```

See Also: "Creating Tabs" on page 6-5

Image Based Tab Attributes Use this subtemplate for tabs that are entirely based on images. Available attributes include:

- **Current Image Tab.** Enter HTML to be used to indicate that an image based tab is currently selected. Include the `#TAB_TEXT#` substitution string to show the displayed name of the tab.
- **Non Current Image Tab.** Enter the HTML to be used to indicate that an image tab is not currently selected. Include the `#TAB_TEXT#` substitution string to show the displayed name of the tab.

Multi Column Region Table Attribute If the Application Express engine displays regions in multiple columns in the same region position, then Oracle Application Express will render an HTML table. This attribute enables you to control the attributes of the `<table>` tag.

Error Page Template Control Use this attribute only when a page template will be designated as an error template. Use the `#MESSAGE#` substitution string to place the error message and the `#BACK_LINK#` substitution string to display a link back to the previous page. A template can be designated as an error template by editing the application attributes. For example:

```
#MESSAGE#

<br>

<a href="#BACK_LINK#">back</a>
```

Comments Use this attribute to record developer comments.

Popup LOV Templates

Popup LOV templates control how popup lists display for all items defined as `POPUP`. You can only specify one popup LOV template for each theme.

> **See Also:** "Creating Lists of Values" on page 5-60

Popup List of Values Template Attributes

This section describes specific sections of the Popup List of Values Template page.

Application Identification **Theme** indicates the theme to which the template is a member. **Template Class** identifies a specific use for the template. When you switch to a new theme, all templates in one theme are mapped to corresponding templates in another theme. Application Builder accomplishes this template mapping through the assignment of a template class. Use the **Translatable** check box to indicate that the template contains text strings that require translation.

Template Subscription Use **Template Subscription** to apply an existing template to the current application. When you select an existing template, you become a subscriber to that template.

To load a new copy of a master template, click Refresh.

Form Icon Use **Popup Icon** to specify an icon to display to the right of a form field for items of type `POPUP`. By default, the Application Express engine uses a `list.gif` image. Use **Popup Icon Attr** to defines image attributes (such as height and width) for the Popup Icon

Search Field Use these attributes to specify how a Search field displays. Table 7–8 describes available Search Field attributes.

Table 7–8 Search Field Attributes

Attribute	Description
Before Field Text	Defines text to display before the popup list of values search field displays.
Filter Width	Displays the text field using this width.
Filter Max Width	Displays the text field widget using this maximum width.
Filter Text Attribute	Displays the text field using these attributes. This will be included within the HTML input tag.
After Field Text	Displays this text after displaying the search field, the search button, and the close button.

Buttons Use these attributes to define the button name and attributes for the Find, Close, Next, and Previous buttons.

Window Popup lists of values are executed using JavaScript. Use these attribute to control the values of `scrollbars=`, `resizable=`, `width=`, and `height=`. For information about default values see item Help.

Pagination Defines how row count results display.

Result Set Use these attributes to define text or HTML to display before and after a result set.

Page Attributes Use these attributes to define popup pages. For more information, see item Help.

Region Templates

Region templates control the appearance and placement of region attributes. Region templates frequently use HTML tables to arrange content.

Region templates apply style elements to regions. Region templates display substitution strings. The only required substitution string, `#BODY#`, identifies where the source of the region should be placed. All other substitution strings are optional. You can use these substitution strings to indicate the existence and placement of a page control (such as a button) within the region.

Region Template Attributes

This section describes specific sections of the Region Template page.

Region Template Identification **Name** identifies the name of the template. Use the **Translatable** check box to indicate that the template contains text strings that require translation. **Theme** indicates the theme to which the template is a member.

Template Class identifies a specific use for the template. When you switch to a new theme, all templates in one theme are mapped to corresponding templates in another theme. Application Builder accomplishes this template mapping through the assignment of a template class.

Template Subscription Use **Template Subscription** to apply an existing template to the current application. When you select an existing template, you become a subscriber to that template.

To load a new copy of a master template, click Refresh.

Region Template `##BODY#` is the only required substitution string. It identifies where the source of the region should be placed. All other substitution strings are optional. The following are valid substitution strings:

- `#TITLE#`
- `#EXPAND#`
- `#CHANGE#`
- `#BODY#`
- `#FORM_OPEN#`
- `#FORM_CLOSE#`

When you create a button in a region position, the positions you defined will appear in a select list. Use the following substitution strings to define positions for placement of buttons in a region:

- `#EDIT#`
- `#CLOSE#`
- `#CREATE#`
- `#EXPAND#`
- `#HELP#`
- `#DELETE#`
- `#COPY#`
- `#NEXT#`
- `#PREVIOUS#`

> **See Also:** "Using Substitution Strings" on page 3-13

Form Table Attributes Page items display within regions. Items are rendered as HTML form elements in an HTML table. With this template property, you can define attributes that will be placed in the `<table>` tag. For example:

```
class="instructiontext"
```

Comments Use this attribute to record developer comments.

Report Templates

Report column templates provide you with control over the results of a row from a SQL query. This type of template defines a cell not an entire row

Each report template identifies column names using the syntax `#1#, #2#, #3#` and so on. You can also name columns using column name substitution syntax such as `#ENAME#` or `#EMPNO#`. You can reference any item from your application within your template. For example, to reference an item called *`ABC.`* in your template, you could include the exact substitution string *`&ABC..`* The actual value of ABC. would be provided by an end user editing an item in your application named *`ABC`*.

Topics in this section include:

- About Generic Column Templates and Named Column Templates
- Report Column Template Attributes for Generic Column Templates
- Report Column Template Attributes for Named Column Templates
- About Using JavaScript in Column Templates

About Generic Column Templates and Named Column Templates

Oracle Application Express includes two types of report templates:

- Generic column templates
- Named column templates

Generic Column Templates A generic column template determines the appearance of a report by defining the look of the column once. This look is then repeated as many times as is necessary based on the number of columns specified in the report's definition. This type of template is limited to reports that have a standard row and column structure. Additional style can be applied to a report using this type of template through the use of conditions.

The following example demonstrates how to have each column use a specific style:

```
<td class="tabledata" align="#ALIGN#">#COLUMN_VALUE#</td>
```

This example assumes your page template includes a CSS containing the class `tabledata`. This example also demonstrates the use the substitution strings `#ALIGN#` and `#COLUMN_VALUE#`. If you actually ran this report, these substitution strings would be replaced with values generated by the results of a SQL query.

If your query uses an expression in the select list, it is a good idea to create an alias for the columns to avoid run time errors. For example, suppose your query was as follows:

```
SELECT ename, (sal + comm) * 12 FROM emp
```

You could rewrite the query to alias the columns as follows:

```
SELECT ename, (sal + comm) * 12 yearly_comp FROM emp
```

Named Column Templates Named column templates allow for more flexibility in report design. However, because they reference columns by name, they can only be used by reports that are based on those columns. For example:

```
<tr><td>#ENAME#</td><td>#SAL#</td></tr>
```

Although named column templates offer flexibility, you may need to create a new template for each query. You can also include a position notation. The following example demonstrates how to use following HTML and substitution strings:

```
<tr><td>#ENAME#</td><td>#SAL#</td></tr>

<tr><td>#1#</td><td>#2#</td></tr>
```

Report Column Template Attributes for Generic Column Templates

This section describes specific sections of the Report Column Template page for Generic Column Templates.

Report Template Identification **Template Name** identifies the name of the template. Use the **Translatable** check box to indicate the template contains text strings that require translation. **Template Type** indicates the type of template. Named Column templates reference column names in the template. Generic Column Templates reference the `#COLUMN_VALUE#` substitution string in the template.

Theme indicates the theme to which the template is a member. **Template Class** identifies a specific use for the template. When you switch to a new theme, all templates in one theme are mapped to corresponding templates in another theme. Application Builder accomplishes this template mapping through the assignment of a template class.

Template Subscription Use **Template Subscription** to apply an existing template to the current application. When you select an existing template, you become a subscriber to that template.

To load a new copy of a master template, click Refresh.

Before Rows In **Before Rows**, enter HTML that displays once at the beginning of a report template. Opening an HTML table is a common use of this attribute as shown in the following example:

```
<table>
```

You can identify column headers using the syntax `#1#`, `#2#`, `#3#`. For example:

```
<th>#1#</th><th>#2#</th><th>#3#</th>
```

You can include pagination above a report by including the substitution string `#TOP_PAGINATION#`. This substitution string generates HTML which starts with an opening `<tr>` tag and ends with a closing `</tr>` tag. For example, to include an open table tag and the `#TOP_PAGINATION#` substitution string you would enter the following:

```
<table>#TOP_PAGINATION#
```

You can also include the substitution string `#CSV_LINK#` to include support for exporting your report to comma separated value (CSV) format, a format compatible with most spreadsheet programs.

Column Headings Use **Column Heading Template** to add color to each column header cell. Note that the text of this attribute must indicate where the cell heading text will be colorized. For example:

```
 <th #ALIGNMENT#>#COLUMN_HEADER#</th>
```

If you do not want any column headings, enter the following:

```
OMIT
```

If you do use this attribute, Application Express engine applies the default column heading template.

Before Each Row In **Before Each Row,** enter text to display before all columns in the report. Use this attribute to open a new HTML row. **Before Each Row** supports the following substitution strings:

- `#ROWNUM#`

  Use this substitution strings to specify the current row.

- `#COLCOUNT#`

  Use this substitution strings to specify the number of columns.

- `#HIGHLIGHT_ROW#`

  Use this substitution strings to specify the number of highlighted rows.

Column Templates Column templates define the look of each column. You can define up to four column templates, each can be conditional. For example, you can have different background colors for even and odd rows, or highlight rows which meet a PL/SQL defined condition.

In each Column Template, you define the look of each column. Column Templates support the substitution strings described in Table 7–9.

Table 7–9 Column Template Substitution Strings

Substitution String	Description
`#ALIGNMENT#`	Determines the column alignment. Specified by the user.
`#COLCOUNT#`	Defines the count of the number of columns.
`#COLNUM#`	Defines the current column number.
`#COLUMN_HEADER#`	Defines the column header.
`#COLUMN_VALUE#`	Replaced with the value of the column.
`#ROWNUM#`	Specifies the current row number.

Consider the following example:

```
<td #ALIGNMENT#>#COLUMN_VALUE#</td>
```

If you actually ran this report, these substitution strings would be replaced with values generated by the results of a SQL query.

By creating conditions, you can create a report that displays columns differently depending on whether or not the specified condition is met. To specify a column template be used conditionally, select a condition type from the Column Template Condition list. Valid values include:

- **Use Based on PL/SQL Expression.** Conditionally formats columns based on data in that row.
- **Use for Even Numbered Rows.** Conditionally formats even numbered rows.
- **Use for Odd Numbered Rows.** Conditionally formats odd numbered row.

If you select **Use Based on PL/SQL Expression**, the next step is to enter a PL/SQL expression in Column Template Expression field. For example, the following expression displays a value in bold if the value is greater than 2000:

```
#SAL# > 2000
```

Note that you could also use the substitution string `#ROWNUM#`. For example:

```
#ROWNUM# > 2000
```

After Each Row In **After Each Row,** enter HTML that displays after all columns in the report display. This attribute is often used to close an HTML table row. For example:

```
</tr>
```

After Rows Use **After Rows** to specify text that should display after the last row. A common use of this attribute is to close the HTML table tag. For example:

```
</table>
```

The After Rows attribute supports the following substitution strings:

- `#PAGINATION#`

  Replaced with a pagination attribute.
- `#COLCOUNT#`

  Substituted at run time with the number of columns defined in the report.

Row Highlighting Use **Background color for checked row** to control the background color of a report row when the row selector is checked. Use **Background color for current row** to control the background color of a report row when the user moves the mouse over the row.

Pagination Subtemplate The Pagination Subtemplate section contains attributes for editing the Pagination Template, Next Page Template, Previous Page Template, Next Set Template, and Previous Template. Pagination Subtemplates support the substitution strings `#PAGINATION_NEXT#`, `#PAGINATION_NEXT_SET#`, `#PAGINATION_PREVIOUS#` and `#PAGINATION_PREVIOUS_SET#`. Table 7–12 describes these templates.

Table 7–10 Pagination Subtemplate Attributes

Pagination Subtemplate Attribute	Description
Pagination Template	Applies to the entire pagination subtemplate. For example: `<span class="instructiontext">#TEXT#</span>` You can use the substitution string `#TEXT#` to specify where you want the pagination subtemplate to display. Use the other Pagination Subtemplate attributes to modify individual items.
Next Page Template	Enter HTML to modify how the Next Page portion of the pagination subtemplate appears. For example: `<a href="#LINK#">next</a>`
Previous Page Template	Enter HTML to modify how the Previous Page portion of the pagination subtemplate appears. For example: `<a href="#LINK#">previous</a>`
Next Set Template	Enter HTML to modify how the Next Set portion of the pagination subtemplate appears. For example: `<a href="#LINK#">next set</a>`
Previous Set Template	Enter HTML to modify how the Previous Set portion of the pagination subtemplate appears. For example: `<a href="#LINK#">previous set</a>`

Comments Use this attribute to record developer comments.

Report Column Template Attributes for Named Column Templates

This section describes specific sections of the Report Column Template page for Named Column Templates.

Identification **Template Name** identifies the name of the template. Use the **Translatable** check box to indicate the template contains text strings that require translation. **Template Type** indicates the type of template. Named Column templates reference column names in the template. Generic Column Templates reference the `#COLUMN_VALUE#` substitution string in the template.

Theme indicates the theme to which the template is a member. **Template Class** identifies a specific use for the template. When you switch to a new theme, all templates in one theme are mapped to corresponding templates in another theme. Application Builder accomplishes this template mapping through the assignment of a template class.

Subscription Use Subscription to apply an existing template to the current application. When you select an existing template, you become a subscriber to that template.

To load a new copy of a master template, click Refresh.

Row Templates Row templates define the look of each column. You can define up to four row templates, each of which can be conditional.

In each Row Template, you define the look of each row. Row Templates support the substitution strings described in Table 7–11.

Table 7–11 Row Template Substitution Strings

Substitution String	Description
`#ALIGNMENT#`	Determines the row alignment. Specified by the user.
`#COLCOUNT#`	Defines the count of the number of columns.
`#COLNUM#`	Defines the current column number.
`#COLUMN_HEADER#`	Defines the column header.
`#COLUMN_VALUE#`	Replaced with the value of the column.
`#ROWNUM#`	Specifies the current row number.

By creating conditions, you can create a report that displays rows differently depending on whether or not the specified condition is met. To specify a row template be used conditionally, select a condition type from the Column Template Condition list. Valid values include:

- **Use Based on PL/SQL Expression.** Conditionally formats columns based on data in that row.
- **Use for Even Numbered Rows.** Conditionally formats even numbered rows.
- **Use for Odd Numbered Rows.** Conditionally formats odd numbered row.

If you select **Use Based on PL/SQL Expression**, the next step is to enter a PL/SQL expression in the Column Template Expression field. For example, the following expression displays a value in bold if the value is greater than 2000:

```
#SAL# > 2000
```

Note that you could also use the substitution string `#ROWNUM#`. For example:

```
#ROWNUM# > 2000
```

Column Headings Use this template to add color to each column header cell. The text of this attribute must include help to indicate where the cell heading text should be colorized. If you do not enter a Column Heading Template, then a default column header template is applied. If you do not want any column headings, then enter `OMIT`.

For example:

```
<th #ALIGNMENT#>#COLUMN_HEADER#</th>
```

Before first and after last row text In **Before Rows**, enter HTML that displays once at the beginning of a report template. Opening an HTML table is a common use of this attribute, as shown in the following example:

```
<table>
```

You can identify column headers using the syntax `#1#`, `#2#`, `#3#`. For example:

```
<th>#1#</th><th>#2#</th><th>#3#</th>
```

You can include pagination above a report by including the substitution string `#TOP_PAGINATION#`. This substitution string generates HTML that starts with an opening `<tr>` tag and ends with a closing `</tr>` tag. For example, to include an open table tag and `#TOP_PAGINATION#` substitution string you would enter the following:

```
<table>#TOP_PAGINATION#
```

You can also include the substitution string `#CSV_LINK#` to include support for exporting your report to CSV format, a format compatible with most spreadsheet programs.

Use **After Rows** to specify text that should display after the last row. A common use of this attribute is to close the HTML table tag. For example:

```
</table>
```

The After Rows attribute supports the following substitution strings:

- `#PAGINATION#`

  Replaced with a pagination attribute.
- `#COLCOUNT#`

  Substituted at run time with the number of columns defined in the report.

Row Highlighting Use **Background color for checked row** to control the background color of a report row when the row selector is checked. Use **Background color for current row** to control the background color of a report row when the user moves the mouse over the row.

Pagination The Pagination section contains attributes for editing the Pagination Template, Next Page Template, Previous Page Template, Next Set Template, and Previous Template. Pagination Subtemplates support the substitution strings `#PAGINATION_NEXT#`, `#PAGINATION_NEXT_SET#`, `#PAGINATION_PREVIOUS#` and `#PAGINATION_PREVIOUS_SET#`. Table 7–12 describes these templates.

Table 7–12 Pagination Subtemplate Attributes

Pagination Subtemplate Attribute	Description
Pagination Template	Applies to the entire pagination subtemplate. For example: `<span class="instructiontext">#TEXT#</span>` You can use the substitution string `#TEXT#` to specify where you want the pagination subtemplate to display. Use the other Pagination Subtemplate attributes to modify individual items.

Table 7–12 (Cont.) Pagination Subtemplate Attributes

Pagination Subtemplate Attribute	Description
Next Page Template	Enter HTML to modify how the Next Page portion of the pagination subtemplate appears. For example: `<a href="#LINK#">next</a>`
Previous Page Template	Enter HTML to modify how the Previous Page portion of the pagination subtemplate appears. For example: `<a href="#LINK#">previous</a>`
Next Set Template	Enter HTML to modify how the Next Set portion of the pagination subtemplate appears. For example: `<a href="#LINK#">next set</a>`
Previous Set Template	Enter HTML to modify how the Previous Set portion of the pagination subtemplate appears. For example: `<a href="#LINK#">previous set</a>`

Comments Use this attribute to record developer comments.

About Using JavaScript in Column Templates

You can conditionally display HTML depending upon values in the database using JavaScript. The following example displays an HTML row only if the `GROUP_DESC` query column is not null.

```
<script language="javascript">
IF ( "#GROUP_DESC#" != "" )
document.writeln( "<TR>;
<TD BGCOLOR=#336699>;</TD>
</TR>
</TR>
<TD>#GROUP_DESC#</TD>
          </TR>" );
</TR>" );
```

See Also:

- Online Help for information about using specific sections of the Edit Report Template page
- "Customizing Regions" on page 7-2

Optimizing a Page for Printing

You can optimize a page for printing by creating a specific Print Mode template and specifying that template in the User Template Defaults section of the Edit Application Attributes page. Generally, a Print Mode template optimizes a page for printing. For example, this template might:

- Not display tabs or navigation bars
- Have items display as text instead of as form elements

If the theme you select does not include a printer friendly template, you can create a Print Mode template by creating a new page template.

See Also: "Creating a New Template" on page 7-19

Topics in this section include:

- Setting a Print Mode Template for an Application
- Using f?p Syntax to Toggle to Print Mode

Setting a Print Mode Template for an Application

You enable your Print Mode template by selecting it in Default Templates by the Component section of the Define Theme page.

To enable Print Mode mode:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click **Shared Components**.
4. Under User Interface, select **Themes**.

   The Themes page appears.
5. In the Themes list, click the theme name.
6. Make a new selection from the Printer Friendly Page.
7. Click **Apply Changes**.

See Also: "Changing Default Templates in a Theme" on page 7-10

Using f?p Syntax to Toggle to Print Mode

Once you create a Print Mode template and select it as an application attribute, you can use `f?p` syntax to toggle to Print Mode. Including the ninth `f?p` syntax argument (`PrinterFriendly`) renders the page in printer friendly mode (optimize printed output). For example, you could include this argument when coding a link, or a creating navigation bar icon.

See Also: "Using f?p Syntax to Link Pages" on page 3-11

Using Custom Cascading Style Sheets

A cascading style sheet (CSS) provides a way to control the style of a Web page without changing its structure. When used properly, a CSS separates visual attributes such as color, margins, and fonts from the structure of the HTML document. Oracle Application Express includes themes that contain templates that reference their own CSS. The style rules defined in each CSS for a particular theme also determine the way reports and regions display.

Topics in this section include:

- Uploading Cascading Style Sheets
- Referencing an Uploaded Cascading Style Sheet in the Page Template

Uploading Cascading Style Sheets

You upload cascading style sheets to your database user account (or workspace) using the Cascading Style Sheet Repository. Uploaded cascading style sheets (CSS) are available to any application created in your database user account. The cascading style sheets are written to the file system, so you can reference them in your HTML source code.

To upload cascading style sheets:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click **Shared Components**.

   The Shared Components page appears.
4. Under Files, select **Cascading Style Sheets**.

   The CSS Repository appears.
5. To upload a new CSS, click **Create** and follow the on-screen instructions.
6. To edit an existing CSS, select the name.
7. To download an existing CSS, click the **Download** icon.

About the Cascading Style Sheets Page

Once you upload a CSS to the CSS Repository, you control how the page displays by making a selection from the View list. Available options include:

- **Icons** (the default) displays each CSS as a large icon. To edit a CSS, click the appropriate icon.
- **Details** displays each CSS as a line in a report. To edit a CSS, click the appropriate name.

Referencing an Uploaded Cascading Style Sheet in the Page Template

You can reference an uploaded cascading style sheet within the Header section of the page template. You use the Header section to enter the HTML that makes up the <HEAD> section of the HTML document.

To reference an uploaded cascading style sheets:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click **Shared Components**.
4. Under User Interface, select **Themes**.

   The Themes page appears.
5. On the Tasks list, click **View Templates**.
6. Select the name of the page template you want to edit.
7. Use a `<link>` tag within the head section to reference the appropriate style sheet.

   To reference an uploaded file that is associated with a specific application, you would use the substitution string `#APP_IMAGES#`. For example:

   ```
   <html>
   <head>
   ```

```
    <title>#TITLE#</title>
    #HEAD#
    <link rel="stylesheet" href="#APP_IMAGES#sample2.css" type="text/css">
</head>
...
```

To reference an uploaded file that is associated with a specific database user account (or workspace), you would use the substitution string `#WORKSPACE_IMAGES#`. For example:

```
<html>
<head>
    <title>#TITLE#</title>
    #HEAD#
    <link rel="stylesheet" href="#WORKSPACE_IMAGES#sample3.css"
type="text/css">
</head>
...
```

See Also: "Uploading Cascading Style Sheets" on page 7-44, "Creating a New Template" on page 7-19, "Editing Templates" on page 7-20, "Page Templates" on page 7-28, "APP_IMAGES" on page 3-15, and "WORKSPACE_IMAGES" on page 3-23

Managing Images

You can reference images within your application by uploading them to the Image Repository. When you upload an image, you can specify whether is it available to all applications or a specific application.

Topics in this section include:

- Uploading Images
- Referencing Images
- Editing Image Attributes
- Deleting an Image

Uploading Images

You upload images to your database user account (or workspace) using the Image Repository.

To upload images to your database user account:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click **Shared Components**.

   The Shared Components page appears.
4. Under Files, select **Images**.

   The Image Repository appears.
5. To upload a new image, click **Create**.
6. On the Create Image page, specify the following:

   a. Application - Select **No Application Associated** to make the image available to all applications within the database user account, or select a specific application ID.
   b. Upload New Image - Click **Browse** to identify a file to upload.
   c. Note - Enter details that describe the image.
7. Click **Upload**.

Referencing Images

You can reference images in your application by referencing the substitution string `#IMAGE_PREFIX#` or including a fully qualified URL to the image.

Topics in this section include:

- Verifying the Prefix for the Virtual Image Directory
- Referencing an Image Using #IMAGE_PREFIX#
- Referencing Images Using a Fully Qualified URL

Verifying the Prefix for the Virtual Image Directory

When you install Application Builder, the installer creates a virtual directory for images. This virtual directory points to the actual path on the file system that contains uploaded images. By default, you reference this virtual directory using the prefix:

```
/i/
```

When you first create an application, you need to verify this prefix on the Edit Application Attributes page:

To verify the Image Prefix for an application:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application home page, click Edit Attributes.
4. Click **Edit Standard Attributes**.
5. When the Edit Application Attributes page appears, locate the Image Prefix field.

   By default, this attribute is defined as `/i/`. Contact your administrator for information about the name of this virtual directory for your installation.

Referencing an Image Using #IMAGE_PREFIX#

When you embed an image in static text (for example, in page headers, region headers, or footers), you can reference the image using the substitution string `#IMAGE_PREFIX#`. For example, to reference the image `go.gif`, you would use the following syntax:

```
<img src="#IMAGE_PREFIX#go.gif">
```

See Also: "Built-in Substitution Strings" on page 3-13, "IMAGE_PREFIX" on page 3-19, "APP_IMAGES" on page 3-15, and "WORKSPACE_IMAGES" on page 3-23

Referencing Images Using a Fully Qualified URL

Alternatively, you can also reference an image using a fully qualified URL. For example:

```
<img
src="http://g-images.amazon.com/images/G/01/associates/navbar2000/logo-no-border(1
).gif">
```

Editing Image Attributes

When you edit image attributes you can add notes that describe an image or change the associated application. However, you cannot change the actual image. To change an image, delete it and then upload it again.

> **See Also:** "Deleting an Image" on page 7-47 and "Uploading Images" on page 7-45

To edit images attributes:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click **Shared Components**.

   The Shared Components page appears.
4. Under Files, select **Images**.

   The Image Repository appears.
5. Use the following controls to filter the view:
   a. Image - Enter text to search for an image name or notes describing the image. Select whether to search for All Images, Workspace Images, or Application Images.
   b. View - Select one of the following:
      – **Icons** (the default) displays each image as a large icon.
      – **Details** displays each image as a line in a report.
   c. Click **Go**.
6. Select an image.

   The Edit Image Attributes page.
7. From Application, select **No Application Associated** to make the image available to all applications within the database user account, or select a specific application ID.
8. In Notes, enter details that describe the image.
9. Click **Apply Changes**.

Deleting an Image

To delete an image:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.

3. Click **Shared Components**.

   The Shared Components page appears.
4. Under Files, select **Images**.

   The Image Repository appears.
5. Use the following to filter the view:
   a. Image - Enter text to search for an image name or notes describing the image. Select whether to search for All Images, Workspace Images, or Application Images.
   b. View - Select one of the following:
      – **Icons** (the default) displays each image as a large icon.
      – **Details** displays each image as a line in a report.
   c. Click **Go**.
6. Select an image.
7. Click **Delete**.

Managing Static Files

You can upload static files to your workspace (or database user account) using the Static File Repository.

Topics in this section include:

- Uploading Static Files
- Editing an Uploaded File
- Downloading an Uploaded File
- Deleting an Uploaded File

Uploading Static Files

To upload static files:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click **Shared Components**.

   The Shared Components page appears.
4. Under Files, select **Static Files**.

   The Static Files Repository appears.
5. To upload a file, click **Create**.
6. Follow the on-screen instructions.

Editing an Uploaded File

You may edit static files smaller than 30,000 bytes by selecting the file name. Otherwise, you must edit the file offline and upload it again.

To edit a static file smaller than 30,000 bytes:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click **Shared Components**.

   The Shared Components page appears.
4. Under Files, select **Static Files**.

   The Static Files Repository appears.
5. Use the following to filter the view:
   a. Static File - Enter text to search for a file name or notes describing the file.
   b. View - Select one of the following:
      – **Icons** (the default) displays each file as a large icon.
      – **Details** displays each file as a line in a report.
   c. Click **Go**.
6. Select a file.
7. To edit or add notes, edit the Notes field.
8. Click **Apply Changes**.

Downloading an Uploaded File

To download an uploaded file:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click **Shared Components**.

   The Shared Components page appears.
4. Under Files, select **Static Files**.

   The Static Files Repository appears.
5. From View, select **Details** and click **Go**.
6. Select the **Download** icon adjacent to the appropriate file.

Deleting an Uploaded File

To delete an uploaded static file:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click **Shared Components**.

   The Shared Components page appears.
4. Under Files, select **Static Files**.

   The Static Files Repository appears.
5. Use the following to filter the view:
   a. Static File - Enter text to search for a file name or notes describing the file.
   b. View - Select one of the following:

   - **Icons** (the default) displays each file as a large icon.
   - **Details** displays each file as a line in a report.

   c. Click **Go**.

6. Select a file.
7. Click **Delete**.

Creating a Multiple Column Layout

A region is an area of a page that uses a specific template to format HTML content. You use regions to group page controls. To create a multiple column layout, you create two regions that display in adjacent cells of the same table.

You can create a multiple column layout by either:

- Manually creating the two adjacent regions
- Defining a page template that contains a multiple column table

Topics in this section include:

- Creating Regions in Multiple Columns
- Creating a Multiple Column Page Template

Creating Regions in Multiple Columns

You create new regions using the Create Region Wizard. To create a two column page, you create two regions. Oracle Application Express replaces the `#BOX_BODY#` substitution string within a two column table and displays the regions in two separate cells.

To create a two-column page by creating regions:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Select a page.

   The Page Definition appears.
4. Create the first region:
   a. Under Regions, click **Create**.

      The Create Region Wizard appears.

   b. Select a region type.

   c. From the Column field, select **1**.

   d. Follow the on-screen instructions.
5. Create the second region:
   a. Under Regions, click **Create**.

      The Create Region Wizard appears.

   b. Select a region type.

   c. From the Column field, select **2**.

   d. Follow the on-screen instructions.

Creating a Multiple Column Page Template

Page templates define the appearance of individual pages, including the placement of page controls and components. Each page template is divided into three sections: Header, Body, and Footer. The most basic template must include the substitution string `#BOX_BODY#` in the Body attribute. When the page is rendered, the Application Express engine replaces `#BOX_BODY#` with HTML to display the regions on that page.

You can create a multiple column page by defining a page template that contains a multiple column table. You then explicitly place regions within specific table cells.

The following example demonstrates how to create a two-column page and specify a region position using the `#REGION_POSITION_XX#` substitution string in each column. You would enter this code in the Body section of the page-level template.

```
<body #ONLOAD#>
  #FORM_OPEN#
  <table style="width:100%">
    <tr>
      <td style="width:50%;padding:5px;">#REGION_POSITION_01#</td>
      <td style="width:50%; border-left:2px #bbbbbb dashed; padding:5px;">#REGION_
POSITION_02#</td>
    </tr>
  <br />
  #BOX_BODY#
  #FORM_CLOSE#
</body>
```

Once you create this page-level template, the newly defined positions would be available as Display Point options when you run the Create Region Wizard.

Rendering HTML Using Custom PL/SQL

If you need to generate specific HTML content not handled by Oracle Application Express forms, reports, and charts, you can use the region type PL/SQL. To generate HTML in this type of region, you need to use the PL/SQL Web Toolkit. You can reference session state using bind variable syntax. Keep in mind that when you generate HTML in this way you do not get the same consistency and control provided with templates.

See Also:

- *Oracle Database Application Developer's Guide - Fundamentals* for information about developing Web applications with PL/SQL
- *Oracle Database PL/SQL Packages and Types Reference* for information about htp packages

To give you more control over HTML dynamically generated within a region, you can use PL/SQL. For example, to print the current date you could create a region with the following source:

```
htp.p(TO_CHAR(SYSDATE,'Day Month DD, YYYY'));
```

This next example accesses tables:

```
DECLARE
   l_max_sal NUMBER;
BEGIN
   SELECT max(sal) INTO l_max_sal FROM emp;
   htp.p('The maximum salary is: '||TO_CHAR(l_max_sal,'999,999.00'));
END;
```

8

Understanding Application Administration

As you develop applications in Application Builder, you can mange your application development environment on the Application Administration page.

This section contains the following topics:

- About the Application Administration Page
- About the Manage Services Page
- Managing Session State and User Preferences
- Managing Log Files
- Disabling PL/SQL Program Unit Editing
- Managing Application Models
- Managing Application Express End Users
- Monitoring Activity
- How to Send Email from an Application

About the Application Administration Page

Use the Application Administration page to mange your application development environment. The Application Administration page contains the following icons:

- **Manage Services** links to the Mange Services page. Use this page to manage session state, log files, preferences, and application models. See "About the Manage Services Page" on page 8-2.
- **Manage Application Express Users** links to the Manage Application Express Users page. Use this page to manage Application Express user accounts and user groups. See "Managing Application Express End Users" on page 8-11.
- **Monitor Activity** links to the Monitor Activity page. Use this page to monitor changes to page views and entire applications. See "Monitoring Activity" on page 8-15.
- **Email Configuration** links to the Email Configuration page. Use this page to configure email environment settings as well as manage the email log and email queue. See "How to Send Email from an Application" on page 8-16.

A Tasks list displays on the right side of the Application Administration page and displays the following link:

- **About Application Express** links to an About page that lists basic product information. See "About Application Express" on page 8-2.

This section contains the following topics:

- Accessing Application Administration
- About Application Express

Accessing Application Administration

Use the Application Administration page to mange your application development environment.

To access the Application Administration page:

1. On the Database Home Page, click the **Application Builder** icon.
2. On the Tasks list, click **Administration**.

   The Application Administration appears.

About Application Express

The About Application Express page lists basic product information.

To access the About Application Express page:

1. On the Database Home Page, click the **Application Builder** icon.
2. On the Tasks list, click **About Application Express**.

   The About Application Express page appears.

The following Oracle Application Express product information displays:

- Product build
- Schema compatibility
- Last DDL time
- Host schema
- Application Owner
- Workspace ID
- Current user
- Language Preference
- Current Time (on server)

About the Manage Services Page

The Manage Services pages page contains the following icons:

- **Session State**. Use the Session State page to purge session state or user preferences. See "Managing Session State and User Preferences" on page 8-3.
- **Manage Logs**. Use the Manage Logs page to purge the Developer Activity and External Count Clicks log. See "Managing Log Files" on page 8-8.
- **Preferences**. Use the Preferences page to disable and enable application preferences such as PL/SQL program unit editing in Object Browser. See "Disabling PL/SQL Program Unit Editing" on page 8-10.
- **Application Models**. Use the Application Models page to search for and delete application models. See "Managing Application Models" on page 8-10.

This section contains the following topic:

- Accessing the Manage Services Page

Accessing the Manage Services Page

To access the Manage Services Page:

1. On the Database Home Page, click the **Application Builder** icon.
2. On the Tasks list, click **Manage Services**.

   The Manage Services page appears.

Managing Session State and User Preferences

A session is a logical construct that establishes persistence (or stateful behavior) across page views. Each session is assigned a unique ID which the Application Express engine uses to store and retrieve an application's working set of data (or session state) before and after each page view. Sessions persist in the database until purged.

Topics in this section include:

- Managing Session State
- Managing User Preferences

See Also: "Understanding Session State Management" on page 3-4

Managing Session State

A session establishes persistence (or stateful behavior) across page views. You view and purge session state for the current session, purge existing sessions by age, or first review session details on the Session State page and then purge.

Topics in this section include:

- Viewing and Purging Session State for the Current Session
- Purging Sessions by Age
- Viewing Session Details Prior to Removing Session State

Viewing and Purging Session State for the Current Session

To view session state for the current session:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Manage Services**.
3. Click **Session State**.
4. When the Session State page appears, click **Current preferences and session state with an option to purge**.
5. To view information about the current session, click **View Session State**.
6. To reset the session state for the current session, click **Purge Session State**.

Purging Sessions by Age

To purge existing session by age:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Manage Services**.
3. Click **Session State**.
4. Select **Purge sessions by age**.
5. Make a selection from Sessions older than.
6. Click one of the following buttons:
   - **Report Session** generates a report detailing the total number of sessions for the database user account, the number of users, and the number of old sessions.
   - **Purge Sessions** purges existing sessions by age.

   See Also: "Viewing Session State" on page 3-5

Viewing Session Details Prior to Removing Session State

You can determine whether to remove existing sessions by first reviewing session details on the Session State page.

To view session details prior to removing session state:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Manage Services**.
3. Click **Session State**.
4. Select **Recent sessions with drill down to session details**.
5. To narrow the results, select a time increment, specify a user, and click **Go**.
6. To view session details, select the session ID.

   The Session Details page appears.
7. Click one of the following buttons:
   - **Remove Session**. Removes the record of the session from the `SESSIONS` table along with session state (including collections data) associated with the session.

     Any user using a session that is removed will no longer be able to use the session and will be prompted to re-authenticate in most situations upon their next page request. This option could be used by administrators who might have a need to make sure a certain a specific user could no longer access to an Oracle Application Express applications.
   - **Remove State**. Clears the session data from the session state tables (including collections data) but does not remove the session record. Removing a session is a good approach for developers during debugging.

     This is the equivalent of clearing session state for the current session using the Clear Cache argument value `SESSION` in the f?p URL. This option might be used by developers during debugging.

   See Also: "Clearing Cache for the Current User Session" on page 3-9 and "Debugging an Application" on page 10-1

Managing User Preferences

You can use preferences to store values for a specific Application Express user across distinct sessions. Once set, these preferences can removed programatically or manually. You can set user preferences by creating a page process, by the calculation of a preference Item Source Value, or programatically using a PL/SQL API.

Topics in this section include:

- Viewing and Resetting Preferences for the Current User
- Viewing Preferences for Users
- Setting User Preferences
- Removing User Preferences Programatically
- Resetting User Preferences Using a Page Process
- Purging Preferences for a Specific User

Viewing and Resetting Preferences for the Current User

To manage user preferences for the current user:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Manage Services**.
3. Click **Session State**.
4. When the Session State page appears, click **Current preferences and session state with an option to purge**.
5. To view preferences for the current user, click **View Preferences**.
6. To reset user preferences for the current user, click **Reset Preferences**.

Viewing Preferences for Users

You view preferences for a specific user on the Purge Preferences report.

To view the Purge Preferences report:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Manage Services**.
3. Click **Session State**.
4. On the Session State page, select **Preferences by user**.

   The Preferences by Users page appears.
5. Specify a user and click **Go**.

Setting User Preferences

You can set user preferences within your application through the creation of a page process, by creating a preference item, or programatically.

Topics in this section include:

- Setting User Preferences Using a Page Process
- Setting the Source of an Item Based on a User Preference
- Setting User Preferences Programatically

Setting User Preferences Using a Page Process To set user preference values by creating a page process:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Page Processes, click the **Create** icon.

   The Create Page Process Wizard appears.
3. For the process category, select **Session State**.
4. For the process type, select one of the following:
   - **Set Preference to value of item**
   - **Set Preference to value of item if item is not NULL**
5. Specify a process name, sequence, and processing point.
6. Specify the preference value in the field provided using the format:

   ```
   PreferenceName:Item
   ```
7. Click **Page Items** to see a list of available items.
8. Follow the on-screen instructions.

Setting the Source of an Item Based on a User Preference You can set the source of an item based on a user preference by defining the item source type as Preference.

To define the source of item based on a user preference:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Item, click the **Create** icon.

   The Create Item Wizard appears.
3. For the item type, select **Hidden**.
4. Specify the Item Name, sequence, and region.
5. From the Item Source list, select **Preference**.
6. In Item Source Value, enter the name of the preference.
7. Click **Create Item**.

Setting User Preferences Programatically To set or reference user preferences programatically, you must use a PL/SQL API. User-level caching is available programmatically. You can use the `set_preference` function to set a user level preference called `NAMED_PREFERENCE`. For example:

```
HTMLDB_UTIL.SET_PREFERENCE(
 p_preference=>'NAMED_PREFERENCE',
 p_value =>:ITEM_NAME);
```

You can reference the value of a user preference using the function `GET_PREFERENCES`. For example:

```
NVL(HTMLDB_UTIL.GET_PREFERENCE('NAMED_PREFERENCE'),15)
```

In the previous example, the preference would default to the value 15 if the preference contained no value.

> **See Also:** "GET_PREFERENCE Function" on page 15-18 and "SET_PREFERENCE Procedure" on page 15-29

Removing User Preferences Programatically

To remove user preferences programatically, you must use a PL/SQL API. You can use the `REMOVE_PREFERENCE` procedure to remove a user level preference called `NAMED_PREFERENCE`, for example:

```
HTMLDB_UTIL.REMOVE_PREFERENCE(
p_preference=>'NAMED_PREFERENCE',
p_value =>:ITEM_NAME);
```

Resetting User Preferences Using a Page Process

You can reset user preferences by creating a page process and selecting the process type Reset Preferences.

To reset user preferences using a page process:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Under Page Processes, click the **Create** icon.

   The Create Page Process Wizard appears.
3. For the process category, select Session State.
4. From Type, select **Reset Preferences**.
5. Specify a process name, sequence, and process point.
6. Follow the on-screen instructions

Purging Preferences for a Specific User

You can purge preferences for a specific user on the Purge Preferences page.

To purge preferences for a specific user:

1. On the Database Home Page, click the **Application Builder** icon.

2. From the Administration list, select **Manage Services**.
3. Click **Session State**.
4. On the Session State page, select **Purge preferences by user**.

   The Purge Preferences page appears.
5. Select a specific user and click **Report**.

   A report appears at the bottom of the page.
6. To purge the displayed user preferences, click **Purge User Preferences**.

Managing Log Files

Log entries older then one month are automatically deleted. You can also manually purge log files on the Manage Logs page.

To purge log files, you must have a database account that has been granted a DBA role.

Topics in this section include:

- Truncating Page Views
- Purging the External Clicks Log
- Purging the Builder Activity Log
- Deleting SQL History Logs
- Deleting SQL Script Execution Logs

Truncating Page Views

Page Views track user activity for an application. Application developers enable logging within their application on the Edit Application Attributes page.

Note: "Configuring Standard Application Attributes" on page 4-6

The Application Express engine actually uses two logs to track user activity. At any given time, one log is designated as current. For each rendered page view, the Application Express engine inserts one row into the log file. A log switch occurs at the interval listed under **Log Interval**. At that point, the Application Express engine removes all entries in the noncurrent log and designates it as current.

To truncate the activity logs manually:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Manage Services**.
3. Select **Manage Logs**.

   The Manage Logs page appears.
4. Select **Page Views**.
5. If prompted, enter the appropriate administrator username and password and click **Login**.

   The entries in the current logs display.
6. Click either **Truncate Log 1** or **Truncate Log 2**.

See Also: "Monitoring Activity" on page 8-15

Purging the External Clicks Log

The External Clicks Log counts clicks from an application to an external site. You can implement this functionality using `COUNT_CLICK` procedure.

See Also: "COUNT_CLICK Procedure" on page 15-4

To purge the External Clicks Log:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Manage Services**.
3. Click **Manage Logs**.
4. Click **External Clicks**.
5. If prompted, enter the appropriate administrator username and password and click **Login**.
6. Click **Truncate Logs**.

Purging the Builder Activity Log

The Application Builder Activity log tracks changes to applications.

To purge the Application Builder Activity log:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Manage Services**.
3. Click **Manage Logs**.
4. Click **Builder Activity**.
5. If prompted, enter the appropriate administrator username and password and click **Login**.
6. Click **Truncate Logs**.

See Also: "Monitoring Activity" on page 8-15

Deleting SQL History Logs

The SQL History logs maintain a history of recent commands and scripts run in SQL Commands.

To delete SQL History logs:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Manage Services**.
3. Select **Manage Logs**.

   The Manage Logs page appears.
4. Click **SQL History**.
5. If prompted, enter the appropriate administrator username and password and click **Login**.

   The SQL History page appears, listing the current number of history entries.

6. To delete all entries, click **Truncate Logs**.

   See Also: "Using SQL Commands" on page 19-1

Deleting SQL Script Execution Logs

The SQL Scripts Execution Log maintains a history of script executions from SQL Scripts.

To delete SQL Scripts Execution Log:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Manage Services**.
3. Select **Manage Logs**.

   The Manage Logs page appears.
4. Click **SQL Script Execution**.
5. If prompted, enter the appropriate administrator username and password and click **Login**.

   The SQL Script Execution page appears, listing the current number of entries.
6. To delete all entries, click **Truncate Logs**.

   See Also: "Using SQL Scripts" on page 18-1

Disabling PL/SQL Program Unit Editing

By default, developers can change and compile PL/SQL source code when browsing database procedures, packages, and functions in Object Browser. You can disable PL/SQL program unit editing by selecting **Do not allow PL/SQL program unit editing** on the Preferences page.

To disable PL/SQL program unit editing:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Manage Services**.
3. Click **Preferences**.

   The Preferences page appears.
4. From PL/SQL Editing select one of the following:
   - **Allow PL/SQL program unit editing**
   - **Do not allow PL/SQL program unit editing**

   If you select **Do not allow PL/SQL program unit editing**, developers can still create and replace PL/SQL program units using scripts or SQL Commands.

   See Also: "Using SQL Commands" on page 19-1

Managing Application Models

Running the Create Application Wizard creates an application model. This model contains basic application property values, such as the application pages and page definitions, DML processes, and multi-row operation processes. When you create a new application, you can base it on an existing application model, making the creation process more productive.

See Also: "About the Create Application Wizard" on page 5-2

This section contains the following topics:

- Deleting an Application Model

Deleting an Application Model

You can remove unwanted application models on the Application Models page.

To delete an application model:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Manage Services**.
3. Click **Application Models**.

   The Application Models page appears.
4. To search for a model, enter a case insensitive query in the Model field and click **Go**.
5. Select the models you want to delete and click **Delete Checked**.

See Also: "Creating an Application" on page 5-1

Managing Application Express End Users

The simplest way to make an application available to other users is to create Application Express end users. Application Express end users do not have application development privileges and can only access applications that do not use an external authentication scheme. Use the Manage Application Express Users page to create new user accounts, manage existing user accounts, and change user passwords.

Topics in this section include:

- Editing Existing End User Accounts
- Creating New End User Accounts
- Changing an End User Password
- Managing Application Express Users Using Groups

See Also: "Exporting Application Express Users" on page 11-6, "About Publishing the Application URL" on page 11-14 and "About Application Express Account Credentials" on page 12-18 for information about implementing Application Express Authentication

Editing Existing End User Accounts

You edit existing end user accounts on the View User page.

To edit an existing a user account:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Manage Application Express Users**.

   The Manage Application Express Users page appears.
3. Click **View Users**.

The View Users page appears.

You control how the page displays by making a selection from the View list. Available options include:

- **Icons** (the default) displays each user as a large icon. To edit a user, click the appropriate icon.
- **Details** displays each user as a line in a report. To edit a user, click a user name.

4. Select a user.

   The Edit User page appears.
5. Under Edit User, edit the User Name or Email address.
6. Under Password, edit the current password by typing a new password in the Password and Confirm Password fields
7. Under User Groups, select an optional user group.

   You can use groups to restrict access to various parts of an application. Groups are primarily useful when using Application Express Authentication.
8. Click **Apply Changes**.

See Also: "Managing Application Express Users Using Groups" on page 8-13 and "Removing Users from a Group" on page 8-14

Creating New End User Accounts

You create new user accounts on the Create User page.

To create a new user account:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Manage Application Express Users**.

   The Manage Application Express Users page appears.
3. Click **Create User**.

   The Create User page appears.
4. Under User Identification, enter the appropriate information. Required fields are marked with a red asterisk (*).
5. Under User Groups, select an optional user group.

   You can use groups to restrict access to various parts of an application. Groups are primarily useful when using Application Express Authentication.
6. Click **Create User** or **Create and Create Another**.

See Also: "Managing Application Express Users Using Groups" on page 8-13 and "Removing Users from a Group" on page 8-14

Changing an End User Password

To change an end user password:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Manage Application Express Users**.

The Manage Application Express Users page appears.

3. Click **View Users**.

   The View Users page appears.

4. To search for an existing user, enter a query in the Find field and clicking **Go**.
5. Select a user.
6. Scroll down to Password.
7. Type a new password in the Password and Confirm Password fields and click **Apply Changes**.

Managing Application Express Users Using Groups

You can create groups to restrict access to various parts of an application. Keep in mind, however, that groups are not portable over different authentication schemes. Groups are primarily useful when using Application Express Authentication.

Topics in this section include:

- Creating a Group
- Editing an Existing Group Assignment
- Viewing Group Assignment Reports
- Adding Users to a Group
- Removing Users from a Group

> **See Also:** "About Application Express Account Credentials" on page 12-18 for information about implementing Application Express Authentication and "Managing Application Express End Users" on page 8-11

Creating a Group

To create a new group:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Manage Application Express Users**.

   The Manage Application Express Users page appears.

3. Click **Create Group**.

   The Create/Edit Group page appears.

4. Specify a group name, description, and click **Create Group**.

Editing an Existing Group Assignment

To edit an existing group assignment:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Manage Application Express Users**.

   The Manage Application Express Users page appears.

3. Click **User Group Assignments.**

   The User Group Assignments page appears.

4. Click the **Edit** icon adjacent to the Group Name.
5. Scroll down to User Groups, make a new selection, and click **Apply Changes**.

Viewing Group Assignment Reports

To view a report of user group assignments:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Manage Application Express Users**.

   The Manage Application Express Users page appears.
3. Click **User Group Assignments**.

   The User Groups Assignments report appears.

Adding Users to a Group

To add a user to a group:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Manage Application Express Users**.

   The Manage Application Express Users page appears.
3. Click **View Users**.

   The Existing Users page appears.
4. Select a user.

   The Edit User page appears.
5. Scroll down to User Groups.
6. Select a group from the Groups list.
7. Click **Apply Changes**.

Removing Users from a Group

To remove a user to a group:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Manage Application Express Users**.

   The Manage Application Express Users page appears.
3. Click **View Users**.

   The Existing Users page appears.
4. Select a user.

   The Edit User page appears.
5. Scroll down to User Groups.
6. Deselect the selected group in the Groups list.
7. Click **Apply Changes**.

Monitoring Activity

You can monitor changes to page views and entire applications by viewing reports on the Monitor page.

Topics in this section include:

- Accessing the Monitor Page
- Viewing Application Changes by Application
- Viewing Application Changes by Day
- Viewing Active Sessions

See Also: "Creating Custom Activity Reports Using HTMLDB_ACTIVITY_LOG" on page 13-10

Accessing the Monitor Page

To access the Monitor page:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Monitor Activity**.

   The Monitor page is divided into the following sections:

   - Page Views
   - Application Changes
   - Sessions
3. Select a report to review.

Viewing Application Changes by Application

To view application changes by developer:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Monitor Activity**.
3. Under Application Changes, select **By Application.**
4. Specify a time frame. Make a selection form the Time list and click **Go**.
5. To view additional details, select a user ID.

Viewing Application Changes by Day

To view application changes by day:

1. Navigate to the Database Home Page.
2. Click the **Administration** icon.
3. Click **Monitor Activity**.
4. Under Application Changes, select **By Day**.

   The Application Changes by Day page appears.
5. Select the appropriate report:

   - **Month View**. Offers a listing of application changes by day in a Calendar view.

- **Report**. Includes a report of application changes by day.
- **Line Chart**. Make a selection from the Developer list and click **Go**.
- **By Developer Report**. Specify a time frame by making a selection form the Time list and click **Go**. To view additional details, select a developer.

Viewing Active Sessions

The Active Sessions Report displays active sessions for the current workspace (or database user).

To view application changes by developer:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Monitor Activity**.
3. Under Sessions, select **Active Sessions.**
4. Select the session ID to view the Session Details page.

See Also: "What Is a Session?" on page 3-5

How to Send Email from an Application

This section describes how to send email from an Application Builder application. Before you can send email from an Application Builder application, you first need to configure your Email Environment Settings.

Topics in this section include:

- Configuring Email Environment Settings
- Sending Email from an Application
- Viewing the Email Log
- Viewing the Mail Queue

Configuring Email Environment Settings

To send email from an Oracle Application Express application, you must configure settings on the Email Environment Preferences page. To access the Email Environment Settings page, you must have a database account that has been granted a DBA role.

To configure Email Environment Preferences:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Email Configuration**.

   The Email Configuration page appears.
3. Click **Email Environment Settings**.
4. If prompted, enter the appropriate administrator username and password and click **Login**.
5. On the Email Environment Setting page:
   - **SMTP Host Address** - Enter the server address of the SMTP server. On installation, this will be set to `localhost`. If you are using another server as an SMTP relay, change localhost to that server's address.

- **For SMTP Host Port** - Enter the port the SMTP server listens to for email requests. By default, this setting will be set to 25 at the time of installation.

6. Click **Apply Changes**.

Sending Email from an Application

You can send an email from an Application Builder application by:

- Creating a background job to periodically send all mail messages stored in the active mail queue
- Calling the PL/SQL package `HTMLDB_MAIL`

Topics in this section include:

- Sending Email Using a Background Job
- Sending Email Manually by Calling HTMLDB_MAIL

See Also: "Configuring Email Environment Settings" on page 8-16

Sending Email Using a Background Job

Oracle Application Express stores unsent email messages in a table named `HTMLDB_MAIL_QUEUE. A DBMS_JOB` background process is automatically created when you install Oracle Application Express. This background process pushes the mail queue every 15 minutes.

The most efficient approach to sending email is to create a background job (using a `DBMS_JOB` package) to periodically send all mail messages stored in the active mail queue.

Sending Email Manually by Calling HTMLDB_MAIL

You can also send an email from an Oracle Application Express application by calling the PL/SQL `HTMLDB_MAIL` package. This package is built on top of the Oracle supplied `UTL_SMTP` package. Because of this dependence, in order to use `HTMLDB_MAIL`, the `UTL_SMTP` package must be installed and functioning.

See Also: *Oracle Database PL/SQL Packages and Types Reference* for more information about the UTL_SMTP package and "HTMLDB_MAIL" on page 15-32

`HTMLDB_MAIL` contains two procedures for manually sending email:

- Use the `HTMLDB_MAIL.SEND` procedure to manually send an outbound email message from your application
- Use `HTMLDB_MAIL` to deliver mail messages stored in `HTMLDB_MAIL_QUEUE`

Oracle Application Express stores unsent email messages in a table named `HTMLDB_MAIL_QUEUE`. You can deliver mail messages stored in this queue to the specified SMTP gateway by calling the procedure `HTMLDB_MAIL.PUSH_QUEUE`.

Oracle Application Express logs successfully submitted messages in the table `HTMLDB_MAIL_LOG` with the timestamp reflecting your server's local time.

The following UNIX/LINUX example demonstrates the use of the `HTMLDB_MAIL.PUSH_QUEUE` procedure using a shell script.

```
SQLPLUS / <<EOF
FLOWS_020100.HTMLDB_MAIL.PUSH_QUEUE;
```

```
DISCONNECT
EXIT
EOF
```

See Also: "HTMLDB_MAIL" on page 15-32 for information about using the `HTMLDB_MAIL`

Viewing the Email Log

The Email log records message header information and send date of successfully sent mail message.

To view the email log:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Email Configuration**.
3. Click **Manage Email Log**.
4. If prompted, enter the appropriate administrator username and password and click **Login**.

   The Email Log page appears.
5. To control the number of rows that display, make a selection from the Display list and click **Go**.
6. To delete all log entries, click **Truncate Log**.

Viewing the Mail Queue

Use the Manage Mail Queue page to monitor email messages in the mail queue.

To monitor messages in the mail queue:

1. On the Database Home Page, click the **Application Builder** icon.
2. From the Administration list, select **Email Configuration**.
3. Click **Manage Email Queue**.
4. If prompted, enter the appropriate administrator username and password and click **Login**.

   The Manage Email Queue page appears.
5. To send email messages, click **Send All Mail**.
6. To delete email messages, select the messages to be deleted and click **Delete**.

9

Managing User Interface Defaults

User interface defaults enables you to assign default user interface properties to a table, column, or view within a specified schema. When you create a form or report using a wizard, the wizard uses this information to create default values for region and item properties. Utilizing user interface defaults can save you valuable development time and has the added benefit of providing consistency across multiple pages in an application.

Because user interface defaults are associated with a table, you can use them with applications created using the form and report wizards.

This section contains the following topics:

- Viewing Tables or Views Utilizing User Interface Defaults
- Editing Column Attributes
- Comparing User Interface Defaults Across Applications
- About Exporting and Importing User Interface Defaults

See Also: "Leveraging Application Models and User Interface Defaults" on page 5-5

Viewing Tables or Views Utilizing User Interface Defaults

You can view tables or views utilizing user interface defaults by either navigating to the User Interface Defaults page or viewing the UI Defaults report in Object Browser.

Topics in this section include:

- Navigating to the User Interface Defaults Page
- Viewing the UI Defaults Report in Object Browser

See Also: "Managing Database Objects with Object Browser" on page 16-1

Navigating to the User Interface Defaults Page

You can access the User Interface Defaults page:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application, click **Shared Components**.

   The Shared Components page appears.

4. Under User Interface, select **User Interface Defaults**.

   The User Interface Defaults page appears.

   The current schema displays to the right of the breadcrumb menu.

5. To narrow the display, use the following controls at the top of the page and click **Go**:

   - **Table/View** - Enter a case insensitive query for a table or view name within the current schema.
   - **View** - Make a selection to filter the view:
     - **Icons** (default) displays each table or view as large icon.
     - **Details** displays each table or view as a line in a report, identifying the table or view name, the object type, and whether or not user interface defaults currently exist.
     - **Display** determines the number of items that display in the report.

6. Select a specific table or view by selecting the name.

   The Table Defaults page appears.

7. If no user interface defaults exist, click **Create User Interface Defaults**.

Viewing the UI Defaults Report in Object Browser

To view the User Interface Details Report in Object Browser:

1. Click the **Object Browser** icon on the Database Home Page.

   Object Browser appears.

2. From the Object list, select either **Tables** or **Views**.
3. From the Object Selection pane, select an object.
4. Select the **UI Defaults** tab.

   The User Interface Defaults report appears displaying the following information:

   - Column Name - Indicates the name of the column.
   - Label - Specifies the default label text for items in a form and the heading for columns in reports.
   - Report Sequence - Specifies the sequence of items in a report.
   - Report Display - Specifies how the column should be displayed in a report.
   - Tabular Form Display - Specifies how an item should display in a tabular form.
   - Form Sequence - Specifies the sequence of items in a form.
   - Form Display - Specifies how items in a form display.

5. To edit the user interface defaults, click **Edit**.

   The Table Defaults page appears.

6. If no user interface defaults exist, click **Create Defaults**.

Editing Column Attributes

You define user interface defaults for a specific column by editing column attributes.

To edit column attributes:

1. Navigate to the Table Defaults page as described in "Viewing Tables or Views Utilizing User Interface Defaults" on page 9-1.

   The Table Defaults page appears. The following information displays at the top of the page:

   - **Table/View Name** identifies the name of the selected table or view.
   - **Report Region Title** and **Form Region Title** become the default title for all report or form regions. These names are modified versions of Table/View Name in which the first letter is capitalized and any underscores are replaced with spaces.

   Column-level User Interface Defaults appear next. By default, a short report displays.

2. To view a complete report, click **Detailed Report**.
3. To edit select attributes for all displayed columns, click **Grid Edit**.
4. To edit a specific column, select the column name.

   The column defaults appear. Column defaults are divided into two pages:

   - Column Definition
   - List of Values

   The topics that follow describes how to edit specific attributes on these pages.

About the Column Definition

Column Definition is the default page that displays when you edit column attributes. The top of the page displays the selected schema, table or view name, and column name. Click **View Database Column Definition** to view details about a specific column.

Label Default

This attribute is used in report and forms. Use **Label** to specify default label text for items in a form and the heading for columns in reports.

Report Defaults

Available attributes include:

- **Display** - Indicates if the column displays in a report. The default is **Yes**.
- **Display Sequence** - Specifies the display sequence of items in a report. The default value is based on the column ID, which is based on the order of the columns in the table.
- **Display As** - Specifies how the column should be displayed in a report.
- **Mask** - Indicates if a mask should be applied against the data. This attribute is not applicable for character- based items.
- **Alignment** - Specifies report alignment (left, center, or right). If the column is a number, the default is **Right**. Otherwise, the default is **Left**.

- **Searchable** - Indicates whether or not the column should be searchable in reports. If the column is `VARCHAR2` or `CHAR`, the default is **Yes**. If not, the default is **No**.
- **Group By** - Indicates whether or not the column should be used for Group By and then the sequence of the grouping. The default is **Yes**.
- **Aggregate By** - Indicates whether or not the column should be used for aggregation in reports and charts.

Tabular Form Default

Use **Display As** to specify how an item should display in a tabular form.

Form Defaults

Available attributes include:

- **Display** - Indicates if the column displays in a form. The default is **Yes**.
- **Display Sequence** - Specifies the sequence of items in a form. The default is based on the column ID, which is based on the order of the columns in the table.
- **Display As** - Indicates how items in a form display. The default selection is **Text Field**.
- **Mask** - Indicates if a mask should be applied against the data in a form. Not used for character-based items.
- **Default Value** - Specifies the default value associated with this column.
- **Width** - Specifies the display width.
- **maxWidth** - Specifies the maximum string length a user is allowed to enter in this item.
- **Height** - Specifies the display height of an item.
- **Required** - Used to generate a validation in which the resulting item must not be null. If resulting item is not null, select **Yes**.
- **Help Text** - Becomes Item help. By default, this text is pulled from the column hint (if applicable).

About List of Values

You access the List of Values page by clicking the **List of Values** tab. The top of the page displays the selected schema, table or view name, and column name. Click **View Database Column Definition** to view details about a specific column.

The top of the page displays the selected schema, table or view name, and column name. Use the List of Values Type list to specify if the selected column will include a static or dynamic list of values. Once you select the type, you are prompted to enter either display value and return value pairs, or a list of values query.

About the Database Column Definition Report

You can view details about a specific column by accessing the Column Definition report. The Column Definition report displays the schema, table name, column name, data type, data length, and nullable, as well as any check constraints, primary and unique keys, and foreign keys that reference the column. A link to this report appears on both the Column Definition and List of Values pages.

To view the Column Definition report:

1. Navigate to the Table Defaults page as described in "Viewing Tables or Views Utilizing User Interface Defaults" on page 9-1.

   The Table Defaults page appears.
2. Select the column name.
3. Select **View Database Column Definition**.

Comparing User Interface Defaults Across Applications

Use the Compare Defaults report to monitor consistency in user interface design across all pages in a single application or multiple applications. Running the Compare Defaults report compares currently defined user interface defaults (or column attributes) against the item attributes set for forms, reports, and tabular forms.

> **See Also:** "Editing Column Attributes" on page 9-3

To run the Compare Defaults report:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application home page, click **Shared Components**.

   The Shared Components page appears.
4. Under User Interface, select **User Interface Defaults**.

   The User Interface Defaults page appears.
5. On the Tasks list, click **Comparison Report**.

   The current schema displays to the right of the breadcrumb menu.
6. Make selections from the following lists:
   - **Table/View** - Restricts the comparison to the selected table or view.
   - **Column** - Select a column in which to search for form, reports, and tabular forms.
   - **Display** - Select an attribute category.
   - **Application** - Select an application.
7. Click **Go**.

A report appears containing the following sections:

- Form Pages Referencing the Selected Column
- Report Regions Referencing the Selected Column
- Tabular Form Regions Referencing the Selected Column

About Exporting and Importing User Interface Defaults

You export user interface defaults in the same way you export any related application file. Exporting user interface defaults from one development instance to another involves the following steps:

1. Export the user interface defaults using the Export User Interface Defaults utility.
2. Import the exported file into the target Oracle Application Express instance.

3. Install the exported file from Export Repository.

When you export user interface defaults, all user interface defaults for the selected schema are exported to a single script. The file contains an API call to create table hints by making calls to the application PL/SQL API. You can use this file to import user interface defaults to another database and Oracle Application Express instance.

> **See Also:** "Exporting User Interface Defaults" on page 11-10 and "Importing Export Files" on page 11-11

10 Debugging an Application

This section describes approaches to debugging an application including viewing Debug Mode, enabling SQL tracing, viewing page reports, and how to manually remove a control or a component to isolate a problem.

This section contains the following topics:

See Also: "Application Builder Concepts" on page 3-1 and "Using Application Builder" on page 4-1

About Tuning Performance

For applications having a large number of concurrent users, maintaining optimal performance is critical. To optimize your application's performance, remember to utilize the following features:

- Use bind variables within your application whenever possible. You can reference session state values using bind variable syntax in SQL queries and application logic such as PL/SQL executed from processes and validations. Accessing session state using bind variables is the most efficient way to reference session state.
- Include a `#TIMING#` substitution string in the region footer so that you can view the timing of each region.

See Also:

- "About Bind Variables" on page 3-9
- "Using Substitution Strings" on page 3-13

Reviewing Session State

Many applications are based on data contained within application controls. For example, buttons can display conditionally based on a value stored in session state. You can view current session state for your application by clicking the Session link on the Developer Toolbar.

> **See Also:** "Using the Developer Toolbar" on page 4-18, "Viewing Session State" on page 3-5, "Managing Session State Values" on page 3-6, and "Managing Session State and User Preferences" on page 8-3

Accessing Debug Mode

Viewing a page in Debug Mode enables you to track what the Application Express engine is doing as it renders a page. You access Debug mode by clicking the **Debug** link in the Developer Toolbar.

> **See Also:** "Using the Developer Toolbar" on page 4-18

Debug Mode displays time codes that correspond to specific Application Express engine actions. This can be useful if you want to determine when the engine is setting session state. The Debug view also shows additional details about item names and computation and processing points. To exit Debug mode, click **No Debug** in the Developer Toolbar.

You can also use `f?p` syntax to run an application in Debug mode. Simply call the page and set the Debug argument to `YES`. For example:

```
f?p=100:1:&SESSION::YES
```

> **See Also:** "Using f?p Syntax to Link Pages" on page 3-11

Enabling SQL Tracing and Using TKPROF

Tracing your session can be a very effective way to debug an application. From a database perspective, each page request is a single database session. If you enable SQL tracing, then Oracle Application Express creates a temporary file you can then analyze using the TKPROF utility.

You enable SQL tracing in Oracle Application Express by using `f?p` syntax to set the argument `p_trace=YES`. For example, to trace the display of page 1 in application 100, you would use the syntax:

```
http:/.../f?p=100:1&p_trace=YES
```

To use the TKPROF utility:

- Navigate to the directory in which the trace file is created.
- Enter the following to view instructions about using TKPROF utility:

  ```
  tkprof help=yes
  ```

> **See Also:** "Using Application Tracing Tools" in *Oracle Database Performance Tuning Guide* for information about using the TKPROF program

Monitoring Application and Page Resource Use

Oracle Application Express facilitates the monitoring of resources used by applications and pages by calling the package `DBMS_APPLICATION_INFO`. Whenever the Application Express engine renders or processes a page, the module is set to `APEX` and includes the application ID and page ID. Once set, you can query the `V$SESSION` and `V$SQLAREA` views to monitor transactions.

Viewing Reports

When isolating an issue within a page, it is important to clearly understand the functions it is performing. To accomplish this goal, Application Builder includes a number of page and application reports.

To view page reports:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Click one of the following buttons at the top of the Page Definition:
   - **Event View** links to a report that details currently defined page controls and processes.
   - **Object References** displays a list of database objects referenced by the current page.
   - **History** displays a history of recently changed pages.

   See Also: "About Groups" on page 4-40, "About Objects" on page 4-39, and "About History" on page 4-39

To view application reports:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Tasks list, click **View Application Reports**.
4. Select the type of reports to view:
   - **Shared Components** reports offer information on common elements that can display or be applied on every page within an application. Report examples include Application Comments, Breadcrumb Entries, Database Object Dependencies, Lists of Values, Static Entries, and Messages.
   - **Page Components** reports offer detailed information on controls and logic that execute when the page is rendered (for example, branches, buttons, computations, items, and regions).
   - **Activity** reports offer details about developer activity within the current application. Available reports include Changes by Developer, Changes by Developer by Day, Chart of Changes by Developer, and Recent Changes.

See Also: "About the Database Object Dependencies Report" on page 4-50 and "About the Region Search Source Report" on page 4-50

Debugging Problematic SQL Queries

If your query does not seem to be running correctly, try running it in SQL Command Line (SQL*Plus) or in SQL Commands. Either approach will test your query outside the context of your application, making it easier to determine what the problem is.

See Also: "Using SQL Commands" on page 19-1 and "Using SQL Command Line" in *Oracle Database Express Edition 2 Day Developer Guide*

Removing Controls and Components to Isolate a Problem

If you have problems running a page, try removing controls and components one at a time. Using this approach, you can quickly determine which control or component may be the source of your problem. You can disable a control or component by selecting the Condition attribute Never.

To remove a control or component using conditional attributes:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. Select the name of the control or component you want to disable.

   The appropriate attributes page appears.
3. Scroll down to Condition Type and select **Never** from the Condition Type list.
4. Click **Apply Changes** and return to the Page Definition.
5. Try running the page again.
6. Continue to remove controls or components until the page runs correctly.

See Also: "About the Page Definition" on page 4-16, "Editing Page Attributes" on page 4-20, "Understanding Conditional Rendering and Processing" on page 3-2, and "Running a Page or Application" on page 5-11

11 Deploying an Application

This section describes how to deploy an application built within Application Builder.

This section contains the following topics:

> **See Also:** "Managing HTTP Access to the User Interface" on page 21-19, "Advanced Programming Techniques" on page 13-1 and "Managing Application Express End Users" on page 8-11

About the Oracle Application Express Application Development Life Cycle

When developing applications using Application Builder, you need to find a balance between two dramatically different development methodologies:

- Iterative, rapid application development
- Planned, linear style development

The first approach offers so much flexibility, you run the risk of never completing your project. In contrast, the second approach can yield applications that do not meet the needs of end users even if they meet the stated requirements on paper.

System Development Life Cycle Methodologies to Consider

The system development life cycle (SDLC) is the overall process of developing software using a series of defined steps. There are a number of SDLC models that work well for developing applications in Oracle Application Express.

The **SDLC waterfall** is probably the best known model. In this methodology, the development process is broken down into the following stages:

1. Project Planning
2. Requirements Definition

3. Design
4. Development
5. Integration and Testing
6. Installation and Acceptance
7. Maintenance

This methodology is referred to as a waterfall because the output from one stage is the input for the next stage. One of the primary problems with this approach is that it is assumed that all requirements can be established in advanced. Unfortunately, requirements often change and evolve during the development process.

The Oracle Application Express development environment enables developers to take a more iterative approach to development. Unlike many other development environments, creating prototypes is easy. With Oracle Application Express, developers can:

- Use built-in wizards to quickly design an application user interface
- Make protoypes available to users and gather feedback
- Implement changes in real time, creating new prototypes instantly

Other methodologies what work well with Oracle Application Express include:

- **Spiral** - This approach is actually a series of short waterfall cycles. Each waterfall cycle yields new requirements and enables the development team to create a robust series of prototypes.
- **Rapid application development (RAD) life cycle** - This approach has a heavy emphasis on creating a prototype that closely resembles the final product. The prototype is an essential part of the requirements phase. One disadvantage of this model is that the emphasis on creating the prototype can cause scope creep, Developers can lose sight of their initial goals in the attempt to create the perfect application.

About Deploying an Application to Another Instance

Deploying an application from one Oracle Application Express instance to another is a two step process:

1. Move the supporting database objects. Review the Database Dependencies report to determine what objects to move. See "About the Database Object Dependencies Report" on page 4-50.
2. Move the application definition and all associated files. See "How to Deploy an Application to Another Oracle Application Express Instance" on page 4.

This section contains the following topics:

- Deployment Options to Consider
- Deciding to Copy the Workspace
- Deciding to Copy the Database
- About the Application ID

Deployment Options to Consider

When you develop an application in Application Builder, you create the application within a specific database user account (or workspace). Each workspace has a unique ID and name. A common scenario is to create the application in a development instance and then deploy it to a production instance.

During the deployment process, you need to decide whether you want to use the existing application ID, the existing workspace, the existing database, or the existing Oracle HTTP Server, or create new ones. Deployment options to consider include:

1. **Create Application Express End Users.** The simplest way to deploy an application is to create Application Express end users and then send the URL and login information to users. This approach works well for applications with a small and tolerant user population. See "About Publishing the Application URL" on page 11-14 and "Managing Application Express End Users" on page 8-11.
2. **Same workspace and same schema.** Export and then import the application and install it using a different application ID. The approach works well when there are few changes to the underlying objects, but frequent changes to the application functionality.
3. **Different workspace and same schema.** Export and then import the application into a different workspace. This is an effective way to prevent a production application from being modified by developers.
4. **Different workspace and different schema.** Export and then import the application into a different workspace and install it using a different schema.
5. **Different database with all its variations.** Export and then import the application into a different Oracle Application Express instance and install it using a different schema and database.

Deciding to Copy the Workspace

When deciding whether or not to copy an existing workspace (or database user account) is a matter of preference. Keep in mind that the production version must have access to all the appropriate objects. For example, you might want to copy a workspace in the following situations:

- When the application subscribes to other objects within the workspace.
- When the application relies on Oracle Application Express authentication. Copying the workspace automatically migrates all the required user data.

Deciding to Copy the Database

When deciding whether or not to copy the database, remember that the schema against which the application runs must have access to the same objects as the development instance. The actual name of the schema is unimportant. You can change it during the import process.

See Also: "Importing Export Files" on page 11-11

About the Application ID

It is not necessary to have matching application IDs for a development version and production version of an application. In fact, as a best practice never hard code the application ID into your application. Instead, use the application alias (defined on the Edit Application Attributes page), or use a built-in substitution string (such as `APP_ID` and APP_ALIAS). Using a substitution string is the best approach because it enables you to change the application ID without affecting any application functionality.

See Also: "Name" on page 4-7 for information about defining an application alias and "Built-in Substitution Strings" on page 3-13 for information about using APP_ID and APP_ALIAS

How to Deploy an Application to Another Oracle Application Express Instance

Whether you want to move an application to another database user account or just make a copy of it, the deployment process involves the following steps:

1. Move the supporting database objects (if appropriate). Review the Database Dependencies report to determine what objects to move. See "About the Database Object Dependencies Report" on page 4-50.
2. Export the application definition and all related files. See "Exporting an Application and Related Files" on page 11-5.
3. Import the exported files into the target Oracle Application Express instance. See "Importing Export Files" on page 11-11.

   Note that if the target instance is a different schema, you also need to export and import any required database objects.
4. Install the exported files from Export Repository. See "Installing Export Files" on page 11-12.

You can import an application into your workspace regardless of the workspace in which it was developed.

About Managing Database Objects

Before you export an application and the appropriate related files, you need to determine if you also need to migrate the database objects referenced by the application. If you are unsure of which database objects to move, review the Database Object Dependencies report.

See Also: "About the Database Object Dependencies Report" on page 4-50

If the target schema is different from the schema used in the development environment, you will need to migrate the database objects referenced by the application. In many cases, this process can be as simple as using Oracle database export and import utilities to copy the application schema from the development environment to target instance. The following are two common scenarios where this approach will not work:

- When the object development schema refers to tablespaces to which the target instance schema does not have access
- When the development instance schema has sample data that you do not want to migrate to the target instance schema

See Also: "Loading Data" on page 20-2 and "Unloading Data" on page 20-3

Exporting an Application and Related Files

You export and import application definitions and all associated files using the Workspace, Application, CSS, Images, Script Files, Themes, and User Interface Defaults buttons located at the top the Export page.

Topics in this section include:

- Exporting an Application
- Exporting Application Express Users
- Exporting a Page in an Application
- Exporting Cascading Style Sheets
- Exporting Images
- Exporting Static Files
- Exporting Script Files
- Exporting Themes
- Exporting User Interface Defaults

See Also: "Importing Export Files" on page 11-11 and "Installing Export Files" on page 11-12

Exporting an Application

When you export a application, Oracle Application Express generates a text file containing PL/SQL API calls.

To export an application:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application Builder, click **Export/Import**.
4. When prompted to select a task, select **Export** and click **Next**.
5. From File Format, select how rows in the export file will be formatted:
   - Choose **UNIX** to have the resulting file contain rows delimited by line feeds.
   - Choose **DOS** to have the resulting file contain rows delimited by carriage returns and line feeds.
6. From Build Status Override, select one of the following:
   - **Run Application Only** - Developers can only run an application
   - **Run and Build Application** - Developers can both run and edit an application

   Selecting **Run Application Only** is an effective way to protect an application from modifications from other developers. Note that if you select **Run Application Only** you cannot set the argument `p_trace` to `Yes`.
7. Use the **As of** field to export your application as it was previously defined. Specify the number of minutes in the field provided.

   This utility uses the `DBMS_FLASHBACK` package. Because the timestamp to System Change Number (SCN) mapping is refreshed approximately every five minutes, you may have to wait that amount of time to locate the version for which you are looking. The time undo information is retained and influenced by the startup parameter `UNDO_RETENTION` (the default is three hours). However, this only influences the size of the undo tablespace. While two databases can have the same `UNDO_RETENTION` parameter, you will be able to go back further in time on a database with fewer transactions because it is not filling the undo tablespace, forcing older data to be archived.

8. Click **Export Application**.

In addition to exporting the actual application file, you may also need to export other related files such cascading style sheets, images, and script files.

See Also: "Enabling SQL Tracing and Using TKPROF" on page 10-2

Exporting Application Express Users

You can make an application available to other users by creating Application Express end users. When you export Application Express end users, Oracle Application Express creates an ASCII text SQL script of users and any defined user groups.

See Also: "Managing Application Express End Users" on page 8-11

To export Application Express end users:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application Builder, click **Export/Import**.
4. When prompted to select a task, select **Export** and click **Next**.
5. From File Format, select how rows in the export file will be formatted:
   - Choose **UNIX** to have the resulting file contain rows delimited by line feeds.
   - Choose **DOS** to have the resulting file contain rows delimited by carriage returns and line feeds.
6. Click **Export Workspace Users**.

Exporting a Page in an Application

You can export a specific page within an application by clicking the Export button on the Page Definition. When exporting a page, remember the following:

- Exported pages can only be imported sucessfully if they have the same application ID and workspace ID.
- Some pages may reference shared components. To export all pages within an application as well as application shared components, you need to export the entire application.

See Also: "Exporting an Application" on page 11-5 and "Importing Export Files" on page 11-11

To export a page in an application:

1. Navigate to the appropriate Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.

   d. Select a page.

      The Page Definition appears.

2. On the Page Definition, click **Export** at the top of the page.
3. From Page, select the page to be exported.
4. From File Format, select how rows in the export file will be formatted:
   - Choose **UNIX** to have the resulting file contain rows delimited by line feeds.
   - Choose **DOS** to have the resulting file contain rows delimited by carriage returns and line feeds.
5. Use the **As of** field to export a page as it was previously defined. Specify the number of minutes in the field provided.

   This utility uses the `DBMS_FLASHBACK` package. Because the timestamp to System Change Number (SCN) mapping is refreshed approximately every five minutes, you may have to wait that amount of time to locate the version for which you are looking. The time undo information is retained and influenced by the startup parameter `UNDO_RETENTION` (the default is three hours). However, this only influences the size of the undo tablespace. While two databases may have the same `UNDO_RETENTION` parameter, you will be able to go back further in time on a database with fewer transactions because it is not filling the undo tablespace, forcing older data to be archived.
6. Click **Export Page**.

Exporting Cascading Style Sheets

Use the Export Cascading Style Sheets utility to export uploaded cascading style sheets. Note that you can only use this utility to export uploaded cascading style sheets.

To export related cascading style sheets:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application Builder, click **Export/Import**.
4. When prompted to select a task, select **Export** and click **Next**.
5. Click **CSS** at the top of the page.
6. On the Export Cascading Style Sheets page:
   a. Select the cascading style sheets
   b. From File Format, select how rows in the export file will be formatted:
      – Choose **UNIX** to have the resulting file contain rows delimited by line feeds.
      – Choose **DOS** to have the resulting file contain rows delimited by carriage returns and line feeds.
7. Click **Export Style Sheets**.

Exporting Images

Use the Export Images utility to export uploaded images. When you export images using this utility, the images are converted to a text document. Note that you can only use this utility to export uploaded images.

To export upload images:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application Builder, click **Export/Import**.
4. When prompted to select a task, select **Export** and click **Next**.
5. Click **Images** at the top of the page.
6. On the Export Images page:
   a. Select an application from which to export images.

      Be aware that selecting **Workspace Images** only exports those images in your repository that are not associated with a specific application. If all of your images are associated with specific applications, then the workspace image export file will be empty.

   b. From File Format, select how rows in the export file will be formatted:
      – Choose **UNIX** to have the resulting file contain rows delimited by line feeds.
      – Choose **DOS** to have the resulting file contain rows delimited by carriage returns and line feeds.
7. Click **Export Images**.

Exporting Static Files

Use the Export Static Files utility to export static files you have imported. Note that you can only use this utility to export uploaded static files.

To export related static files:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application Builder, click **Export/Import**.
4. When prompted to select a task, select **Export** and click **Next**.
5. Click **Files** at the top of the page.
6. On Export Static Files:
   a. Select the files to be exported.

   b. From File Format, select how rows in the export file will be formatted:
      – Choose **UNIX** to have the resulting file contain rows delimited by line feeds.
      – Choose **DOS** to have the resulting file contain rows delimited by carriage returns and line feeds.
7. Click **Export File(s)**.

About Importing into Another Oracle Application Express Instance

Note that you cannot use the Web interface described in this section to import exported static files into another Oracle Application Express instance. To import exported static files into another Oracle Application Express instance, use SQL Command Line (SQL*Plus) while connected as the Oracle Application Express database user. Note that you must export from and to a workspace having the same name and workspace ID.

> **See Also:** "Using SQL Command Line" in *Oracle Database Express Edition 2 Day Developer Guide*

Exporting Script Files

You can transfer selected scripts from your current Script Repository to a Script Repository in a different Workspace by using the Export and Import tasks.

> **See Also:** "Using SQL Scripts" on page 18-1

To export script files:

1. On the Database Home Page, click the **SQL** icon.
2. Click **SQL Scripts**.
3. On the Tasks list, click **Export**.
4. Select the appropriate script files and click **Add to Export**.
5. Review the file name and click **Export All**.

   Select the Remove check box to remove the script.

Exporting Themes

Use the Export Theme utility to export themes from one Oracle Application Express development instance to a file.

To export an application theme from the Export page:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application Builder home page, click **Export/Import**.
4. When prompted to select a task, select **Export** and click **Next**.
5. Click **Themes** at the top of the page.
6. On the Export Application Theme page:
   a. Select the theme to be exported.
   b. From File Format, select how rows in the export file will be formatted:
      – Choose **UNIX** to have the resulting file contain rows delimited by line feeds.
      – Choose **DOS** to have the resulting file contain rows delimited by carriage returns and line feeds.
7. Click **Export Theme**.

To export an application theme from the Themes page:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application Builder home page, click **Shared Components**.
4. Under User Interface, select **Themes**.

   The Themes page appears.
5. On the Tasks list, click **Export Theme**.

   The Export page appears.
6. Click **Themes** at the top of the page.
7. On the Export Application Theme page:
   a. Select the theme to be exported.
   b. From File Format, select how rows in the export file will be formatted:
      – Choose **UNIX** to have the resulting file contain rows delimited by line feeds.
      – Choose **DOS** to have the resulting file contain rows delimited by carriage returns and line feeds.
8. Click **Export Theme**.

See Also: "Managing Themes" on page 7-8

Exporting User Interface Defaults

Exporting User Interface Defaults is useful when you plan to develop on the target machine.

When you export User Interface Defaults, all User Interface Defaults for the selected schema are exported to a single SQL Command script. When prompted, save this file to your hard drive. The file contains an API call to create table hints by making calls to the application PL/SQL API. You can use this file to import User Interface Defaults to another database and Oracle Application Express instance.

See Also: "How to Deploy an Application to Another Oracle Application Express Instance" on page 11-4 and "Managing User Interface Defaults" on page 9-1

To export User Interface Defaults from the Export page:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application Builder home page, click **Export/Import**.
4. When prompted to select a task, select **Export** and click **Next**.
5. Click **User Interface Defaults** at the top of the page.
6. On the Export User Interface Defaults page:
   a. From Schema, select the schema that owns the table associated with the User Interface Defaults.
   b. From File Format, select how rows in the export file will be formatted:

  - Choose **UNIX** to have the resulting file contain rows delimited by line feeds.
  - Choose **DOS** to have the resulting file contain rows delimited by carriage returns and line feeds.

7. Click **Export**.

To export User Interface Defaults from the User Interface Defaults page:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application Builder home page, click **Shared Components**.
4. Under User Interface, select **User Interface Defaults**.

   The User Interface Defaults page appears.
5. Click **Export**.

   The Export page appears.
6. Click **Themes** at the top of the page.
7. On the Export User Interface Defaults page:

   a. From Schema, select the schema that owns the table associated with the User Interface Defaults.

   b. From File Format, select how rows in the export file will be formatted:

      - Choose **UNIX** to have the resulting file contain rows delimited by line feeds.
      - Choose **DOS** to have the resulting file contain rows delimited by carriage returns and line feeds.
8. Click **Export**.

See Also: "Managing User Interface Defaults" on page 9-1

Importing Export Files

Once you export an application and any related files, you need to import them into the target Oracle Application Express instance before you can install them. As a general rule, always import the application first and then the related files.

To import an application and related files:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application Builder home page, click **Export/Import**.
4. When prompted to select a task, select **Import** and click **Next**.
5. On Import File:

   a. In Import file, specify the file you are importing.

   b. From File Type, select the type of file you are importing.

   Once you have imported a file, you have the option to install it. You can also install it later from the Export Repository.

   c. Verify that File Character Set is correct.

   d. Click **Next**.

6. To install an imported file, click **Install**. On the Application Install page:
   a. From Parse As Schema, select a schema.

      This is the schema against which all of the application's SQL and PL/SQL will be parsed.
   b. From Build Status Override, select one of the following:
      – **Run Application Only** - Users can only run an application.
      – **Run and Build Application** - Users can run an application and developers can both run and edit an application

      Selecting **Run Application Only** is an effective way to protect an application from modifications from other developers.
   c. From Install As Application, select one of the following:
      – **Auto Assign New Application ID**
      – **Reuse Application ID From Export File**
      – **Change Application ID**

      Use these options to avoid application ID conflicts. These options come in handy when you need to have two versions of the same application in the same instance. For example, you might be migrating an application to a production instance and still need to maintain development version.
   d. Click **Install Application**.

      See Also: "Installing Export Files" on page 11-12 and "Enabling SQL Tracing and Using TKPROF" on page 10-2

Installing Export Files

There are two ways to install export files:

- Import the export files into Application Builder and then install the files from the Export Repository
- Install the export files from SQL Command Line (SQL*Plus)

Topics in this section include:

- Installing Files from the Export Repository
- Installing Export Files from SQL Command Line

Installing Files from the Export Repository

After you import files into the target Oracle Application Express instance, you must install them before they can become active or available in Application Builder.

To install files stored in the Export Repository:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application Builder home page, click **Export/Import**.
4. Select **View Repository** on the bottom of the page.

5. Select the file to be installed and click **Install** in the Action column adjacent to the appropriate file.

   a. From Parse As Schema, select a schema.

      This is the schema against which all of the application's SQL and PL/SQL will be parsed.

   b. From Build Status, select one of the following:

      – **Run Application Only**

      – **Run and Build Application**

      Select **Run Application Only** to run the application in the target instance and make it inaccessible to developers.

   c. From Install As Application, select one of the following:

      – **Reuse Application ID from Export File**

      – **Auto Assign New Application ID**

      – **Change Application ID**

      Use these options to avoid application ID conflicts. Use these options when you need to have two versions of the same application in the same instance. For example, you might be migrating an application to a production instance and still need to maintain development version.

   d. Click **Install Application**.

In addition to installing files, you can also use this page to delete a file from the Export Repository.

To delete a file from the Export Repository:

1. Navigate to the Export Repository.
2. Select the file to be deleted and click **Delete Checked**.

Installing Export Files from SQL Command Line

You can also install export files from SQL Command Line (SQL*Plus). Note there are two restrictions:

- The export file must originate from the same user database account as the one into which you are installing.
- If the export file is an application, the application ID will be overwritten. Therefore, the target workspace must own the ID of the application being installed.

Topics in this section include:

- Verifying If Source and Target Workspace IDs Are Identical
- Using SQL Command Line to Install Export Files

See Also: "Using SQL Command Line" in *Oracle Database Express Edition 2 Day Developer Guide*

Verifying If Source and Target Workspace IDs Are Identical

You can verify that the source and target database user accounts are identical by running query in SQL Command Processor.

To verify that the source and target database user accounts are identical:

1. Log in to the source workspace.
2. Click the **SQL** icon on the Database Home Page.
3. Click **SQL Commands**.
4. Enter the following in the SQL editor pane and click **Run**.

   ```
   SELECT &WORKSPACE_ID. FROM DUAL
   ```

5. Make note of workspace ID.
6. Log in to the target workspace.
7. Repeat steps 2 through 5 to verify the workspace IDs match.

Using SQL Command Line to Install Export Files

To install Oracle Application Express export files from SQL Command Line (SQL*Plus):

1. Log in to SQL Command Line using your Oracle Application Express database user account.
2. Run the export file.

   For example, if your export file is names f144.sql by default, you would type `@f144` at the command prompt.

About Publishing the Application URL

Once you have deployed your application, loaded data, and created users, you can publish your production URL.

You can determine the URL to your application by positioning the mouse over the **Run** icon on the Application home page. The URL displays in the status bar at the bottom of the page.

The Run icon gets its value from the Home link attribute on the Edit Security Attributes page. This link is only referenced by this icon and by applications that do not use the Oracle Application Express Login API. Consider the following example:

```
http://apex.oracle.com/pls/apex/f?p=11563:1:3397731373043366363
```

Where:

- `apex.oracle.com` is the URL of the server.
- `pls` is the indicator to use the `mod_plsql` cartridge.
- `apex` is the database access descriptor (DAD) name. The DAD describes how Oracle HTTP Server connects to the database server so that it can fulfill an HTTP request. The default value is `apex`.
- `f?p=` is a prefix used by Oracle Application Express.
- `11563` is the application being called.
- `1` is the page within the application to be displayed.
- `3397731373043366363` is the session number.

To run this example application, you would use the URL:

```
http://apex.oracle.com/pls/apex/f?p=11563:1
```

When each user logs in, he or she will receive an unique session number.

> **See Also:** "Accessing the Edit Security Attributes Page" on page 4-12 and "Managing HTTP Access to the User Interface" on page 21-19

Using Build Options to Control Configuration

Build options enable you to conditionally display specific functionality within an application.

Build options have two possible values: INCLUDE and EXCLUDE. If you specify an attribute as being included, then the Application Express engine considers it part of the application definition at run time. Conversely, if you specify an attribute as being excluded, then the Application Express engine treats it as if it does not exist.

Topics in this section include:

- Creating Build Options
- About the Build Options Page
- Viewing the Build Option Utilization Report

Creating Build Options

To create a build option:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application Builder home page, click **Shared Components**.
4. Under Logic, select **Build Options**.
5. To create a new build option, click Create.
6. Follow the on-screen instructions.

You can choose to enable or disable a build option on the appropriate attributes page. Most attributes pages contain a Configuration section where you can select defined build options.

> **See Also:** "Configuring Standard Application Attributes" on page 4-6 and "Editing Page Attributes" on page 4-20 for information about specifying build options

About the Build Options Page

Once you create a build option, it appears on the Build Options page. You control how the Build Options page displays by making a selection from the View list. Available options include:

- **Icons** (the default) displays each build option as a large icon. To edit a build option, click the appropriate icon.
- **Details** displays each build option as a line in a report. Each line includes the application ID, build option name, status, and a link to the Build Option Utilization report. To edit a build option, click the **appropriate** name.

Viewing the Build Option Utilization Report

Once you create a build option, a Utilization button appears on the Build Options page. This report details build option utilization in the current application.

Note: The Utilization button only appears on the Build Options page after you create a build option.

To view the Build Option Utilization report:

1. Navigate to the Build Options page:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. On the Application Builder home page, click **Shared Components**.
   e. Under Logic, select **Build Options**.

      Build Options page appears.
2. On the Build Options page, click **Utilization**.

   The Build Option Utilization report appears.

12 Managing Application Security

This section describes how to provide security for Oracle Application Express applications. You can provide security for an Oracle Application Express application by utilizing cross-site scripting protection, session state protection, authentication, and authorization.

This section contains the following topics:

See Also: "Application Builder Concepts" on page 3-1 and "Using Application Builder" on page 4-1

About Cross-Site Scripting Protection

Cross site-scripting (also referred to as XSS) is a security breach that takes advantage of dynamically generated Web pages. In a XSS attack, a Web application is sent a script that activates when it is read by a user's browser. Once activated, these scripts can steal data, or even session credentials and return the information to the attacker. If malicious code were introduced into an Oracle Application Express application, it could be rendered into HTML regions and other places within the application during normal page rendering. To prevent the introduction of malicious code into session state, the Application Express engine escapes characters in certain cases.

Topics in this section include:

Protecting HTML Regions and Other Static Areas

In HTML regions and other static display areas, you can reference session state using the `&ITEM.` notation. Examples of static displays areas include HTML regions, page headers and footers, region headers and footers, region titles, button labels, help text, form item labels and post-element text, templates, radiogroup (before and after field text), event success messages, event error messages, navigation bar attributes, application static substitution string values, chart labels and legends, breadcrumbs and list framing text, and calendar text, labels, or legends.

When session state is referenced in this way, the value emitted to the page will have special characters (<, >, &, ") escaped if the referenced item has one of the following safe item display types:

- Display as Text (does not save state)
- Display as Text (escape special characters, does not save state)
- Display as Text (based on LOV, does not save state)
- Display as Text (based on PL/SQL, does not save state)
- Text Field (Disabled, does not save state)
- Stop and Start HTML Table (Displays label only)

If the referenced item has a display type other than one of the above types, the value emitted to the page will not have special characters escaped. Although application-level items are also considered to have a safe display type, they do not actually have display properties like form items do.

See Also: "Creating Items" on page 5-51

About the Rules Used to Determine Whether to Escape Values

Application Express engine uses predefined smart escaping rules to determine when and whether to escape values fetched from session state.

The reason for these rules is that items that use the display types listed previously are often for text containing HTML which is intended to be emitted to the browser without being filtered (escaped). The only way this can be made safe is by the enforcement of the rule that these types of items are always escaped on input to the application. For example, if a user passes some text into a safe item using an Oracle Application Express f?p URL syntax, the Application Express engine escapes special characters when saving the value into session state. This has two intended results:

1. If the value contained no special characters, the value passed in is saved into session state exactly as it was provided.
2. If the value contained special characters, those characters are escaped when the value is saved into session state.

In either situation, the item can now safely be referenced using `&ITEM.` notation in any HTML region or other static area mentioned previously.

Using Safe Item Types to Hold and Emit HTML Markup

You can use safe item types listed previously to hold and emit HTML markup to the browser. For example, suppose you have a requirement to render some text in bold face by referencing a safe page item named `P1_XXX` (using `&P1_XXX.`) The item `P1_XXX` is presumed to contain the following HTML:

```
<b>ABABABAB</b>
```

You can achieve this by using application controls (computations, processes, item source expressions, item default values, and so on) to store values into these safe items. When values are introduced in this way, you ensure that the safety of the content. When you use these methods, the Application Express engine does not escape any special characters when saving the values into session state.

Finally, the safety of safe items is ensured by a rule that prevents those items from being posted on a page and submitted to the Application Express engine as part of a page submission.

Protecting Dynamic Output

Items fetched from session state and rendered using `htp.p` or other methods should be explicitly escaped by the code where it is appropriate to do so. For example, suppose PL/SQL dynamic content region on a page uses the following:

```
 htp.p(v('SOME_ITEM'));
```

If the value of the item fetched from session state could contain unintended tags or script, you might want to use the following in the region:

```
htp.p(htf.escape_sc(v('SOME_ITEM')));
```

However, if you are confident that the fetched value is safe for rendering, you do not need to escape the value. As a developer, you need to determine when it is appropriate to not escape output.

As a best practice, follow this rule:

- Never emit an item fetched from session state without escaping it unless the item is one of the safe types.

The reason for this is that as a developer, there is no way you can prevent a hacker from posting a malicious value into a non-safe item. Even if your application does not present these items visibly to ordinary users, be aware that a hacker can mount a XSS attack using your application if you do not follow this rule.

Protecting Report Regions

The Application Express engine escapes data rendered in the body of a report. References to session state in report headings and messages, are fetched from session state using the smart escaping rules so that the values of safe item types are not escaped and the values of other item types are escaped.

Protecting Form Items

When form items, including hidden items, obtain their values during the generation of the form page to be sent to the browser, the resulting text is escaped before rendering. Some of the safe item types are exceptions to this rule in order to support the intended behavior of each display type.

Understanding Session State Protection

Session State Protection is a built-in functionality that prevents hackers from tampering with the URLs within your application. URL tampering can adversely affect program logic, session state contents, and information privacy.

Enabling Session State Protection is a two step process. First, you enable the feature. Second, you set page and item security attributes.

Topics in this section include:

- How Session State Protection Works
- Enabling Session State Protection

- Configuring Session State Protection

How Session State Protection Works

When enabled, Session State Protection uses the Page Access Protection attributes and the Session State Protection item attributes in conjunction with checksums positioned in `f?p=` URLs to prevent URL tampering and unauthorized access to and alteration of session state. When Session State Protection is disabled, the page and item attributes related to session state protection are ignored and checksums are not included checksums in generated `f?p=` URLs.

Enabling Session State Protection

You can enable session state protection from either the Edit Security Attributes page or the Session State Protection page.

Enabling Session State Protection is a two step process. First, you enable the feature. Second, you set page and item security attributes. You can perform these steps using a wizard, or you can set security attributes for pages and items manually on the Session State Protection page.

Topics in this section include:

- Enabling Session State Protection from Edit Security Attributes
- Enabling Session State Protection from Session State Protection

Enabling Session State Protection from Edit Security Attributes

To enable Session State Protection from the Edit Security Attributes page:

1. On the Database Home Page, click the **Application Builder** icon
2. Select an application.
3. Click Edit Attributes.
4. Click **Edit Security Attributes**.
5. Scroll down to Session State Protection and select **Enabled** from the Session State Protection list.
6. To configure session Session State Protection, click **Manage Session State Protection**.

   The Session State Projection page appears.
7. Navigate to the Edit Security Attributes page to set page and item security attributes.

> **Tip:** To disable Session State Protection, perform the previous steps again, but select **Disabled** instead of **Enabled**. Disabling Session State Protection will not change existing security attribute settings, but those attributes will be ignored at runtime.

About the Expire Bookmarks Button Enabling Session State Protection affects whether or not bookmarked links to the current application will work. Consider the following rules:

1. Bookmarked links created after Session State Protection is enabled will work if the bookmarked link contains a checksum.

2. Bookmarked links created before Session State Protection is enabled will not work if the bookmarked link contains a checksum.
3. Bookmarks that do not contain checksums or contain unnecessary checksums will not be affected by Session State Protection.

During page rendering, the Application Express engine uses a hidden application attribute (a checksum salt) during computation and to verify checksums included in `f?p` URLs. When you enable Session State Protection, the Application Express engine includes checksums. You can reset the checksum salt attribute by clicking **Expire Bookmarks** on the Edit Security Attributes page. Note that if you click **Expire Bookmarks,** bookmarked URLs used to access your application that contain previously generated checksums will fail.

Enabling Session State Protection from Session State Protection

To enable Session State Protection:

1. Navigate to the Shared Components page:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Under Security, select **Session State**.

      The Session State Protection page appears. Note the current Session State Protection status (Enabled or Disabled) displays at the top of the page.
2. Click the **Set Protection** button.

   The Session State Protection wizard appears.
3. Under Select Action, select **Enable** and click **Next**.

   Next, determine whether to set security attributes for pages and items.
4. Select **Enable** and click **Next**.
5. Click **Enable Session State Protection**.

Tip: To disable Session State Protection, perform the same steps, but select **Disable** instead of **Enable**. Disabling Session State Protection will not change existing security attribute settings, but will be ignore them at run time.

Configuring Session State Protection

Once you have enabled Session State Protection, the next step is to configure security attributes. You can configure security attributes in two ways:

- Use a wizard and select a value for specific attribute categories. Those selections will then be applied to all pages and items within the application.
- Configure values for individual pages, items, or application items.

Topics in this section include:

- Reviewing Existing Session State Protection Settings
- Configuring Session State Protection Using a Wizard
- Configuring Session State Protection for Pages
- Configuring Session State Protection for Items
- Configuring Session State Protection for Application Items

Tip: Before you can configure security attributes, you must first enable Session State Protection. See "Enabling Session State Protection" on page 12-4.

Reviewing Existing Session State Protection Settings

You can review a summary of Session State Protection settings for pages, items, and application items on the first page of the Session State Protection wizard.

To view summaries of existing Session State Protection settings:

1. Navigate to the Session State Protection page:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under Security, select **Session State Protection**.

      The Session State Protection page appears.
2. Click **Set Protection**.
3. Expand the following reports at the bottom of the page:
   - Page Level Session State Protection Summary
   - Page Item Session State Protection Summary
   - Application Item Session State Protection

Configuring Session State Protection Using a Wizard

When you configure Session State Protection using a wizard, you set a value for specific attribute categories. Those selections are then applied to all pages and items within the application.

To configure Session State Protection using a wizard:

1. Navigate to the Session State Protection page:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under Security, select **Session State Protection**.

      The Session State Protection page appears.
2. Click **Set Protection**.

   The Session State Protection wizard appears.
3. Under Select Action, select **Configure** and click **Next**.
4. For Page Access Protection, select one of the following:
   - **Unrestricted** - The page may be requested using an URL with or without session state arguments (Request, Clear Cache, Name/Values).
   - **Arguments Must Have Checksum** - If Request, Clear Cache, or Name/Value arguments appear in the URL, a checksum must also be provided. The checksum type must be compatible with the most stringent Session State Protection attribute of all the items passed as arguments.

- **No Arguments Allowed** - A URL may be used to request the page but no Request, Clear Cache, or Name/Value arguments are allowed.
- **No URL Access** - The page may not be accessed using a URL, however the page may be the target of a Branch to Page branch type, which does not do a URL redirect.

5. For Application Item Protection, select one of the following:

   - **Unrestricted** - The item's session state may be set by passing the item name/value in a URL or in a form. No checksum is required in the URL.
   - **Checksum Required: Application Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the schema. A user-level checksum or a session-level checksum will also suffice (see next bullets). Use this option when you want to allow the item to be set only by URLs having checksums that were generated by any user running the same application in the current workspace but in a different session.
   - **Checksum Required: User Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the workspace, application, and user. A session-level checksum will also suffice (see next bullet). Use this option when you want to allow the item to be set only by URLs having checksums that were generated by the same named user, running the same application in the current workspace but in a different session.
   - **Checksum Required: Session Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the current session. Use this option when you want to allow this item to be set only by URLs having checksums that were generated in the current session.
   - **Restricted - May not be set from browser** - The item may not be altered using the URL or POSTDATA. Use this option when you want to restrict the way that the item value can be set to internal processes, computations, and so on. This attribute is applicable only to items that cannot be used as data entry items and is always observed even if Session State Protection is disabled.

     Use this attribute for application items or for page items with any of these Display As types:

     – Display as Text (escape special characters, does not save state)

     – Display as Text (does not save state)

     – Display as Text (based on LOV, does not save state)

     – Display as Text (based on PLSQL, does not save state)

     – Text Field (Disabled, does not save state)

     – Stop and Start HTML Table (Displays label only)

6. For Page Data Entry Item Protection, select one of the following:

   - **Unrestricted** - The item's session state may be set by passing the item name/value in a URL or in a form. No checksum is required in the URL.
   - **Checksum Required: Application Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the schema. A user-level checksum or a session-level checksum will also suffice (see next bullets). Use this option when you want to allow the

item to be set only by URLs having checksums that were generated by any user running the same application in the current workspace but in a different session.

- **Checksum Required: User Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the workspace, application, and user. A session-level checksum will also suffice (see next bullet). Use this option when you want to allow the item to be set only by URLs having checksums that were generated by the same named user, running the same application in the current workspace but in a different session.
- **Checksum Required: Session Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the current session. Use this option when you want to allow this item to be set only by URLs having checksums that were generated in the current session.

7. For Page Display-Only Item Protection, select one of the following:
   - **Unrestricted** - The item may be set by passing the item name/value in a URL or in a form. No checksum is required in the URL.
   - **Checksum Required: Application Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the schema. A user-level checksum or a session-level checksum will also suffice (see next bullets). Use this option when you want to allow the item to be set only by URLs having checksums that were generated by any user running the same application in the current workspace but in a different session.
   - **Checksum Required: Session Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the current session. Use this option when you want to allow this item to be set only by URLs having checksums that were generated in the current session.
   - **Checksum Required: User Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the workspace, application, and user. A session-level checksum will also suffice (see next bullet). Use this option when you want to allow the item to be set only by URLs having checksums that were generated by the same named user, running the same application in the current workspace but in a different session.
   - **Restricted: May not be set from browser** - The item may not be altered using the URL or POSTDATA. Use this when you want to restrict the way that the item value can be set to internal processes, computations, and so on. This attribute is always observed, even if Session State Protection is disabled.

     This attribute may be used with any of these Display As types:

     – Display as Text (escape special characters, does not save state)

     – Display as Text (does not save state)

     – Display as Text (based on LOV, does not save state)

     – Display as Text (based on PLSQL, does not save state)

     – Text Field (Disabled, does not save state)

     – Stop and Start HTML Table (Displays label only)

8. Click **Next**.
9. Click **Finish**.

Configuring Session State Protection for Pages

To configure Session State Protection for Pages:

1. Navigate to the Session State Protection page:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under Security, select **Session State Protection**.

      The Session State Protection page appears.
2. Click the **Page** icon.
3. To filter the view, use the Page, Display, and Page Access Protection lists at the top of the page.
4. Select a page ID.

   The Set Page and Item Protection page appears. The following information displays at the top of the page:
   - Application ID and name
   - Session State Protection status (Enabled or Disabled)
   - Page ID
   - Page name
5. For Page Access Protection, select one of the following:
   - **Unrestricted** - The page may be requested using an URL with or without session state arguments (Request, Clear Cache, Name/Values).
   - **Arguments Must Have Checksum** - If Request, Clear Cache, or Name/Value arguments appear in the URL, a checksum must also be provided. The checksum type must be compatible with the most stringent Session State Protection attribute of all the items passed as arguments.
   - **No Arguments Allowed** - A URL may be used to request the page but no Request, Clear Cache, or Name/Value arguments are allowed.
   - **No URL Access** - The page may not be accessed using a URL, however the page may be the target of a Branch to Page branch type, which does not do a URL redirect.
6. For Item Types, select **Data Entry Items** or **Display-only Items**.

   Data Entry items are items that can be altered using forms and include hidden items. Display-Only items are rendered only and are not submitted with the form.
7. If you select **Data Entry Items**, select a session state protection level for each item:
   - **Unrestricted** - The item's session state may be set by passing the item name/value in a URL or in a form. No checksum is required in the URL.
   - **Checksum Required: Application Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the schema. A user-level checksum or a session-level checksum

will also suffice (see next bullets). Use this option when you want to allow the item to be set only by URLs having checksums that were generated by any user running the same application in the current workspace but in a different session.

- **Checksum Required: User Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the workspace, application, and user. A session-level checksum will also suffice (see next bullet). Use this option when you want to allow the item to be set only by URLs having checksums that were generated by the same named user, running the same application in the current workspace but in a different session.
- **Checksum Required: Session Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the current session. Use this option when you want to allow this item to be set only by URLs having checksums that were generated in the current session.

8. If you select **Display-only Item**, select a session state protection level for each item:
   - **Unrestricted** - The item may be set by passing the item name/value in a URL or in a form. No checksum is required in the URL.
   - **Restricted: May not be set from browser** - The item may not be altered using the URL or POSTDATA. Use this when you want to restrict the way that the item value can be set to internal processes, computations, and so on. This attribute is always observed, even if Session State Protection is disabled. This attribute may be used with any of these Display As types:
     - Display as Text (escape special characters, does not save state)
     - Display as Text (does not save state)
     - Display as Text (based on LOV, does not save state)
     - Display as Text (based on PLSQL, does not save state)
     - Text Field (Disabled, does not save state)
     - Stop and Start HTML Table (Displays label only)
   - **Checksum Required: Application Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the schema. A user-level checksum or a session-level checksum will also suffice (see next bullets). Use this option when you want to allow the item to be set only by URLs having checksums that were generated by any user running the same application in the current workspace but in a different session.
   - **Checksum Required: User Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the workspace, application, and user. A session-level checksum will also suffice (see next bullet). Use this option when you want to allow the item to be set only by URLs having checksums that were generated by the same named user, running the same application in the current workspace but in a different session.
   - **Checksum Required: Session Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the current session. Use this option when you want to allow this

item to be set only by URLs having checksums that were generated in the current session.

9. Click **Apply Changes**.

Configuring Session State Protection for Items

To configure Session State Protection for items:

1. Navigate to the Session State Protection page:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under Security, select **Session State Protection**.

      The Session State Protection page appears.
2. Click the **Item** icon.
3. To filter the view, select from the Page, Display, and Item Session State Protection lists at the top of the page and click **Go**.
4. Select a page ID.

   The Edit Session State Protection for Page and Items page appears. The following information displays at the top of the page:
   - Application ID and name
   - Session State Protection status (Enabled or Disabled)
   - Page ID
   - Page name
5. For Page Access Protection, select a session state protection level for each item:
   - **Unrestricted** - The page may be requested using an URL with or without session state arguments (Request, Clear Cache, Name/Values).
   - **Arguments Must Have Checksum** - If Request, Clear Cache, or Name/Value arguments appear in the URL, a checksum must also be provided. The checksum type must be compatible with the most stringent Session State Protection attribute of all the items passed as arguments.
   - **No Arguments Allowed** - A URL may be used to request the page but no Request, Clear Cache, or Name/Value arguments are allowed.
   - **No URL Access** - The page may not be accessed using a URL, however the page may be the target of a Branch to Page branch type, which does not do a URL redirect.
6. For Item Types, select **Data Entry Items** or **Display-only Items**.

   Data Entry items are items that can be altered using forms and include hidden items. Display-Only items are rendered only and are not submitted with the form.
7. If you select **Data Entry Items**, select a session state protection level for each item:
   - **Unrestricted** - The item's session state may be set by passing the item name/value in a URL or in a form. No checksum is required in the URL.
   - **Checksum Required: Application Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that

is specific to the schema. A user-level checksum or a session-level checksum will also suffice (see next bullets). Use this option when you want to allow the item to be set only by URLs having checksums that were generated by any user running the same application in the current workspace but in a different session.

- **Checksum Required: User Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the workspace, application, and user. A session-level checksum will also suffice (see next bullet). Use this option when you want to allow the item to be set only by URLs having checksums that were generated by the same named user, running the same application in the current workspace but in a different session.
- **Checksum Required: Session Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the current session. Use this option when you want to allow this item to be set only by URLs having checksums that were generated in the current session.

8. If you select **Display-only Item**, select a session state protection level for each item:
   - **Unrestricted** - The item may be set by passing the item name/value in a URL or in a form. No checksum is required in the URL.
   - **Restricted: May not be set from browser** - The item may not be altered using the URL or POSTDATA. Use this when you want to restrict the way that the item value can be set to internal processes, computations, and so on. This attribute is always observed, even if Session State Protection is disabled. This attribute may be used with any of these Display As types:
     - Display as Text (escape special characters, does not save state)
     - Display as Text (does not save state)
     - Display as Text (based on LOV, does not save state)
     - Display as Text (based on PLSQL, does not save state)
     - Text Field (Disabled, does not save state)
     - Stop and Start HTML Table (Displays label only)
   - **Checksum Required: Application Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the schema. A user-level checksum or a session-level checksum will also suffice (see next bullets). Use this option when you want to allow the item to be set only by URLs having checksums that were generated by any user running the same application in the current workspace but in a different session.
   - **Checksum Required: User Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the workspace, application, and user. A session-level checksum will also suffice (see next bullet). Use this option when you want to allow the item to be set only by URLs having checksums that were generated by the same named user, running the same application in the current workspace but in a different session.
   - **Checksum Required: Session Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is

specific to the current session. Use this option when you want to allow this item to be set only by URLs having checksums that were generated in the current session.

9. Click **Apply Changes**.

Configuring Session State Protection for Application Items

To configure Session State Protection for application items:

1. Navigate to the Session State Protection page:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under Security, select **Session State Protection**.

      The Session State Protection page appears.
2. Click the **Application Item** icon.
3. Select an application item.
4. Under Security, select one of the following from the Session State Protection list:
   - **Unrestricted** - The item's session state may be set by passing the item name/value in a URL or in a form. No checksum is required in the URL.
   - **Restricted - May not be set from browser** - The item may not be altered using the URL or POSTDATA. Use this option when you want to restrict the way that the item value can be set to internal processes, computations, and so on. This attribute is only applicable only to items that cannot be used as data entry items and is always observed even if Session State Protection is disabled. This attribute may be used for application items or for page items with any of these Display As types:
     – Display as Text (escape special characters, does not save state)
     – Display as Text (does not save state)
     – Display as Text (based on LOV, does not save state)
     – Display as Text (based on PLSQL, does not save state)
     – Text Field (Disabled, does not save state)
     – Stop and Start HTML Table (Displays label only)
   - **Checksum Required: Application Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the schema. A user-level checksum or a session-level checksum will also suffice (see next bullets). Use this option when you want to allow the item to be set only by URLs having checksums that were generated by any user running the same application in the current workspace but in a different session.
   - **Checksum Required: User Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the workspace, application, and user. A session-level checksum will also suffice (see next bullet). Use this option when you want to allow the item to be set only by URLs having checksums that were generated by the same named user, running the same application in the current workspace but in a different session.

- **Checksum Required: Session Level** - The item's session state may be set by passing the item name/value in a URL if a checksum is also provided that is specific to the current session. Use this option when you want to allow this item to be set only by URLs having checksums that were generated in the current session.

5. Click **Apply Changes**.

Understanding the Security Risks of File Upload Tables

Oracle Application Express enables you to easily build an application which enables users to upload files and access uploaded files. These files are uploaded into a common file storage table. Although the database view `HTMLDB_APPLICATION_FILES` will only show those files associated with your database account (or workspace), authentication is not required to access any of the files stored in the underlying table, including those outside of your database account (or workspace) and owned by other users. Using the various APIs in Oracle Application Express, a user can specify the numeric ID associated with a file in this common file storage table and access it without requiring authentication. Files stored in this table are accessible by anyone.

To implement an Oracle Application Express application which supports file upload, but does not expose this security vulnerability, please refer to the Oracle Application Express How To Documents for file upload on OTN at:

```
http://www.oracle.com/technology/products/database/htmldb/howtos/index.html
```

See Also: "Creating a Page-Level Item" on page 5-51 and "About Item Types" on page 5-52 to learn more about creating a File Browse page-level item

Establishing User Identity Through Authentication

Authentication is the process of establishing each user's identify before they can access your application. Authentication may require a user identify a username and password or could involve the use of digital certificates or a secure key.

When you create an authentication scheme, you have the option of choosing from a number of preconfigured authentication schemes, copying an authentication scheme from an existing application, or creating your own custom authentication scheme.

Topics in this section include:

- Understanding How Authentication Works
- Determining Whether to Include Authentication
- Creating an Authentication Scheme
- Using the Authentication Scheme Repository
- Viewing the Current Authentication Scheme for an Application
- Changing the Current Authentication Scheme for an Application
- Viewing Authentication Scheme Utilization
- About Preconfigured Authentication Schemes
- About Creating an Authentication Scheme from Scratch

Understanding How Authentication Works

You determine how your application interacts with users. If all users have the same rights and privileges they are referred to as public users. However, if your application needs to track each user individually, you need to specify an authentication method.

Authentication establishes the identity of each user who accesses your application. Many authentication processes require a user provide some type of credentials such as a username and password. These credentials are then evaluated and they either pass or fail. If the credentials pass, the user has access to the application. Otherwise, access is denied.

Once a user has been identified, the Application Express engine keeps track of each user by setting the value of the built-in substitution string `APP_USER`. As a user navigates from page to page, the Application Express engine sets the value of `APP_USER` to identify the user. The Application Express engine uses `APP_USER` as one component of a key for tracking each user's session state.

From a programming perspective, you can access `APP_USER` using the following syntax:

- From PL/SQL:

  ```
  V('APP_USER')
  ```

- As a bind variable from either PL/SQL or SQL:

  ```
  :APP_USER
  ```

You can use `APP_USER` to perform your own security checks and conditional processing. For example, suppose you created the following table:

```
CREATE TABLE my_security_table (
  user_id   VARCHAR2(30),
  privilege VARCHAR2(30));
```

Once created, you could populate this table with user privilege information and then use it to control the display of pages, tabs, navigation bars, buttons, regions, or any other control or component.

See Also: "APP_USER" on page 3-17 and "Configuring Security Attributes" on page 4-12

Determining Whether to Include Authentication

As you create your application, you need to determine whether to include authentication. You can:

- **Choose to not require authentication**. Oracle Application Express does not check any user credentials. All pages of your application are accessible to all users.
- **Select a built-in authentication scheme**. Create an authentication method based on available preconfigured authentication schemes. Depending on which scheme you choose, you may also have to configure the corresponding components of Oracle 10*g*iAS, Oracle Internet Directory, or other external services.
- **Create custom authentication scheme**. Create a custom authentication method, giving you complete control over the authentication interface. To implement this approach, you must provide a PL/SQL function the Application Express engine executes before processing each page request. This function's Boolean return value determines whether the Application Express engine processes the page normally or displays a failure page.

Creating an Authentication Scheme

To create an authentication scheme:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application home page, click **Shared Components**.

   The Shared Components page appears.
4. Under Security, select **Authentication Schemes**.

   The Authentication Schemes page appears.
5. To create a new authentication scheme, click **Create**.
6. Specify how the scheme should be created by selecting one of the following:
   - **Based on preconfigured scheme**
   - **As a copy of an existing scheme**
   - **From scratch**
7. Follow the on-screen instructions

   See Also: "About Preconfigured Authentication Schemes" on page 12-17 and "About Creating an Authentication Scheme from Scratch" on page 12-19

Using the Authentication Scheme Repository

Once created, available authentication schemes display in the Authentication Schemes Repository.

To navigate to the Authentication Schemes Repository:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application home page, click **Shared Components**.

   The Shared Components page appears.
4. Under Security, select **Authentication Schemes**.

   The Authentication Schemes page appears. You can change the appearance of the page by making a selection from the View list. Available options include:
   - **Icons** (the default) displays each authentication scheme as a large icon. To edit an authentication scheme, click the appropriate icon.
   - **Details** displays each application item as a line in a report.

     In Details view you can:
     - Edit an authentication scheme by selecting the scheme name
     - View a list of the steps performed on each page by clicking the **Show** icon
     - Apply an authentication scheme to an application by clicking the **make current** link

Viewing the Current Authentication Scheme for an Application

To view the current authentication scheme for an application:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click Edit Attributes.
4. Click **Edit Security Attributes**.
5. Locate the Authentication section. The current authentication scheme displays next to **Authentication Scheme**.
6. To link to the Authentication Scheme pages and select the scheme name.

Changing the Current Authentication Scheme for an Application

To change the authentication scheme for an application:

1. Navigate to the Authentication Schemes:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. On the Application home page, click **Shared Components**.

      The Shared Components page appears.
   d. Under Security, select **Authentication Schemes**.
2. Click the **Change Current** tab at the top of the page.
3. Select a new authentication scheme and click **Next**.
4. Click **Make Current**.

Viewing Authentication Scheme Utilization

The Authentication Schemes report lists authentication scheme utilization for all applications in the current workspace.

To view the Authentication Schemes report:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.

   The Applications home page appears.
3. On the Tasks list, click **View Application Reports**.
4. Click **Cross Application Reports**.
5. Select **Authentication Schemes**.
6. Click the application ID to link to the appropriate Application home page.

About Preconfigured Authentication Schemes

When you select a preconfigured authentication scheme, Oracle Application Express creates an authentication scheme for your application that follows a standard behavior for authentication and session management. The following list describes available preconfigured authentication schemes:

- **Open Door Credentials** enables anyone to access your application using a built-in login page which captures a username. This can be useful during application development.

- **Application Express Account Credentials** refers to the internal user accounts (also known as "cookie user" accounts) created and managed in the Oracle Application Express user repository. Using method, your application can easily authenticate against these accounts. See "About Application Express Account Credentials" on page 12-18.
- **Database Account Credentials** refers to the use of database schema accounts. When using this method the user name and password of the database account is used to authenticate the user. See "About Database Account Credentials" on page 12-18.
- **LDAP Credentials Verification** requires you specify configuration parameters about the external Lightweight Directory Access Protocol (LDAP) directory you will be using. See "About LDAP Credentials Verification" on page 12-18.
- **No Authentication (using DAD)** gets the username from the database access descriptor (DAD), either as the value stored in the DAD configuration or, if the account information is not stored in the DAD configuration, as the username captured using the basic authentication challenge. See "About DAD Credentials Verification" on page 12-19.

About Application Express Account Credentials

Application Express Account Credentials authentication uses internal user accounts (also known as "cookie user" accounts) created and managed in the Oracle Application Express user repository. You can create and edit database user accounts on the Manage Users page.

> **See Also:** "Managing Database User Accounts" on page 21-10 for information about creating and managing database user accounts

Application Express Account Credentials is a good solution when:

- You want control of the user account repository
- Username and password based approach to security is sufficient
- You do not need to integrate into a single sign-on framework

This is an especially good approach when you need to get a group of users up and running on a new application quickly.

About Database Account Credentials

Database Account Credentials requires that a database user (schema) exist in the local database for the user to be authenticated. You can create and edit database user accounts on the Manage Users page.

> **See Also:** "Managing Database User Accounts" on page 21-10 for information about creating and managing database user accounts

Database Account Credentials is a good choice if having one database account for each named user of your application is feasible and account maintenance using database tools meets your needs.

About LDAP Credentials Verification

Any authentication scheme that uses a login page may be configured to use Lightweight Directory Access Protocol (LDAP) to verify the username and password submitted on the login page. Application Builder includes wizards and edit pages that explain how to configure this option. These wizards assume that an LDAP directory accessible to your application for this purpose already exists and that it can respond to a `SIMPLE_BIND_S` call for credentials verification. When you create a LDAP Credentials authentication scheme, the wizard requests and saves the LDAP host name, LDAP port, and the DN string. An optional pre-processing function can be specified to adjust formatting of the username passed to the API.

About DAD Credentials Verification

DAD database authentication uses the Oracle database native authentication and user mechanisms to authenticate users using a basic authentication scheme. To use DAD credentials verification:

- Each application user must have a user account in the Oracle database.
- You must configure a PL/SQL DAD for basic authentication (without account information).

  This results in one username/password challenge for browser session for your application users. The user identity token is then made available in the `APP_USER` item.

DAD database authentication is useful when you need to implement an authentication method that requires minimal setup for a manageable number of users. Ideally these users would already have self-managed accounts in the database and your use of this authentication method would be short lived (for example, during the demonstration or prototyping stages of development).

The main drawback of this approach is burdensome account maintenance, especially if users do not administer their own passwords, or if their database accounts exist only to facilitate authentication to your application.

About Creating an Authentication Scheme from Scratch

Creating an authentication scheme from scratch gives you complete control over your authentication interface. This is the best approach for applications when any of the following is true:

- Database authentication, or other methods are not adequate.
- You want to develop your own login form and associated methods.
- You want to delegate all aspects of user authentication to external services such as Oracle 10*g*AS Single Sign-On.
- You want to control security aspects of session management.
- You want to record or audit activity at the user or session level.
- You want to enforce session activity or expiry limits.
- You want to program conditional one-way redirection logic before Oracle Application Express page processing.
- You want to integrate your application with non-Oracle Application Express applications using a common session management framework.
- Your application consists of multiple applications that operate seamlessly (for example, more than one application ID).

See Also: "HTMLDB_CUSTOM_AUTH" on page 15-60 for more information

About Session Management Security

When running custom authentication, Oracle Application Express attempts to prevent two improper situations:

- Intentional attempts by a user to access session state belonging to someone else. However, users can still type in an arbitrary application session ID into the URL.
- Inadvertent access to a stale session state (probably belonging to the same user from an earlier time). This would commonly result from using bookmarks to application pages.

Oracle Application Express checks that the user identity token set by the custom authentication function matches the user identity recorded when the application session was first created. If the user has not yet been authenticated and the user identity is not yet known, the session state being accessed does not belong to someone else. These checks determine whether the session ID in the request can be used. If not, the Application Express engine redirects back the same page using an appropriate session ID.

Building a Login Page

When you create a new application in Oracle Application Express, a login page is created. The alias for the page is `'LOGIN'`. You can use this page as the 'invalid session page' in an authentication scheme. The page is constructed with processes that call the Oracle Application Express login API to perform credentials verification and session registration.

You can also build your own login pages using the pre-built pages as models and tailoring all of the user interface and processing logic to your requirements.

To create a login page for your application:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click **Create Page**.
4. Select **Login Page**.
5. Specify Login page attributes and click **Create**.

About Deep Linking

Deep linking refers to the ability to link to an Oracle Application Express page out of context (for example, from a hyperlink in an email or workflow notification). When you link to a page out of context and the application requires the user be authenticated, the user will be taken to the login page. After credentials verification, the Application Express engine automatically displays the page that was referenced in the original link. Deep linking is supported for applications that use authentication schemes.

Providing Security Through Authorization

Authorization is a broad term for controlling access to resources based on user privileges. While conditions control the rendering and processing of specific page controls or components, authorization schemes control user access to specific controls or components.

Topics in this section include:

- How Authorization Schemes Work
- Creating an Authorization Scheme
- Attaching an Authorization Scheme to an Application, Page, or Components
- Viewing Authorization Reports

How Authorization Schemes Work

An authorization scheme extends the security of your application's authentication scheme. You can specify an authorization scheme for an entire application, a page, or specific control such as a region, item, or button. For example, you could use an authorization scheme to selectively determine which tabs, regions, or navigation bars a user sees.

An authorization scheme either succeeds or fails. If a component or control level authorization scheme succeeds, the user can view the component or control. If it fails, the user cannot view the component or control. If an application or page level authorization scheme fails, then Oracle Application Express displays a previously defined message.

When you define an authorization scheme you give it a unique name. Once defined, you can attach it to any component or control in your application. To attach an authorization scheme to a component or control in your application, simply navigate to the appropriate attributes page and select an authorization scheme from the Authorization Scheme list.

Creating an Authorization Scheme

Before you can attach an authorization scheme to an application or an application component or control, you must first create it.

To create an authorization scheme:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application home page, click **Shared Components**.

   The Shared Components page appears.
4. Under Security, select Authorization Schemes.
5. Click **Create**.
6. Specify how to create an authorization scheme by selecting one of the following:
   - **From Scratch**
   - **As a Copy of an Existing Authorization Scheme**
7. Follow the on-screen instructions.

Editing Attributes of an Existing Authorization Scheme

To edit attributes of an existing authorization scheme:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application home page, click **Shared Components**.

   The Shared Components page appears.

4. Under Security, select Authorization Schemes.

   The Authorization Schemes page appears. By default, each scheme displays as an icon.

5. To access a detail view of all schemes, select **Details** from the View list.

   The Authorization Schemes page appears. You can change the appearance of the page by making a selection from the View list. Available options include:

   - **Icons** (the default) displays each authentication scheme as a large icon. To edit an authorization scheme, click the appropriate icon.
   - **Details** displays each application item as a line in a report. To edit an authorization scheme, select the scheme name.

About the Evaluation Point Attribute

You can specify when your authorization scheme is validated in the Evaluation Point attribute. You can choose to have your authorization scheme validated once for each session or once for each page view.

Keep in mind, that if you specify that an authorization scheme should be evaluated once for each session and the authorization scheme passes, the underlying code, test, or query will not be executed again for the duration of the application session. If your authorization scheme consists of a test whose results might change if evaluated at different times during the session, then you should specify that the evaluation point be once for each page view.

About Resetting Authorization Scheme State

If an authorization scheme is validated once for each session, Oracle Application Express caches the validation results in each user's session cache. You can reset a session's authorization scheme state by calling the `HTMLDB_UTIL.RESET_AUTHORIZATIONS` API.

Calling this procedure nulls out any previously cached authorization scheme results for the current session. Be aware that this procedure takes no arguments and is part of the publicly executable `HTMLDB_UTIL` package.

See Also: "RESET_AUTHORIZATIONS Procedure" on page 15-25

Attaching an Authorization Scheme to an Application, Page, or Components

Once you have created an authorization scheme you can attach it to an entire application, page, control, or component.

Topics in this section include:

- Attaching an Authorization Scheme to an Application
- Attaching an Authorization Scheme to a Page
- Attaching an Authorization Scheme to a Control or Component

Attaching an Authorization Scheme to an Application

To attach an authorization scheme to an application:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click the Edit Attributes icon.

The Application Attributes page appears.

4. Click the **Edit Security Attributes** icon.
5. Scroll down to Authorization and make a selection from the Authorization Scheme list.

Attaching an Authorization Scheme to a Page

To attach an authorization scheme to a page:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Select a page.
4. Click **Edit Attributes**.
5. Scroll down to Security and make a selection from the Authorization Scheme list.

Attaching an Authorization Scheme to a Control or Component

To attach an authorization scheme to a page component or control:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Select a page.
4. Click the name of the component or control to which you want to apply the authorization scheme.
5. Scroll down to Security and make a selection from the Authorization Scheme list.

Viewing Authorization Reports

You can use the Authorization Scheme Subscription and Authorization Scheme Utilization reports to better manage authorization schemes within your application.

To view authorization scheme reports:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application home page, click **Shared Components**.

   The Shared Components page appears.
4. Under Security, select **Authorization Schemes**.
5. Click the appropriate tab at the top of the page:
   - **Subscription**
   - **Utilization**

Subscription

Use the Authorization Scheme Subscription report to view details about authorization schemes subscription.

Utilization

Use the Authorization Scheme Utilization report to view details about authorization schemes utilization.

To view additional reports indicating which pages having authorization schemes and which do not, select one of the following from the Tasks list:

- **Report Pages With Authorization Schemes**
- **Report Pages Without Authorization Schemes**

13

Advanced Programming Techniques

This section provides information about advanced programming techniques including establishing database links, using collections, running background SQL, utilizing Web services, and managing user preferences.

This section contains the following topics:

See Also: "Oracle Application Express APIs" on page 15-1 and "Deploying an Application" on page 11-1

Accessing Data with Database Links

Because the Database Home Page runs on top of an Oracle database, you have access to all distributed Oracle database capabilities. Typically, you perform distributed database operations using database links.

A database link is a schema object in one database that enables you to access objects on another database. Once you have created the database link you can access the remote objects by appending `@dblink` to the table or view name where `dblink` is the Database Link Name you specify in the Create Database Object Wizard.

To create a database link:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. Click **Create**.
3. Select **Database Link** and click **Next**.
4. Follow the on-screen instructions.

   Note that Database Link names must conform to Oracle naming conventions and cannot contain spaces, or start with a number or underscore.

To view an existing a database link:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. Select the object type **Database Links** at the top of the page.

   See Also: "Managing Database Objects with Object Browser" on page 16-1 and "Database Links" in *Oracle Database Administrator's Guide*

Using Collections

Collections enable you to temporarily capture one or more nonscalar values. You can use collections to store rows and columns currently in session state so they can be accessed, manipulated, or processed during a user's specific session. You can think of a collection as a bucket in which you temporarily store and name rows of information.

The following are examples of when you might use collections:

- When you are creating a data-entry wizard in which multiple rows of information first need to be collected within a logical transaction. You can use collections to temporarily store the contents of the multiple rows of information, before performing the final step in the wizard when both the physical and logical transactions are completed.
- When your application includes an update page on which a user updates multiple detail rows on one page. The user can make many updates, apply these updates to a collection and then call a final process to apply the changes to the database.
- When you are building a wizard where you are collecting an arbitrary number of attributes. At the end of the wizard, the user then performs a task that takes the information temporarily stored in the collection and applies it to the database.

Topics in this section include:

- About the HTMLDB_COLLECTION API
- Creating a Collection
- Truncating a Collection
- Accessing a Collection
- Deleting a Collection
- Adding Members to a Collection
- Updating Collection Members
- Deleting Collection Members
- Determining Collection Status
- Merging Collections
- Managing Collections
- Clearing Collection Session State

About the HTMLDB_COLLECTION API

Every collection contains a named list of data elements (or members) which can have up to 50 attributes (or columns). You insert, update, and delete collection information using the PL/SQL API `HTMLDB_COLLECTION`.

About Collection Naming

When you create a new collection, you must give it a name that cannot exceed 255 characters. Note that collection names are not case-sensitive and will be converted to uppercase.

Once the collection is named, you can access the values in the collection by running a SQL query against the view `HTMLDB_COLLECTIONS`.

See Also: "Accessing a Collection" on page 13-4

Creating a Collection

Every collection contains a named list of data elements (or members) that can have up to 50 attributes (or columns). You use the following methods to create a collection:

- `CREATE_COLLECTION`
- `CREATE_OR_TRUNCATE_COLLECTION`
- `CREATE_COLLECTION_FROM_QUERY`
- `HTMLDB_COLLECTION.CREATE_COLLECTION_FROM_QUERY_B`

The `CREATE_COLLECTION` method raises an exception if the named collection already exists, for example:

```
HTMLDB_COLLECTION.CREATE_COLLECTION(
    p_collection_name => collection name );
```

The `CREATE_OR_TRUNCATE_COLLECTION` method creates a new collection if the named collection does not exist. If the named collection already exists, this method truncates it. Truncating a collection empties it, but leaves it in place, for example:

```
HTMLDB_COLLECTION.CREATE_OR_TRUNCATE_COLLECTION(
    p_collection_name => collection name );
    p_generate_md5    => YES or NO );
```

The `CREATE_COLLECTION_FROM_QUERY` method creates a collection and then populates it with the results of a specified query, for example:

```
HTMLDB_COLLECTION.CREATE_COLLECTION_FROM_QUERY(
    p_collection_name => collection name,
    p_query           => your query );
    p_generate_md5    => YES or NO );
```

The `CREATE_COLLECTION_FROM_QUERY_B` method also creates a collection and then populates it with the results of a specified query, for example:

```
HTMLDB_COLLECTION.CREATE_COLLECTION_FROM_QUERY_B(
    p_collection_name => collection name,
    p_query           => your query );
```

The `CREATE_COLLECTION_FROM_QUERY_B` method offers significantly faster performance than the `CREATE_COLLECTION_FROM_QUERY` method by performing bulk SQL operations, but has the following limitations:

- No column value in the select list of the query can be more than 2,000 bytes. If a row is encountered that has a column value of more than 2,000 bytes, an error will be raised during execution.
- The MD5 checksum will not be computed for any members in the collection.

About the Parameter p_generate_md5

Use the p_generate_md5 flag to specify if the message digest of the data of the collection member should be computed. By default, this flag is set to NO. Use this parameter to check the MD5 of the collection member (that is, compare it with another member or see if a member has changed).

> **See Also:** "Determining Collection Status" on page 13-7 for information about using the GET_MEMBER_MD5 function

Truncating a Collection

If you truncate a collection, you remove all members from the specified collection, but the named collection remains in place, for example:

```
HTMLDB_COLLECTION.TRUNCATE_COLLECTION(
    p_collection_name => collection name );
```

Accessing a Collection

You can access the members of a collection by querying the database view HTMLDB_COLLECTIONS. The HTMLDB_COLLECTIONS view has the following definition:

```
COLLECTION_NAME   NOT NULL VARCHAR2(255)
SEQ_ID            NOT NULL NUMBER
C001                       VARCHAR2(4000)
C002                       VARCHAR2(4000)
C003                       VARCHAR2(4000)
C004                       VARCHAR2(4000)
C005                       VARCHAR2(4000)
...
C050                       VARCHAR2(4000)
CLOB001                    CLOB
MD5_ORIGINAL               VARCHAR2(4000)
```

Use the HTMLDB_COLLECTIONS view in an application just as you would use any other table or view in an application, for example:

```
SELECT c001, c002, c003
   FROM htmldb_collections
 WHERE collection_name = 'FIREARMS'
```

Deleting a Collection

If you delete a collection, you delete the collection and all of its members, for example:

```
HTMLDB_COLLECTION.DELETE_COLLECTION (
    p_collection_name => collection name );
```

Be aware that if you do not delete a collection, it will eventually be deleted when the session is purged. For example:

Deleting All Collections for the Current Application

Use the DELETE_ALL_COLLECTIONS method to delete all collections defined in the current application, for example:

```
HTMLDB_COLLECTION.DELETE_ALL_COLLECTIONS;
```

Deleting All Collections in the Current Session

Use the `DELETE_ALL_COLLECTIONS_SESSION` method to delete all collections defined in the current session., for example:

```
HTMLDB_COLLECTION.DELETE_ALL_COLLECTIONS_SESSION;
```

Adding Members to a Collection

When data elements (or members) are added to a collection, they are assigned a unique sequence ID. As you add members to a collection, the sequence ID will change in increments of 1, with the newest members having the largest ID.

You add new members to a collection using the `ADD_MEMBER function`. Calling this function returns the sequence ID of the newly added member. The following example demonstrates how to use the `ADD_MEMBER` function.

```
HTMLDB_COLLECTION.ADD_MEMBER(
    p_collection_name => collection name,
    p_c001          => [member attribute 1],
    p_c002          => [member attribute 2],
    p_c003          => [member attribute 3],
    p_c004          => [member attribute 4],
    p_c005          => [member attribute 5],
    p_c006          => [member attribute 6],
    p_c007          => [member attribute 7],
  ...
    p_c050          => [member attribute 50]);
    p_clob001       => [CLOB member attribute 1],
    p_generate_md5  => YES or NO);
```

You can also add new members (or an array of members) to a collection using the `ADD_MEMBERS` method, for example:

```
HTMLDB_COLLECTION.ADD_MEMBERS(
    p_collection_name => collection name,
    p_c001          => member attribute array 1,
    p_c002          => member attribute array 2,
    p_c003          => member attribute array 3,
    p_c004          => member attribute array 4,
    p_c005          => member attribute array 5,
    p_c006          => member attribute array 6,
    p_c007          => member attribute array 7,
    ...
    p_c050          => member attribute array 50);
    p_generate_md5  => YES or NO);
```

This method raises an error if the specified collection does not exist with the specified name of the current user and in the same session. Also any attribute exceeding 4,000 characters will be truncated to 4,000 characters. The number of members added is based on the number of elements in the first array.

About the Parameters p_generate_md5 and p_clob001

Use the `p_generate_md5` flag to specify if the message digest of the data of the collection member should be computed. By default, this flag is set to `NO`. Use this parameter to check the MD5 of the collection member (that is, compare it with another member or see if a member has changed).

Use `p_clob001` for collection member attributes which exceed 4,000 characters.

See Also: "Determining Collection Status" on page 13-7 for information about using the function GET_MEMBER_MD5

Updating Collection Members

You can update collection members by calling the UPDATE_MEMBER procedure and referencing the desired collection member by its sequence ID, for example:

```
HTMLDB_COLLECTION.UPDATE_MEMBER (
    p_collection_name => collection name,
    p_seq             => member sequence number,
    p_c001            => member attribute 1,
    p_c002            => member attribute 2,
    p_c003            => member attribute 3,
    p_c004            => member attribute 4,
    p_c005            => member attribute 5,
    p_c006            => member attribute 6,
    p_c007            => member attribute 7,
    ...
    p_c050            => member attribute 50);
    p_clob001         => [CLOB member attribute 1],
```

The UPDATE_MEMBER procedure replaces an entire collection member, not individual member attributes. This procedure causes an error if the named collection does not exist. For example:

Use the p_clob001 parameter for collection member attributes which exceed 4,000 characters.

If you want to update a single attribute of a collection member, use the UPDATE_MEMBER_ATTRIBUTE procedure, for example:

```
HTMLDB_COLLECTION.UPDATE_MEMBER_ATTRIBUTE(
    p_collection_name       => collection_name,
    p_seq                   => member sequence number,
    p_attr_number           => member attribute number,
    p_attr_value            => member attribute value )

HTMLDB_COLLECTION.UPDATE_MEMBER_ATTRIBUTE(
    p_collection_name        => collection_name,
    p_seq                    => member sequence number,
    p_clob_number            => CLOB member attribute number,
    p_clob_value             => CLOB member attribute value );
```

Calling the UPDATE_MEMBER_ATTRIBUTE procedure causes an error if the named collection does not exist.

Note that the only valid value for the p_clob_number parameter is 1.

Deleting Collection Members

You can delete a collection member by calling the DELETE_MEMBER procedure and referencing the desired collection member by its sequence ID, for example:

```
HTMLDB_COLLECTION.DELETE_MEMBER(
    p_collection_name => collection name,
    p_seq             => member sequence number);
```

Note that this procedure leaves a gap in the sequence IDs in the specified collection. In addition, calling this procedure causes an error if the named collection does not exist.

You can also delete all members from a collection by when an attribute matches a specific value, for example:

```
HTMLDB_COLLECTION.DELETE_MEMBERS(
    p_collection_name => collection name,
    p_attr_number     => number of attribute used to match for the specified
                         attribute value for deletion,
    p_attr_value      => attribute value of the member attribute used to
                         match for deletion);
```

Note that the `DELETE_MEMBER` procedure also leaves a gap in the sequence IDs in the specified collection. This procedure causes an error if:

- The named collection does not exist.
- The specified attribute number is outside the range of 1 to 50, or not valid.

If the supplied attribute value is null, then all members of the named collection will deleted.

Determining Collection Status

The `p_generate_md5` parameter determines if the MD5 message digests are computed for each member of a collection. The collection status flag is set to `FALSE` immediately after you create a collection. If any operations are performed on the collection (such as add, update, truncate, and so on), this flag is set to `TRUE`.

You can reset this flag manually by calling `RESET_COLLECTION_CHANGED`, for example:

```
HTMLDB_COLLECTION.RESET_COLLECTION_CHANGED (
    p_collection_name => collection name)
```

Once this flag has been reset, you can determine if a collection has changed by calling `COLLECTION_HAS_CHANGED`, for example:

```
l_changed := HTMLDB_COLLECTION.COLLECTION_HAS_CHANGED(
  p_collection_name => collection_name);
```

When you add a new member to a collection, an MD5 message digest is computed against all 50 attributes and the CLOB attribute if the `p_generated_md5` parameter is set to `YES`. You can access this value from the `MD5_ORIGINAL` column of the view `HTMLDB_COLLECTION`. You can access the MD5 message digest for the current value of a specified collection member by using the function `GET_MEMBER_MD5`. For example:

```
HTMLDB_COLLECTION.GET_MEMBER_MD5 (
    p_collection_name => collection name,
    p_seq             => member sequence number );
    RETURN VARCHAR2;
```

Merging Collections

You can merge members of a collection with values passed in a set of arrays. By using the `p_init_query` argument, you can create a collection from the supplied query, for example:

```
HTMLDB_COLLECTION.MERGE_MEMBERS
p_collection_name => collection_name
```

Note that if the collection exists, the following occurs:

- Rows in the collection (not in the arrays) will be deleted.
- Rows in the collection and in the arrays will be updated.
- Rows in the array and not in the collection will be inserted.

Any attribute value exceeding 4,000 characters will be truncated to 4,000 characters. Table 13–1 describes the available arguments you can use when merging collections.

Table 13–1 Available Arguments for Merging Collections

Argument	Description
`p_c001`	Array of first attribute values to be merged. Maximum length is 4,000 characters. If the maximum length is greater, it will be truncated to 4,000 characters. The count of elements in the P_C001 PL/SQL table is used as the total number of items across all PL/SQL tables. For example, if P_C001.count = 2 and P_C002.count = 10, only 2 members will be merged. Note that if P_C001 is null, an application error will be raised.
`p_c0xx`	Attribute of `XX` attributes values to be merged. Maximum length is 4,000 characters. If the maximum length is greater, it will be truncated to 4,000 characters.
`p_collection_name`	Name of the collection. **See Also:** "About Collection Naming" on page 13-3
`p_null_index`	Use this argument to identify rows the merge function should ignore. This argument identifies an row as null. Null rows are automatically removed from the collection.
`p_null_value`	Use this argument in conjunction with the `p_null_index`. Identifies the null value. If used this value cannot be null. A typical value for this argument is 0.
`p_init_query`	Use the query defined by this argument to create a collection if the collection does not exist.

Managing Collections

You can use the following utilities to manage collections.

Topics in this section include:

- Obtaining a Member Count
- Resequencing a Collection
- Verifying Whether a Collection Exists
- Adjusting a Member Sequence ID
- Sorting Collection Members

Obtaining a Member Count

Use `COLLECTION_MEMBER_COUNT` to return the total count of all members in a collection. Note that this count does not indicate the highest sequence in the collection, for example:

```
l_count := HTMLDB_COLLECTION.COLLECTION_MEMBER_COUNT (
  p_collection_name => collection name );
```

Resequencing a Collection

Use RESEQUENCE_COLLECTION to resequence a collection to remove any gaps in sequence IDs while maintaining the same element order, for example:

```
HTMLDB_COLLECTION.RESEQUENCE_COLLECTION (
   p_collection_name => collection name )
```

Verifying Whether a Collection Exists

Use COLLECTION_EXISTS to determine if a collection exists, for example:

```
l_exists := HTMLDB_COLLECTION.COLLECTION_EXISTS  (
  p_collection_name => collection name );
```

Adjusting a Member Sequence ID

You can adjust the sequence ID of a specific member within a collection by moving the ID up or down. When you adjust a sequence ID, the specified ID is exchanged with another ID. For example, if you were to move the ID 2 up, 2 becomes 3, and 3 would become 2.

Use MOVE_MEMBER_UP to adjust a member sequence ID up by one. Alternately, use MOVE_MEMBER_DOWN to adjust a member sequence ID down by one, for example:

```
HTMLDB_COLLECTION.MOVE_MEMBER_DOWN(
    p_collection_name => collection name,
    p_seq             => member sequence number);
```

Note that while using either of these methods an application error displays:

- If the named collection does not exist for the current user in the current session
- If the member specified by the p_seq sequence ID does not exist

However, an application error will not be returned if the specified member already has the highest or lowest sequence ID in the collection (depending on if you are calling MOVE_MEMBER_UP or MOVE_MEMBER_DOWN).

Sorting Collection Members

Use the SORT_MEMBERS method to reorder members of a collection by the column number. This method sorts the collection by a particular column number and also reassigns the sequence IDs for each member to remove gaps, for example:

```
HTMLDB_COLLECTION.SORT_MEMBERS(
    p_collection_name       => collection name,
    p_sort_on_column_number => column number to sort by);
```

Clearing Collection Session State

Clearing the session state of a collection removes the collection members. A shopping cart is a good example of when you might need to clear collection session state. When a user requests to empty the shopping cart and start again, you need to clear the session state for a collection. You can remove session state of a collection by calling the CREATE_OR_TRUNCATE_COLLECTION method or by using f?p syntax.

Calling the CREATE_OR_TRUNCATE_COLLECTION method deletes the existing collection and then recreates it, for example:

```
HTMLDB_COLLECTION.CREATE_OR_TRUNCATE_COLLECTION(
    p_collection_name       => collection name,
```

You can also use the sixth `f?p` syntax argument to clear session state, for example:

```
f?p=App:Page:Session::NO:1,2,3,collection name
```

See Also: "Understanding URL Syntax" on page 3-10

Creating Custom Activity Reports Using HTMLDB_ACTIVITY_LOG

The `HTMLDB_ACTIVITY_LOG` view records all activity in a workspace (or database user account), including developer activity and application runtime activity. You can use `HTMLDB_ACTIVITY_LOG` to view to query all activity for the current workspace. For example, you can use this view to develop monitoring reports within a specific application to provide real-time performance statistics.

Table 13–2 describes the columns in the `HTMLDB_ACTIVITY_LOG` view.

Table 13–2 Columns in HTMLDB_ACTIVITY_LOG

Column	Type	Description
`time_stamp`	DATE	Date and time that activity was logged at the end of the page view.
`component_type`	VARCHAR2(255)	Reserved for future use.
`component_name`	VARCHAR2(255)	Reserved for future use.
`component_attribute`	VARCHAR2(4000)	Title of page.
`information`	VARCHAR2(4000)	Reserved for future use.
`elap`	NUMBER	Elapsed time of page view in seconds.
`num_rows`	NUMBER	Number of rows processed on page.
`userid`	VARCHAR2(255)	User ID performing page view.
`ip_address`	VARCHAR2(4000)	IP address of client.
`user_agent`	VARCHAR2(4000)	Web browser user agent of client.
`flow_id`	NUMBER	Application ID.
`step_id`	NUMBER	Page ID.
`session_id`	NUMBER	Oracle Application Express session identifier.

To conserve space in the activity log, only the first log entry of each unique session will contain the IP address and Web browser user agent.

The following example demonstrates how to create a report that displays the total number of page views and the average page view time in the past 24 hours for application 9529, and grouped by userid:

```
SELECT COUNT(*), AVG(elap), userid
    FROM HTMLDB_ACTIVITY_LOG
 WHERE time_stamp > (SYSDATE-1)
   AND flow_id = 9529
GROUP BY userid
```

Running Background PL/SQL

You can use the HTMLDB_PLSQL_JOB package to run PL/SQL code in the background of your application. This is an effective approach for managing long running operations that do not need to complete for a user to continue working with your application.

Topics in this section include:

- Understanding the HTMLDB_PLSQL_JOB Package
- About System Status Updates
- Using a Process to Implement Background PL/SQL

Understanding the HTMLDB_PLSQL_JOB Package

HTMLDB_PLSQL_JOB is a wrapper package around DBMS_JOB functionality offered in the Oracle database. Note that the HTMLDB_PLSQL_JOB package only exposes that functionality which is necessary to run PL/SQL in the background. The following is a description of the HTMLDB_PLSQL_JOB package:

```
SQL> DESC HTMLDB_PLSQL_JOB
FUNCTION JOBS_ARE_ENABLED RETURNS BOOLEAN
PROCEDURE PURGE_PROCESS
 Argument Name                  Type                    In/Out Default?
 ------------------------------ ----------------------- ------ --------
 P_JOB                          NUMBER                  IN
FUNCTION SUBMIT_PROCESS RETURNS NUMBER
 Argument Name                  Type                    In/Out Default?
 ------------------------------ ----------------------- ------ --------
 P_SQL                          VARCHAR2                IN
 P_WHEN                         VARCHAR2                IN     DEFAULT
 P_STATUS                       VARCHAR2                IN     DEFAULT
FUNCTION TIME_ELAPSED RETURNS NUMBER
 Argument Name                  Type                    In/Out Default?
 ------------------------------ ----------------------- ------ --------
 P_JOB                          NUMBER                  IN
PROCEDURE UPDATE_JOB_STATUS
 Argument Name                  Type                    In/Out Default?
 ------------------------------ ----------------------- ------ --------
 P_JOB                          NUMBER                  IN
 P_STATUS                       VARCHAR2                IN
 P_DESC
```

Table 13–1 describes the functions available in the HTMLDB_PLSQL_JOB package.

Table 13–3 HTMLDB_PLSQL_JOB Package: Available Functions

Function or Procedure	Description
SUBMIT_PROCESS	Use this procedure to submit background PL/SQL. This procedure returns a unique job number. Because you can use this job number as a reference point for other procedures and functions in this package, it may be useful to store it in your own schema.
UPDATE_JOB_STATUS	Call this procedure to update the status of the currently running job. This procedure is most effective when called from the submitted PL/SQL.
TIME_ELAPSED	Use this function to determine how much time has elapsed since the job was submitted.

Table 13–3 (Cont.) HTMLDB_PLSQL_JOB Package: Available Functions

Function or Procedure	Description
JOBS_ARE_ENABLED	Call this function to determine whether or not the database is currently in a mode that supports submitting jobs to the HTMLDB_PLSQL_JOB package.
PURGE_PROCESS	Call this procedure to clean up submitted jobs. Submitted jobs stay in the HTMLDB_PLSQL_JOBS view until either Oracle Application Express cleans out those records, or you call PURGE_PROCESS to manually remove them.

You can view all jobs submitted to the HTMLDB_PLSQL_JOB package using the HTMLDB_PLSQL_JOBS view. The following is the description of HTMLDB_PLSQL_JOBS view:

```
SQL> DESCRIBE HTMLDB_PLSQL_JOBS
 Name                              Null?    Type
 --------------------------------- -------- ----------------------------
 ID                                         NUMBER
 JOB                                        NUMBER
 FLOW_ID                                    NUMBER
 OWNER                                      VARCHAR2(30)
 ENDUSER                                    VARCHAR2(30)
 CREATED                                    DATE
 MODIFIED                                   DATE
 STATUS                                     VARCHAR2(100)
 SYSTEM_STATUS                              VARCHAR2(4000)
 SYSTEM_MODIFIED                            DATE
 SECURITY_GROUP_ID                          NUMBER
```

Table 13–4 describes the columns available in HTMLDB_PLSQL_JOBS view.

Table 13–4 HTMLDB_PLSQL_JOBS View Columns

Name	Description
ID	An unique identifier for each row.
JOB	The job number assigned to each submitted PL/SQL job. The HTMLDB_PLSQL_JOB.SUBMIT_PROCESS function returns this value. This is also the value you pass into other procedures and functions in the HTMLDB_PLSQL_JOB package.
FLOW_ID	The application from which this job was submitted.
OWNER	The database schema that owns the application. This identifies what schema will parse this code when DBMS_JOB runs it.
ENDUSER	The end user (that is, who logged into the application) that caused this process to be submitted.
CREATED	The date when the job was submitted.
MODIFIED	The date when the status was modified.
STATUS	The user-defined status for this job. Calling HTMLDB_PLSQL_JOB.UPDATE_JOB_STATUS updates this column.
SYSTEM_STATUS	The system defined status for this job.
SYSTEM_MODIFIED	The date when the system status was modified.

Table 13–4 (Cont.) HTMLDB_PLSQL_JOBS View Columns

Name	Description
`SECURITY_GROUP_ID`	The unique ID assigned to your workspace. Developers can only see jobs submitted from their own workspace.

About System Status Updates

Submitted jobs can contain any of the following system status settings:

- **SUBMITTED** indicates the job has been submitted, but has not yet started. The DBMS_JOB does not guarantee immediate starting of jobs.
- **IN PROGRESS** indicates that the `DBMS_JOB` has started the process.
- **COMPLETED** indicates the job has finished.
- **BROKEN (sqlcode) sqlerrm** indicates there was a problem in your job that resulted in an error. The SQL code and SQL error message for the error should be included in the system status. Review this information to determine what went wrong.

Using a Process to Implement Background PL/SQL

The simplest way to implement the `HTMLDB_PLSQL_JOB` package is to create a page process that specifies the process type `PLSQL DBMS JOB`. By selecting this process type, Application Builder will submit the PL/SQL code you specify as a job. Because you are not calling the function directly, you can use the `APP_JOB` built-in substitution item to determine the job number of any jobs you submit.

The following example runs a PL/SQL job in the background for testing and explanation:

```
001  BEGIN
002    FOR i IN 1 .. 100 LOOP
003      INSERT INTO emp(a,b) VALUES (:APP_JOB,i);
004      IF MOD(i,10) = 0 THEN
005        HTMLDB_PLSQL_JOB.UPDATE_JOB_STATUS(
006          P_JOB     => :APP_JOB,
007          P_STATUS  => i || 'rows inserted');
008      END IF;
009      HTMLDB_UTIL.PAUSE(2);
010    END LOOP;
011  END;
```

In this example, note that:

- Lines 002 to 010 run a loop that inserts 100 records into the `emp` table.
- `APP_JOB` is referenced as a bind variable inside the VALUES clause of the INSERT, and specified as the `P_JOB` parameter value in the call to `UPDATE_JOB_STATUS`.
- `APP_JOB` represents the job number which will be assigned to this process as it is submitted to `HTMLDB_PLSQL_JOB`. By specifying this reserved item inside your process code, it will be replaced for you at execution time with the actual job number.
- Note that this example calls to `UPDATE_JOB_STATUS` every ten records, INSIDE the block of code. Normally, Oracle transaction rules dictate updates made inside code blocks will not be seen until the entire transaction is committed. The `HTMLDB_PLSQL_JOB.UPDATE_JOB_STATUS` procedure, however, has been

implemented in such a way that the update will happen regardless of whether or not the job succeeds or fails. This last point is important for two reasons:

1. Even if your status shows "100 rows inserted," it does not mean the entire operation was successful. If an error occurred at the time the block of code tried to commit, the `user_status` column of `HTMLDB_PLSQL_JOBS` would not be affected because status updates are committed separately.
2. Updates are performed autonomously. You can view the job status before the job has completed. This gives you the ability to display status text about ongoing operations in the background as they are happening.

Implementing Web Services

Web services enable applications to interact with one another over the Web in a platform-neutral, language independent environment. In a typical Web services scenario, a business application sends a request to a service at a given URL by using the protocol over HTTP. The service receives the request, processes it, and returns a response. You can incorporate calls with external Web services in application developed in Application Builder

Web services are based on Simple Object Access Protocol (SOAP). SOAP is a World Wide Web Consortium (W3C) standard protocol for sending and receiving requests and responses across the Internet. SOAP messages can be sent back and forth between a service provider and a service user in SOAP envelopes.

SOAP offers two primary advantages:

- SOAP is based on XML, and therefore easy to use.
- SOAP messages are not blocked by firewalls because this protocol uses simple transport protocols, such as HTTP.

Topics in this section include:

- Understanding Web Service References
- Creating a Web Service Reference
- Using the Web Service Reference Repository
- Testing a Web Service Reference
- Creating an Input Form and Report on a Web Service
- Creating a Form on a Web Service
- Invoking a Web Service as a Process
- Editing a Web Service Process
- Viewing a Web Service Reference History

Note: The SOAP 1.1 specification is a W3C note. (The W3C XML Protocol Working Group has been formed to create a standard that will supersede SOAP.)

For information about Simple Object Access Protocol (SOAP) 1.1 see:

`http://www.w3.org/TR/SOAP/`

Understanding Web Service References

To utilize Web services in Oracle Application Express, you create a Web service reference using a wizard. Each Web service reference is based on a Web Services Description Language (WSDL) document that describes the target Web service. When you create a Web service reference, the wizard analyzes the WSDL and collects all the necessary information to create a valid SOAP message, including:

- The URL used to post the SOAP request over HTTP
- A Universal Resource Identifier (URI) identifying the SOAP HTTP request
- Operations of the Web Service
- Input parameters for each operation
- Output parameters for each operation

Accessing the Web Service References Page

You manage Web service references on the Web Service References page.

To access the Web Service References page:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.

   Application Builder appears.
3. Click **Shared Components**.

   The Shared Components page appears.
4. Under Logic, select **Web Service References**.

   The Web Service References page appears.

Specifying an Application Proxy Server Address

If your environment requires a proxy server to access the Internet, you must specify a proxy server address on the Application Attributes page before you can create a Web service reference.

To specify a proxy address for an application:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.

   Application Builder appears.
3. Click Edit Attributes.
4. Click **Edit Standard Attributes**.
5. Under Application Definition, enter the proxy server in Proxy Server.
6. Click **Apply Changes**.

Creating a Web Service Reference

When you create a Web service reference, you need to decide how to locate the WSDL. You can locate a WSDL in two ways:

- By searching a Universal Description, Discovery and Integration (UDDI) registry
- by entering the URL to the WSDL document

AUDDI registry is a directory where businesses register their Web services.

Creating a Web Service Reference by Searching a UDDI Registry

To create a new Web service by searching a UDDI registry:

1. Navigate to the Web Service References page:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under Logic, select **Web Service References**.

      The Web Service References page appears.
2. Click **Create**.
3. When prompted to search a UDDI registry to find a WSDL, click **Yes**.
4. For UDDI Location you can either:
   - Enter a URL endpoint to a UDDI registry.
   - Click the **List** icon and select a UDDI registry.
5. Under Search for Services, specify whether to search for a business name or a service name.
   a. For Search Type, specify whether to search for a business name or a service name. You cannot search for both.
   b. In Name, enter the business name or service name to search for.
   c. Optionally indicate if the search should be case-sensitive or an exact match. Use the percent (%) symbol as a wildcard character.
   d. Click **Search**.
   e. When the search results appear, make a selection and click **Next**.

   A summary page appears describing the selected Web service.
6. Review your selection and click **Next** to continue.

   The URL to the WSDL document displays in the WSDL Location field.
7. Click **Finish**.

The Web service reference is added to the Web Service References Repository.

Creating a Web Service Reference by Specifying a WSDL Document

To create a new Web service by specifying a URL to a specific WSDL document:

1. Navigate to the Web Service References page:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under Logic, select **Web Service References**.

      The Web Service References page appears.
2. Click **Create**.

3. When prompted to search a UDDI registry to find a WSDL, click **No**.
4. In WSDL Location, enter the URL to the WSDL document.
5. Click **Finish**.

The Web service reference is added to the Web Service References Repository.

Using the Web Service Reference Repository

Web service references are stored in the Web Service Reference Repository.

To access the Web Service References Repository:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.

   Application Builder appears.
3. Click **Shared Components**.

   The Shared Components page appears.
4. Under Logic, select **Web Service References**.

   The Web Service Reference page appears.

You can change the appearance of the page by making a selection from the View list. Available options include:

- **Icons** (the default) displays each Web service reference as a large icon. To edit a Web service reference, click the appropriate icon.
- **Details** displays each application item as a line in a report.

  In Details view you can:

  - Edit a reference by clicking the **Edit** icon.
  - Test a reference by clicking the **Run** icon.
  - View details about a reference, by clicking the reference name.

Testing a Web Service Reference

After you have created a Web service reference, you can test it on the Test Web Service Reference page.

To test a Web service reference:

1. Navigate to the Web Service References page:
   a. On the Database Home Page, click the **Application Builder** icon.
   b. Select an application.
   c. Click **Shared Components**.
   d. Under Logic, select **Web Service References**.

      The Web Service References page appears.
2. From View, select **Details**.
3. Click the **Run** icon adjacent to the Web Service reference name.

   The Test Web Service Reference page appears. The Web service name and URL endpoint display at the top of the page.

4. From Operation, select an operation (that is, the method to be executed).
5. Under Input Parameters, enter the appropriate value.
6. Click **Test**.

   The message request and response appear at the bottom of the page.

Creating an Input Form and Report on a Web Service

The Create Form and Report on Web Service Wizard creates an input form, a submit button, and a report for displaying results. You can execute this wizard directly after creating the Web service reference, or by adding a new page.

Use this wizard when you expect a nonscalar result from the Web service. The Amazon Web service is a good example. This Web service returns many results based on the search criteria entered in an input form.

Creating a Form and Report After Creating a Reference

To create a form and report after creating a Web Service Reference:

1. Create the Web service reference. See "Creating a Web Service Reference" on page 13-15.
2. After the Web service reference has been added, select **Create Form and Report on Web Service**.
3. For Web Service Reference and Operation, select the Web service reference and operation (that is, the method to be executed).
4. For Identify Page and Region Attributes, review the page and region attributes. If the page you specify does not exist, the wizard creates the page for you.
5. For Items for Input Parameters:
   a. Identify which items to add to the form. To include an item, select **Yes** in the Create column. Otherwise, select **No**.
   b. If necessary, edit the item label.
6. For Base Node:
   a. In Temporary Result Set Name, enter a name for the collection that stores the Web service result.
   b. For Result Tree to Report On, select the portion of the resulting XML document that contains the information you want to include in the report.
7. For Result Parameters to Display, select the parameters to be included in the report.
8. Click **Finish**.

Creating a Form and Report by Adding a New Page

If you have an existing Web service reference, you can create an input form and report by adding a new page.

To create a form and report by adding a new page:

1. Create the Web service reference. See "Creating a Web Service Reference" on page 13-15.
2. Create a new page. See "Adding Pages to an Application" on page 5-8.

In the Create Page Wizard:

   a. Select **Form**.

   b. Select **Form and Report on Web Service**.

3. For Web Service Reference and Operation, select the Web service reference and operation (that is, the method to be executed).

4. For Identify Page and Region Attributes, review the page and region attributes. If the page you specify does not exist, the wizard creates the page for you.

5. For Items for Input Parameters:

   a. Identify which items to add to the form. To include an item, select **Yes** in the Create column. Otherwise, select **No**.

   b. If necessary, edit the item label.

6. For Base Node:

   a. In Temporary Result Set Name, enter a name for the collection that stores the Web service result.

   b. In Result Tree to Report On, select the portion of the resulting XML document that contains the information you want to include in the report.

7. For Result Parameters to Display, select the parameters to be included in the report.

8. Click **Finish**.

Creating a Form on a Web Service

The Create Form on Web Service Wizard creates a form and a submit button. You can execute this wizard after creating the Web service reference, or from the Page Definition.

Use this wizard when you expect a scalar result from the Web service. A Web service that looks up a stock price is a good example because the input is a stock symbol and the output is the scalar value price.

Creating a Form After Creating a Reference

To create a form after creating a Web Service Reference:

1. Create the Web service reference. See "Creating a Web Service Reference" on page 13-15.

2. After the Web service references has been added, select **Create Form on Web Service**.

3. For Web Service Reference and Operation, select the Web service reference and operation (that is, the method to be executed).

4. For Identify Page and Region Attributes, review the page and region attributes. If the page you specify does not exist, the wizard creates the page for you.

5. For Items for Input Parameters:

   a. Identify which items to add. To include an item, select **Yes** in the Create column. Otherwise, select **No**.

   b. If necessary, edit the item label.

6. For Items for Output Parameters:

   a. Identify which items need to be added. To include an item, select **Yes** in the Create column. Otherwise, select **No**.
   b. If necessary, edit the item label.
7. Click **Finish**.

Creating a Form by Adding a New Page

If you have an existing Web service reference, you can create form by adding a new page.

To create a form by adding a new page:

1. Create the Web service reference. See "Creating a Web Service Reference" on page 13-15.
2. Create a new page. See "Adding Pages to an Application" on page 5-8.

   In the Create Page Wizard:

   a. Select **Form**.
   b. Select **Form on Web Service**.
3. For Web Service Reference and Operation, select the Web service reference and operation (that is, the method to be executed).
4. For Identify Page and Region Attributes, review the page and region attributes. If the page you specify does not exist, the wizard creates the page for you.
5. For Items for Input Parameters:
   a. Identify which items need to be added. To include an item, select **Yes** in the Create column. Otherwise, select **No**.
   b. If necessary, edit the item label.
6. For Items for Output Parameters:
   a. Identify which items need to be added. To include an item, select **Yes** in the Create column. Otherwise, select **No**.
   b. If necessary, edit the item label.
7. Click **Finish**.

Invoking a Web Service as a Process

You can also implement a Web service as a process on the page. Running the process submits the request to the service provider. You can then display the request results in report.

To invoke a Web service as a process:

1. Create a new page. See "Adding Pages to an Application" on page 5-8.

   In the Create Page Wizard:

   a. Select **Blank Page**.
   b. When prompted to use tabs, select **No**.
2. Navigate to the Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.

   c. Select an application.
   d. Select a page.

      The Page Definition appears.

3. Under Page Rendering, Processes, click the **Create** icon.

   The Create Page Processes Wizard appears.

4. From the process category, select **Web Services**.
5. Specify a process name, sequence, and processing point.
6. Select the Web service reference and operation (that is, the method to be executed).
7. Define the process. You can store the results in a collection or in items on the page by selecting options under Web Service Output Parameters.
   a. To store the results in a collection:
      – For Store Result in, select **Collection**.
      – Enter a name for the collection in the value field.
   b. To store the results in items on the page:
      – For Store Result in, select **Items**.
      – Enter the appropriate items value in the fields provided.
8. Click **Create Process**.

Displaying Web Service Results in a Report

To create a report in which to display Web Service request results:

1. Navigate to the Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.

2. Under Regions, click the **Create** icon.

   The Create Region Wizard appears.

3. For the region type, select **Report**.
4. For the report implementation, select **Report on collection containing Web service result**.
5. On Identify Region Attributes, enter a region title and optionally edit the region attributes.
6. For Web Service Reference and Operation, select a Web service reference and an operation (that is, the method to be executed).
7. For Result Tree to Report On, select the portion of the resulting XML document that contains the information you want to include in the report.
8. For Result Parameters:
   a. In Temporary Result Set Name, enter a name for the collection that stores the Web service result.

   b. Select and deselect the appropriate parameters.

9. Click **Create SQL Report**.

Editing a Web Service Process

After you create a process of type Web service, you can map input parameters to a static value (for example to pass a key) by editing the Web service process.

To edit a Web service process:

1. Create a Web service process. See "Invoking a Web Service as a Process" on page 13-20.
2. Navigate to the Page Definition containing the Web service process.
3. Select the process name.

   The Edit Page Process page appears.
4. To map an input parameter to a static value:
   a. Scroll down to Web Service Input Parameters.
   b. Enter a value in the Value field, adjacent to the appropriate parameter name.
5. Click **Apply Changes**.

Viewing a Web Service Reference History

The Web Services History displays changes to Web service references for the current application by application ID, Web service references name, developer, and date.

To view a history of Web service reference changes:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.

   Application Builder appears.
3. Click **Shared Components**.

   The Shared Components page appears.
4. Under Logic, select **Web Service References**.
5. Click **History**.

Note: The History button only appears on the Web Service Reference page after you have created a Web service reference.

14 Managing Application Globalization

This section describes how to translate an application built-in Application Builder.

This section contains the following topics:

- About Translating an Application and Globalization Support
- Specifying the Primary Language for an Application
- Understanding the Translation Process
- Translating Messages Used in PL/SQL Procedures
- Translating Data That Supports List of Values
- About Supported Globalization Codes

> **See Also:** "Working in a Global Environment" in *Oracle Database Express Edition 2 Day Developer Guide*

About Translating an Application and Globalization Support

You can develop applications in Application Builder that can run concurrently in different languages. A single Oracle database instance and Oracle Application Express can support multiple database sessions customized to support different languages.

In general, translating an application built-in Application Builder involves the following steps:

- Map primary and target application IDs
- Seed and export text to a file for translation
- Translate the text in the file
- Apply and publish the translated file

> **See Also:** "Understanding the Translation Process" on page 14-6

Topics in this section include:

- About Language Identification
- Rules for Translating Applications Built in Application Builder
- How Translated Applications Are Rendered
- About Translatable Components

About Language Identification

After you create an application, you specify a language preference on the Edit Application Attributes page. Under Globalization, you select a primary application language and select how the Application Express engine determines the application language. You can specify to have the application language based on the user's browser language preference, an application preference, or an item preference.

See Also: "Specifying the Primary Language for an Application" on page 14-4

Rules for Translating Applications Built in Application Builder

Use the following rules to determine which translated version to use:

- Look for an exact match between the user language preference and the language code of the translated application.
- Look for a truncated match. That is, see if the language and locale exist. For example, if the user language preference is `en-us` and the translated version of `en-us` does not exist, look for a translated application that has the language code `en`.
- Use the primary application language.

For example, suppose you create an application with the primary language of German, `de`, and you create a translated version of the application with a language code of `en-us`. Users accessing this application with a browser language of `en-us` execute the English `en-us` version of the application. Users accessing the application with a browser language of `en-gb` view the application in the application's primary language. In this example, these users see the application in German, which is the application's primary language. For this example, you should create the translated English version using language code `en` to encompass all variations of `en`.

How Translated Applications Are Rendered

After Oracle Application Express determines the language for an application, the Application Express engine alters the database language for a specific page request. It then looks for a translated application in the appropriate language. If the Application Express engine finds that language, it render the application using that definition. Otherwise, it renders the application in the base (or primary) application language.

Note that the text that displays within an application is not translated on the fly. Oracle Application Express dynamically collects page attributes from either a base language application definition or an alternative application definition.

See Also: "About Dynamic Translation Text Strings" on page 14-3 and "Translating Data That Supports List of Values" on page 14-12

About Translatable Components

When you build an application in Application Builder, you define a large number of declarative attributes such as field labels, region headings, page header text, and so on. Using the steps described in this section, you can make all the application definition attributes within your application translatable.

About Shortcuts that Support Translatable Messages

Application Builder includes two shortcut types that enable you to reference translatable messages:

- **Message**. Use this shortcut to reference a translatable message at run time. Note that the name of the shortcut must match the corresponding message name. At run time, the name of the shortcut expands to the text of the translatable message for the current language.
- **Message with JavaScript Escaped Single Quotes**. Use this shortcut to reference a shortcut inside of a JavaScript literal string and reference a translatable message at run time. This shortcut defines a text string. When the shortcut is referenced, it escapes the single quotation marks required for JavaScript.

See Also: "Using Shortcuts" on page 5-63

About Messages

If your application includes PL/SQL regions or PL/SQL processes, you may need to translate any generated HTML or text. Within Application Builder, these types of generated HTML and text are called messages. You can define all messages and translate them on the Translatable Messages page. You can use the `HTMLDB_LANG.MESSAGE` API to translate text strings from PL/SQL stored procedures, functions, triggers, packaged procedures, and functions.

See Also: "Translating Messages Used in PL/SQL Procedures" on page 14-10

About Dynamic Translation Text Strings

Dynamic translations are used for database data that needs to be translated at run time. For example, you might use a dynamic translation to translate a list of values based on a database query. A dynamic translation consists of a translate-from language string, a language code, and a translate-to string. You can also use the `HTMLDB_LANG.LANG` API to retrieve dynamic translations programmatically.

See Also: "Translating Data That Supports List of Values" on page 14-12

About Translating Region Titles

By default, page region titles are included in the generated translation file. However, you can mark a region title as not translatable.

To mark a region title as not translatable:

1. Navigate to the Page Definition:
   a. Navigate to the Database Home Page.
   b. Click the **Application Builder** icon.
   c. Select an application.
   d. Select a page.

      The Page Definition appears.
2. On the Page Definition, select the region title.

   The Edit Region page appears.
3. Select the **exclude title from translation** check box.

About Translating Templates

By default, templates are not translatable, and therefore are not included in the generated translation file. Generally, templates do not and should not contain translatable text. However, if you need to mark a template as translatable, select the Translatable check box on the Edit Page Template page.

To identify a template as translatable:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. On the Application home page, click **Shared Components**.
4. Under User Interface, select **Templates**.

   The Templates page appears.
5. Locate the template you want to edit and select the template name.
6. Under Template Identification, select **Translatable**.

You can include translatable text at the application-level by defining the translatable text using static substitution strings. Because application-level attributes are translated, any text defined as a static substitution string will be included in the generated translation file.

See Also:

- "Editing Templates" on page 7-20
- "Substitutions" on page 4-10

Specifying the Primary Language for an Application

Globalization attributes specify how the Application Express engine determines the primary language of an application.

To edit globalization attributes:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click Edit Attributes.
4. Click **Edit Globalization Attributes**.
5. From **Application Primary Language**, select the language in which the application is being developed.
6. From **Application Language Derived From**, specify how the Application Express engine determines (or derives) the application language. Available options are described in Table 14–1.

Table 14–1 Application Language Derived From Options

Option	Description
No NLS (Application not translated)	Select this option if the application will not be translated.
Use Application Primary Language	Determines the application's primary language based on the Application Primary Language attribute. (See step 5.)

Table 14–1 (Cont.) Application Language Derived From Options

Option	Description
Browser (use browser language preference)	Determines the application’s primary language based on the user's browser language preference.
Application Preference (use FSP_ LANGUAGE_PREFERENCE)	Determines the application’s primary language based a value defined using the `HTMLDB_UTIL.SET_ PREFERENCE` API. Select this option to maintain the selected language preference across multiple log ins. **See Also:** "SET_PREFERENCE Procedure" on page 15-29
Item Preference (use item containing preference)	Determines the application’s primary language based on an application-level item called `FSP_LANGUAGE_ PREFERENCE`. Using this option requires Oracle Application Express to determine the appropriate language preference every time the user logs in.

See Also: "Configuring Standard Application Attributes" on page 4-6, "Configuring Globalization Attributes" on page 4-14, and "About Supported Globalization Codes" on page 14-13

Using Format Masks for Items

The Application Express engine applies globalization settings for each rendered page. This default behavior can impact the display of certain items such as numbers and dates.

For example, suppose your application determines the application language based on the user's browser language preference. If the Application Express engine determines the users's browser language preference is French, it displays dates and numbers in a format that conforms to French standards. You can override this default behavior and explicitly control how items display by applying a format mask. You apply a format mask by making a selection from the Display As list:

- When you create the item
- After you create the item by editing the item attributes

To edit item attributes:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Select a page.

   The Page Definition appears.
4. Under Items, select the item name.

   The Edit Page Item page appears.
5. Under Identification, make a selection from the Display As list.

See Also: "Items" on page 4-25 for information about item attributes.

Translating Applications for Multibyte Languages

If your application needs to run in several languages simultaneously (such as Chinese and Japanese), consider configuring your database with a character set to support all of the languages. The same character set has to be configured in the corresponding database access descriptor (DAD) in mod_plsql. UTF8 and AL32UTF8 are the character sets you can use to support almost all languages around the world.

> **See Also:** "Working in a Global Environment" in *Oracle Database Express Edition 2 Day Developer Guide*

Understanding the Translation Process

To translate an application developed in Application Builder, you must map the primary and target application IDs, seed and export text to a translation file, translate the text, and then apply and publish the translation file.

Topics in this section include:

- Step 1: Navigate to the Translate Application Page
- Step 2: Map Primary and Target Application IDs
- Step 3: Seed and Export Text to a Translation File
- Step 4: Translate the XLIFF File
- Step 5: Upload and Publish a Translated XLIFF Document

Step 1: Navigate to the Translate Application Page

You perform the translation process on the Translate Application page.

To navigate to the Translate Application page:

1. On the Database Home Page, click the **Application Builder** icon.
2. Select an application.
3. Click **Shared Components**.
4. Under Globalization, select **Translation Services**.

   The Translate Application page appears.

Step 2: Map Primary and Target Application IDs

The first step in translating an application is to map the primary and target application IDs. The primary application is the application to be translated. The target application is the resulting translated application.

To map the primary and target application IDs:

1. Navigate to the Translate Application page. See "Step 1: Navigate to the Translate Application Page" on page 14-6.
2. On the Translate Application page, select **Map your primary language application to a translated application ID**.

   The Application Mappings page appears.
3. Click **Create**.
4. On the Translation Application Mapping page:

   - Translation Application - Enter a numeric application ID to identify the target application. The translated application ID must be an integer and cannot end in zero.

- Translation Application Language Code - Select the language into which you are translating.
- Image Directory - Enter the directory from where the images will be obtained.

  This attribute determines the virtual path for translated images. For example, if your primary language application had an image prefix of `'/images/'`, you could define additional virtual directories for other languages, such as `'/images/de/'` for German or `'/images/es/'` for Spanish.

5. Click **Create**.

Step 3: Seed and Export Text to a Translation File

The second step is to seed the translation table and then export the translation text to a translation file.

Topics in this section include:

- Seeding Translatable Text
- Exporting Text to a Translation File

Seeding Translatable Text

To translate an application, you must seed the translation. Seeding the translation copies all translatable text into the Translation Text repository. After you have seeded the application and specified the language in the Translation Text repository, you can then generate and export an XLIFF file for translation.

The seeding process keeps your primary language application synchronized with the Translation Text repository. You should run the seed process any time your primary language application changes.

To seed translatable text:

1. Navigate to the Translate Application page. See "Step 1: Navigate to the Translate Application Page" on page 14-6.
2. On the Translate Application page, select **Seed and export the translation text of your application into a translation file**.
3. From Language Mapping, select the appropriate primary and target application ID map.
4. Click **Seed Translatable Text**.

   The XLIFF Export page appears.

Note: XML Localization Interchange File Format (XLIFF) is an XML-based format for exchanging localization data. For information about the XLIFF or to view the XLIFF specification, see:

`http://www.xliff.org`

Exporting Text to a Translation File

After you have seeded translatable text, a status box displays at the top of the XLIFF Export page indicating the total number of attributes that may require translation, including the number of:

- Existing updated attributes that may require translation

- New attributes that may require translation
- Purged attributes that no longer require translation

You can use this information to determine if you need to export translatable text for an entire application or just a specific page.

The XLIFF Export page is divided into two sections. Use the upper section of the page to export translatable text for an entire application (that is, all pages, lists of values, messages, and so on). Use the lower section to export translatable text for a specific page.

To export translatable text for an entire application:

1. Seed the translatable text. See "Seeding Translatable Text" on page 14-7.
2. Under **Step 2, Export XLIFF**:
   a. From Application, select the appropriate primary and target application ID map.
   b. Specify whether or not to include XLIFF target elements.
   c. Under Export, specify what translation text is included in your XLIFF file.
   d. Click **Export XLIFF for Application**.
3. Follow the on-screen instructions.

To export translatable text for a specific page:

1. Seed the translatable text as described in "Seeding Translatable Text" on page 14-7.
2. Under **Export XLIFF for specific Page**:
   a. From Application, select the appropriate primary and target application ID map.
   b. Specify whether or not to include XLIFF target elements.
   c. Under Export, specify what translation text is included in your XLIFF file.
   d. Click **Export XLIFF for Page**.
3. Follow the on-screen instructions.

About Include XLIFF Target Elements When Oracle Application Express generates an XLIFF document, each document contains multiple translation units. Each translation unit consists of a source element and a target element. The XLIFF document can be generated with both the source and target elements for each translation unit. You have the option of generating a file containing only source elements. The updated translations will be applied from the target elements of the translation units.

About Export Use the options under **Export** to specify what translation text is included in your XLIFF file. Select **All translatable elements** to include all translation text for an application. In contrast, select **Only those elements requiring translation** to include only new elements that have not yet been translated. **Only those elements requiring translation** produces an XLIFF file containing new or modified translation units. Also, if translation units were intentionally not previously translated (that is, the source of the translation element equals the target of the translation element), those translation units will also be included in the file.

Step 4: Translate the XLIFF File

After you export a translatable file to XLIFF format, you can translate it into the appropriate languages. Because XLIFF is an open standard XML file for exchanging translation, most translation vendors should support it. Oracle Application Express only supports XLIFF files encoded in UTF-8 character sets. In other words, it exports XLIFF files for translation in UTF-8 and assumes that the translated XLIFF files will be in the same character set.

Translation is a time-consuming task. Oracle Application Express supports incremental translation so that application development can be done in parallel with the translation. A XLIFF file can be translated and uploaded to Oracle Application Express even when only part of the XLIFF file is translated. For strings that have no translation in the corresponding translated application, Oracle Application Express uses the corresponding ones in the primary language.

See Also: For more information about the XLIFF, or to view the XLIFF specification see:

`http://www.xliff.org`

Step 5: Upload and Publish a Translated XLIFF Document

After your XLIFF document has been translated, the next step is to upload it.

To upload a translated XLIFF document:

1. Navigate to the Translate Application page. See "Step 1: Navigate to the Translate Application Page" on page 14-6.
2. On the Translate Application page, select **Apply your translation file and publish**.
3. Click **Upload XLIFF**.
4. On the XLIFF Upload page:
   a. Specify a title.
   b. Enter a description.
   c. Click **Browse** and locate the file to be uploaded.
   d. Click **Upload XLIFF File**.

   The uploaded document appears in the XLIFF Files repository.

After you upload an XLIFF document, the next step is to apply the XLIFF document and then publish the translated application. When you apply an XLIFF document, the Application Express engine parses the file and then updates the translation tables with the new translatable text.

Publishing your application creates a copy of the base language application, substituting the translated text strings from your translations table. This published application can then be used to render your application in alternate languages.

Remember that in order to run an application in an alternative language, you need to run it with globalization settings that will cause an alternative language version to display. For example, if the language is derived from the browser language, you must set the browser language to the same language as the translated application.

See Also: "Specifying the Primary Language for an Application" on page 14-4

To apply and publish a translated XLIFF document:

1. Navigate to the Translate Application page. See "Step 1: Navigate to the Translate Application Page" on page 14-6.
2. On the Translate Application page, select **Apply your translation file and publish**.
3. In the XLIFF Files repository, click the **View** icon adjacent to the document you want to publish.
4. From Apply to, select the appropriate primary and target application ID map.
5. Click **Apply XLIFF Translation File**.
6. Click **Publish Application**.

To delete an uploaded XLIFF document:

1. Navigate to the Translate Application page. See "Step 1: Navigate to the Translate Application Page" on page 14-6.
2. On the Translate Application page, select **Apply your translation file and publish**.
3. In the XLIFF Files repository, select the check box to the left of the document title.
4. Click **Delete Checked**.

You should verify the existence of the translated application after it is published. Translated applications do not display in the Available Applications list on the Application Builder home page. Instead, use the Application Navigate list on the left side of the page.

Note that in order for a translated application to appear in Application Builder, you need to make sure that you have correctly configured the application Globalization attributes.

See Also: "Specifying the Primary Language for an Application" on page 14-4

Translating Messages Used in PL/SQL Procedures

If your application includes PL/SQL regions or PL/SQL processes or calls PL/SQL package, procedures, or functions, you may need to translate generated HTML. First, you define each message on the Translatable Messages page. Second, you use the `HTMLDB_LANG.MESSAGE` API to translate the messages from PL/SQL stored procedures, functions, triggers, or packaged procedures and functions.

Defining Translatable Messages

You create translatable messages on the Translate Messages page.

To define a new translation message:

1. Navigate to the Translate Application page. See "Step 1: Navigate to the Translate Application Page" on page 14-6.
2. On the Translate Application page, select **Optionally translate messages which are used by PL/SQL procedures and functions**.
3. On the Translate Messages page, click **Create**.
4. On the Identify Text Message page:

a. In Name, enter a name to identify the text message
b. In Language, select the language for which the message would be used
c. In text, enter the text to be returned when the text message is called.

   For example, you could define the message `GREETING_MSG` in English as:

```
Good morning %0
```

   Or, you could define the message `GREETING_MSG` in German as:

```
Guten Tag %0
```

5. Click **Create**.

HTMLDB_LANG.MESSAGE API

Use the `HTMLDB_LANG.MESSAGE` API to translate text strings (or messages) generated from PL/SQL stored procedures, functions, triggers, packaged procedures, and functions.

Syntax

```
HTMLDB_LANG.MESSAGE (
    p_name    IN    VARCHAR2 DEFAULT NULL,
    p0        IN    VARCHAR2 DEFAULT NULL,
    p1        IN    VARCHAR2 DEFAULT NULL,
    p2        IN    VARCHAR2 DEFAULT NULL,
    ...
    p9        IN    VARCHAR2 DEFAULT NULL,
    p_lang    IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 14–2 describes the parameters available in the `HTMLDB_LANG.MESSAGE API`.

Table 14–2 HTMLDB_LANG.MESSAGE Parameters

Parameter	Description
`p_name`	Name of the message as defined in Oracle Application Express.

`p0`
...
`p9`
	Dynamic substitution value: `p0` corresponds to 0% in the message; `p1` corresponds to 1% in the message; p2 corresponds to 2% in the message and so on.

`p_lang`	Language code for the message to be retrieved. If not specified, Oracle Application Express uses the current language for the user as defined in the Application Language Derived From attribute. **See Also:** "Specifying the Primary Language for an Application" on page 14-4

Example

The following example assumes you have defined a message called `GREETING_MSG` in your application in English as `Good morning%0` and in German as `Guten Tag%1`. The following example demonstrates how you could invoke this message from PL/SQL:

```
BEGIN
```

```
    --
    -- Print the greeting
    --
    HTMLDB_LANG.MESSAGE('GREETING_MSG', V('APP_USER'));
END;
```

How the `p_lang` attribute is defined depends on how the Application Express engine derives the Application Primary Language. For example, if you are running the application in German and the previous call is made to the `HTMLDB_LANG.MESSAGE` API, the Application Express engine first looks for a message called `GREETING_MSG` with a `LANG_CODE` of `de`. If it does not find anything, then it will revert to the Application Primary Language attribute. If it still does not find anything, the Application Express engine looks for a message by this name with a language code of `en-us`.

> **See Also:** "Specifying the Primary Language for an Application" on page 14-4 for information about the Application Primary Language attribute

Translating Data That Supports List of Values

You create a dynamic translation to translate dynamic pieces of data. For example, you might use a dynamic translation on a list of values based on a database query.

Dynamic translations differ from messages in that you query a specific string rather than a message name. You define dynamic translations on the Dynamic Translations page. You then use the `HTMLDB_LANG.LANG` API to return the dynamic translation string identified by the `p_primary_text_string` parameter.

Defining a Dynamic Translation

You define dynamic translations on the Dynamic Translations page. A dynamic translation consists of a translate-from language string, a language code, and a translate-to string.

To define a dynamic translation:

1. Navigate to the Translate Application page. See "Step 1: Navigate to the Translate Application Page" on page 14-6.
2. On the Translate Application page, select **Optionally identify any data that needs to be dynamically translated to support SQL based lists of values**.
3. On the Dynamic Translations page, click **Create** and specify the following:
   a. In Language, select a target language.
   b. In Translate From Text, enter the source text to be translated.
   c. In Translate To, enter the translated text.
4. Click **Create**.

HTMLDB_LANG.LANG API

Syntax

```
HTMLDB_LANG.LANG (
    p_primary_text_string    IN    VARCHAR2 DEFAULT NULL,
    p0                       IN    VARCHAR2 DEFAULT NULL,
    p1                       IN    VARCHAR2 DEFAULT NULL,
    p2                        IN    VARCHAR2 DEFAULT NULL,
    ...
    p9                        IN    VARCHAR2 DEFAULT NULL,
    p_primary_language        IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 14–3 describes the parameters available in the HTMLDB_LANG.LANG API.

Table 14–3 HTMLDB_LANG.LANG Parameters

Parameter	Description
p_primary_string	Text string of the primary language. This will be the value of the Translate From Text in the dynamic translation.
p0 ... p9	Dynamic substitution value: p0 corresponds to 0% in the in the translation string; p1 corresponds to 1% in the in the translation string; p2 corresponds to 2% in the translation string and so on.
p_primary_language	Language code for the message to be retrieved. If not specified, Oracle Application Express uses the current language for the user as defined in the Application Language Derived From attribute. **See Also:** "Specifying the Primary Language for an Application" on page 14-4

Example

Suppose you have a table that defines all primary colors. You could define a dynamic message for each color and then apply the LANG function to the defined values in a query, for example:

```
SELECT HTMLDB_LANG.LANG(color)
  FROM my_colors
```

If you were running the application in German and RED was a value for the color column in the my_colors table. If you defined the German word for red, the previous example would return ROT.

About Supported Globalization Codes

If you are building a multilingual application, it is important to understand how globalization codes affect the way in which your application runs. These codes are set automatically based on the application-level Globalization attributes you select.

> **See Also:** "Specifying the Primary Language for an Application" on page 14-4

NLS_LANGUAGE and NLS_TERRITORY determine the default presentation of number, dates, and currency.

Table 14–4 describes the globalization codes in Oracle Application Express.

Table 14–4 Oracle Application Express Globalization Codes

Language Name	Language Code	NLS_LANGUAGE	NLS_TERRITORY
Afrikaans	`af`	ENGLISH	SOUTH AFRICA
Arabic	`ar`	ARABIC	UNITED ARAB EMIRATES
Arabic (Algeria)	`ar-dz`	ARABIC	ALGERIA
Arabic (Bahrain)	`ar-bh`	ARABIC	BAHRAIN
Arabic (Egypt)	`ar-eg`	EGYPTIAN	EGYPT
Arabic (Iraq)	`ar-iq`	ARABIC	IRAQ
Arabic (Jordan)	`ar-jo`	ARABIC	JORDAN
Arabic (Kuwait)	`ar-kw`	ARABIC	KUWAIT
Arabic (Lebanon	`ar-lb`	ARABIC	LEBANNON
Arabic (Libya)	`ar-ly`	ARABIC	LIBYA
Arabic (Morocco)	`ar-ma`	ARABIC	MOROCCO
Arabic (Oman)	`ar-om`	ARABIC	OMAN
Arabic (Qatar)	`ar-qa`	ARABIC	QATAR
Arabic (Saudi Arabia)	`ar-sa`	ARABIC	SAUDI ARABIA
Arabic (Syria)	`ar-sy`	ARABIC	SYRIA
Arabic (Tunisia)	`ar-tn`	ARABIC	TUNISIA
Arabic (U.A.E.)	`ar-ae`	ARABIC	UNITED ARAB EMIRATES
Arabic (YEMEN)	`ar-ye`	ARABIC	YEMEN
Assamese	`as`	ASSAMESE	INDIA
Basque	`eu`	FRENCH	FRANCE
Belarusian	`be`	RUSSIAN	RUSSIA
Bengali	`bn`	BANGLA	BANGLADESH
Bulgarian	`bg`	BULGARIAN	BULGARIA
Catalan	`ca`	CATALAN	CATALONIA
Chinese	`zh`	SIMPLIFIED CHINESE	CHINA
Chinese (China)	`zh-cn`	SIMPLIFIED CHINESE	CHINA
Chinese (Hong Kong SAR)	`zh-hk`	TRADITIONAL CHINESE	HONG KONG
Chinese (Macau SAR)	`zh-mo`	TRADITIONAL CHINESE	HONG KONG
Chinese (Singapore)	`zh-sg`	SIMPLIFIED CHINESE	SINGAPORE
Chinese (Taiwan)	`zh-tw`	TRADITIONAL CHINESE	TAIWAN
Croatian	`hr`	CROATIAN	CROATIA
Czech	`cs`	CZECH	CZECH REPUBLIC
Danish	`da`	DANISH	DENMARK
Dutch (Belgium)	`nl-be`	DUTCH	BELGIUM
Dutch (Netherlands)	`nl`	DUTCH	THE NETHERLANDS
English	`en`	AMERICAN	AMERICA

Table 14–4 (Cont.) Oracle Application Express Globalization Codes

Language Name	Language Code	NLS_LANGUAGE	NLS_TERRITORY
English (Australia)	`en-au`	ENGLISH	AUSTRALIA
English (Belize)	`en-bz`	ENGLISH	UNITED KINGDOM
English (Canada)	`en-ca`	ENGLISH	CANADA
English (Ireland)	`en-ie`	ENGLISH	IRELAND
English (Jamaica)	`en-jm`	ENGLISH	UNITED KINGDOM
English (New Zealand)	`en-nz`	ENGLISH	NEW ZEALAND
English (Philippines)	`en-ph`	ENGLISH	PHILIPPINES
English (South Africa)	`en-za`	ENGLISH	SOUTH AFRICA
English (Trinidad)	`en-tt`	ENGLISH	UNITED KINGDOM
English (United Kingdom)	`en-gb`	ENGLISH	UNITED KINGDOM
English (United States)	`en-us`	AMERICAN	AMERICA
English (Zimbabwe)	`en-zw`	ENGLISH	UNITED KINGDOM
Estonian	`et`	ESTONIAN	ESTONIA
Faeroese	`fo`	ENGLISH	UNITED KINGDOM
Farsi	`fa`	ENGLISH	UNITED KINGDOM
Finnish	`fi`	FINNISH	FINLAND
French (Belgium)	`fr-be`	FRENCH	BELGIUM
French (Canada)	`fr-ca`	CANADIAN FRENCH	CANADA
French (France)	`fr`	FRENCH	FRANCE
French (Luxembourg)	`fr-lu`	FRENCH	LUXEMBOURG
French (Monaco)	`fr-mc`	FRENCH	FRANCE
French (Switzerland)	`fr-ch`	FRANCH	SWITZERLAND
FYRO Macedonian	`mk`	MACEDONIAN	FYR MACEDONIA
Gaelic	`gd`	ENGLISH	UNITED KINGDOM
Galician	`gl`	SPANISH	SPAIN
German (Austria)	`de-at`	GERMAN	AUSTRIA
German (Germany)	`de`	GERMAN	GERMANY
German (Liechtenstein)	`de-li`	GERMAN	GERMANY
German (Luxemgourg)	`de-lu`	GERMAN	LUXEMBOURG
German (Switzerland)	`de-ch`	GERMAN	SWITZERLAND
Greek	`el`	GREEK	GREECE
Gujarati	`gu`	GUJARATI	INDIA
Hebrew	`he`	HEBREW	ISRAEL
Hindi	`hi`	HINDI	INDIA
Hungarian	`hu`	HUNGARIAN	HUNGARY
Icelandic	`is`	ICELANDIC	ICELAND

Table 14–4 (Cont.) Oracle Application Express Globalization Codes

Language Name	Language Code	NLS_LANGUAGE	NLS_TERRITORY
Indonesian	`id`	INDONESIAN	INDONESIA
Italian (Italy)	`it`	ITALIAN	ITALY
Italian (Switzerland)	`it-ch`	ITALIAN	SWITZERLAND
Japanese	`ja`	JAPANESE	JAPAN
Kannada	`kn`	KANNADA	INDIA
Kazakh	`kk`	CYRILLIC KAZAKH	KAZAKHSTAN
Konkani	`kok`	KOREAN	KOREA
Korean	`ko`	KOREAN	KOREA
Kyrgyz	`kz`	RUSSIAN	RUSSIA
Latvian	`lv`	LATVIAN	LATVIA
Lithuanian	`lt`	LITHUANIAN	LITHUANIANA
Malay (Malaysia)	`ms`	MALAY	MALAYSIA
Malayalam	`ml`	MALAYALAM	INDIA
Maltese	`mt`	ENGLISH	UNITED KINGDOM
Marathi	`mr`	ENGLISH	INDIA
Nepali (India)	`ne`	ENGLISH	UNITED KINGDOM
Norwegian (Bokmal)	`nb-no`	NORWEGIAN	NORWAY
Norwegian (Bokmal)	`no`	NORWEGIAN	NORWAY
Norwegian (Nynorsk)	`nn-no`	NORWEGIAN	NORWAY
Oriya	`or`	ORIYA	INDIA
Polish	`pl`	POLISH	POLAND
Portuguese (Brazil)	`pt-br`	BRAZILIAN PORTUGUESE	BRAZIL
Portuguese (Portugal)	`pt`	PORTUGUESE	PORTUGAL
Punjabi	`pa`	PUNJABI	INDIA
Romanian	`ro`	ROMANIAN	ROMANIA
Russian	`ru`	RUSSIAN	RUSSIA
Russian (Moldova)	`ru-md`	RUSSIAN	RUSSIA
Serbia	`sr`	CYRILLIC SERBIAN	SERBIA AND MONTENEGRO
Slovak	`sk`	SLOVAK	SLOVAKIA
Slovenian	`sl`	SLOVENIAN	SLOVENIA
Spanish (Argentina)	`es-ar`	LATIN AMERICAN SPANISH	ARGENTINA
Spanish (Bolivia)	`es-bo`	LATIN AMERICAN SPANISH	ARGENTINA
Spanish (Chile)	`es-cl`	LATIN AMERICAN SPANISH	CHILE
Spanish (Columbia)	`ec-co`	LATIN AMERICAN SPANISH	COLUMBIA
Spanish (Costa Rica)	`es-cr`	LATIN AMERICAN SPANISH	COSTA RICA
Spanish (Dominican Republic)	`es-do`	LATIN AMERICAN SPANISH	PUERTO RICO

Table 14–4 (Cont.) Oracle Application Express Globalization Codes

Language Name	Language Code	NLS_LANGUAGE	NLS_TERRITORY
Spanish (Ecudor)	`es-ec`	LATIN AMERICAN SPANISH	ECUDOR
Spanish (El Salvador)	`es-sv`	LATIN AMERICAN SPANISH	EL SALVADOR
Spanish (Guatemala)	`es-gt`	LATIN AMERICAN SPANISH	GUATEMALA
Spanish (Honduras)	`es-hn`	LATIN AMERICAN SPANISH	GUATEMALA
Spanish (Mexico)	`es-mx`	MEXICAN SPANISH	MEXICO
Spanish (Nicaragua)	`es-ni`	LATIN AMERICAN SPANISH	Nicaragua
Spanish (Panama)	`es-pa`	LATIN AMERICAN SPANISH	Panama
Spanish (Paraguay)	`es-py`	LATIN AMERICAN SPANISH	ARGENTINA
Spanish (Peru)	`es-pe`	LATIN AMERICAN SPANISH	PERU
Spanish (Peurto Rico)	`es-pr`	LATIN AMERICAN SPANISH	PEURTO RICO
Spanish (Traditional Sort)	`es`	LATIN AMERICAN SPANISH	SPAIN
Spanish (United States)	`es-us`	LATIN AMERICAN SPANISH	AMERICAN
Spanish (Uruguay)	`es-uy`	LATIN AMERICAN SPANISH	ARGENTINA
Spanish (Venezuela)	`es-ve`	LATIN AMERICAN SPANISH	VENEZUELA
Swedish	`sv`	SWEDISH	SWEDEN
Swedish	`sv-fi`	SWEDISH	FINLAND
Tamil	`ta`	TAMIL	INDIA
Telugu	`te`	TELUGU	INDIA
Thai	`th`	THAI	THAILAND
Turkish	`tr`	TURKISH	TURKEY
Ukrainian	`uk`	UKRAINIAN	UKRAINE
Urdu	`ur`	ENGLISH	UNITED KINGDOM
Uzbek	`uz`	LATIN UZBEK	UZBEKISTAN
Vietnamese	`vi`	VIETNAMESE	VIETNAM
Zulu	`zu`	ENGLISH	UNITED KINGDOM

15 Oracle Application Express APIs

This section describes the APIs available in Oracle Application Express.

This section contains the following topics:

HTMLDB_UTIL

The `HTMLDB_UTIL` package provides utilities you can use when programming in the Oracle Application Express environment. You can use the `HTMLDB_UTIL` package to get and set session state, get files, check authorizations for users, reset different states for users, and also to get and set preferences for users.

Topics in this section include:

- STRING_TO_TABLE Function
- TABLE_TO_STRING Function
- URL_ENCODE Function

CHANGE_CURRENT_USER_PW Procedure

This procedure changes the password of the currently authenticated user, assuming Application Express user accounts are in use.

Syntax

```
HTMLDB_UTIL.CHANGE_CURRENT_USER_PW(
    p_new_password IN VARCHAR2);
```

Parameters

Table 15–1 describes the parameters available in the `CHANGE_CURRENT_USER_PW` procedure.

Table 15–1 CHANGE_CURRENT_USER_PW Parameters

Parameter	Description
`p_new_password`	The new password value in clear text

Example

```
BEGIN
HTMLDB_UTIL.CHANGE_CURRENT_USER_PW ('secret99');
END;
```

CLEAR_APP_CACHE Procedure

This procedure removes session state for a given application for the current session.

Syntax

```
HTMLDB_UTIL.CLEAR_APP_CACHE (
    p_app_id    IN    VARCHAR2 DEFAULT NULL);
```

Parameters

Table 15–2 describes the parameters available in the `CLEAR_APP_CACHE` procedure.

Table 15–2 CLEAR_APP_CACHE Parameters

Parameter	Description
`p_app_id`	The ID of the application for which session state will be cleared for current session

Example

```
BEGIN
        HTMLDB_UTIL.CLEAR_APP_CACHE('100');
END;
```

CLEAR_PAGE_CACHE Procedure

This procedure removes session state for a given page for the current session.

Syntax

```
HTMLDB_UTIL.CLEAR_PAGE_CACHE (
    p_page_id IN NUMBER DEFAULT NULL);
```

Parameters

Table 15–3 describes the parameters available in the `CLEAR_APP_CACHE` procedure.

Table 15–3 CLEAR_PAGE_CACHE Parameters

Parameter	Description
`p_page_id`	The ID of the page in the current application for which session state will be cleared for current session

Example

```
BEGIN
HTMLDB_UTIL.CLEAR_PAGE_CACHE('10');
END;
```

CLEAR_USER_CACHE Procedure

This procedure removes session state and application system preferences for the current user's session. Run this procedure if you reuse session IDs and want to run applications without the benefit of existing session state.

Syntax

```
HTMLDB_UTIL.CLEAR_USER_CACHE;
```

Parameters

None.

Example

```
BEGIN
       HTMLDB_UTIL.CLEAR_USER_CACHE;
END;
```

COUNT_CLICK Procedure

This procedure counts clicks from an application built in Application Builder to an external site. You can also use the shorthand version, procedure `Z`, in place of `HTMLDB_UTIL.COUNT_CLICK`.

Syntax

```
HTMLDB_UTIL.COUNT_CLICK (
    p_url         IN    VARCHAR2,
    p_cat         IN    VARCHAR2,
    p_id          IN    VARCHAR2    DEFAULT NULL,
    p_user        IN    VARCHAR2    DEFAULT NULL,
    p_workspace   IN    VARCHAR2    DEFAULT NULL);
```

Parameters

Table 15–4 describes the parameters available in the COUNT_CLICK procedure.

Table 15–4 COUNT_CLICK Parameters

Parameter	Description
p_url	The URL to which to redirect
p_cat	A category to classify the click
p_id	Secondary ID to associate with the click (optional)
p_user	The application user ID (optional)
p_workspace	The workspace associated with the application (optional)

Example

```
BEGIN
htp.p('<a
href=HTMLDB_UTIL.COUNT_CLICK?p_url=http://yahoo.com&p_cat=yahoo&p_workspace=NNN>
Click</a>');
end;
```

Where NNN equals your workspace ID.

See Also: "Purging the External Clicks Log" on page 8-9

CREATE_USER Procedure

This procedure creates a new account record in the Application Express user account table. To execute this procedure, the current user must have administrative privileges.

Syntax

```
HTMLDB_UTIL.CREATE_USER(
    p_user_id                   NUMBER                  IN      DEFAULT NULL
    p_user_name                 VARCHAR2                IN
    p_first_name                VARCHAR2                IN      DEFAULT NULL
    p_last_name                 VARCHAR2                IN      DEFAULT NULL
    p_description               VARCHAR2                IN      DEFAULT NULL
    p_email_address             VARCHAR2                IN      DEFAULT NULL
    p_web_password              VARCHAR2                IN
    p_web_password_format       VARCHAR2                IN      DEFAULT NULL
    p_group_ids                 VARCHAR2                IN      DEFAULT NULL
    p_attribute_01              VARCHAR2                IN      DEFAULT NULL
    p_attribute_02              VARCHAR2                IN      DEFAULT NULL
    p_attribute_03              VARCHAR2                IN      DEFAULT NULL
    p_attribute_04              VARCHAR2                IN      DEFAULT NULL
    p_attribute_05              VARCHAR2                IN      DEFAULT NULL
    p_attribute_06              VARCHAR2                IN      DEFAULT NULL
    p_attribute_07              VARCHAR2                IN      DEFAULT NULL
    p_attribute_08              VARCHAR2                IN      DEFAULT NULL
    p_attribute_09              VARCHAR2                IN      DEFAULT NULL
    p_attribute_10              VARCHAR2                IN      DEFAULT NULL)
```

Parameters

Table 15–5 describes the parameters available in the CREATE_USER procedure.

Table 15–5 CREATE_USER Procedure Parameters

Parameter	Description
p_user_id	Numeric primary key of user account
p_user_name	Alphanumeric name used for login
p_first_name	Informational
p_last_name	Informational
p_description	Informational
p_email_address	Email address
p_web_address	Clear text password
p_group_ID	Colon separated list of numeric group IDs
p_attribute_01	Arbitrary text accessible with an API
...	
p_attribute_10	

Example

```
BEGIN
HTMLDB_UTIL.CREATE_USER
    P_USER_NAME    => 'NEWUSER1',
    P_WEB_PASSWORD => 'secret99');
END;
```

CREATE_USER_GROUP Procedure

This procedure changes the password of the currently authenticated user, assuming Application Express user accounts are in use. To execute this procedure, the current user must have administrative privilege in the workspace.

Syntax

```
HTMLDB_UTIL.CREATE_USER_GROUP(
    p_id                       NUMBER                   IN
    p_group_name               VARCHAR2                 IN
    p_security_group_id        NUMBER                   IN
    p_group_desc               VARCHAR2                 IN);
```

Parameter

Table 15–6 describes the parameters available in the CREATE_USER_GROUP procedure.

Table 15–6 CREATE_USER_GROUP Parameters

Parameter	Description
p_id	Primary key of group
p_group_name	Arbitrary name
p_security_group_id	Workspace ID
p_group_desc	Descriptive text

Example

```
BEGIN
HTMLDB_UTIL.CREATE_USER_GROUP (
    p_id                => 0 - trigger will assign PK,
    p_group_name        => 'Managers',
    p_security_group_id => null, -- defaults to current workspace ID
    p_group_desc        => 'text');
END;
```

CURRENT_USER_IN_GROUP Function

This function returns a Boolean result based on whether or not the current user is a member of the specified group. You can use the group name or group ID to identify the group.

Syntax

```
HTMLDB_UTIL.CURRENT_USER_IN_GROUP(
    p_group_name    IN VARCHAR2)
RETURN BOOLEAN;

HTMLDB_UTIL.CURRENT_USER_IN_GROUP(
    p_group_id    IN NUMBER)
RETURN BOOLEAN;
```

Parameters

Table 15–7 describes the parameters available in the `CURRENT_USER_IN_GROUP` function.

Table 15–7 CURRENT_USER_IN_GROUP Parameters

Parameter	Description
`p_group_name`	Identifies the name of an existing group in the workspace
`p_group_id`	Identifies the numeric ID of an existing group in the workspace

Example

```
DECLARE VAL BOOLEAN;
BEGIN
  VAL := HTMLDB_UTIL.CURRENT_USER_IN_GROUP(p_group_name=>'Managers');
END;
```

EDIT_USER Procedure

This procedure enables a user account record to be altered. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
EDIT_USER (
    p_user_id                   NUMBER                  IN
    p_user_name                 VARCHAR2                IN
    p_first_name                VARCHAR2                IN      DEFAULT
    p_last_name                 VARCHAR2                IN      DEFAULT
    p_web_password              VARCHAR2                IN      DEFAULT
    p_new_password              VARCHAR2                IN      DEFAULT
    p_email_address             VARCHAR2                IN      DEFAULT
    p_start_date                VARCHAR2                IN      DEFAULT
    p_end_date                  VARCHAR2                IN      DEFAULT
    p_employee_id               VARCHAR2                IN      DEFAULT
    p_allow_access_to_schemas   VARCHAR2                IN      DEFAULT
    p_person_type               VARCHAR2                IN      DEFAULT
    p_default_schema            VARCHAR2                IN      DEFAULT
    p_group_idS                 VARCHAR2                IN      DEFAULT
    P_DEVELOPER_ROLES           VARCHAR2                IN      DEFAULT
    P_DESCRIPTION               VARCHAR2                IN      DEFAULTIN);
```

Parameters

Table 15–8 describes the parameters available in the `EDIT_USER` procedure.

Table 15–8 EDIT_USER Parameters

Parameter	Description
`p_user_id`	Numeric primary key of the user account
`p_user_name`	Alphanumeric name used for login
`p_first_name`	Informational
`p_last_name`	Informational
`p_web_password`	Clear text password
`p_start_date`	Unused
`p_end_date`	Unused
`p_employee_id`	Unused
`p_allow_access_to_schemas`	A list of schemas assigned to the user's workspace to which the user is restricted
`p_person_type`	Unused
`p_default_schema`	A database schema assigned to the user's workspace, used by default for browsing
`p_group_ids`	Colon-separated list of numeric group IDs
`p_developer_privs`	Colon-separated list of developer privileges (only ADMIN: has meaning to Application Express)
`p_description`	Informational

EXPORT_USERS Procedure

When called from an page, this procedure produces an export file of the current workspace definition, workspace users, and workspace groups. To execute this procedure, the current user must have administrative privilege in the workspace.

Syntax

```
HTMLDB_UTIL.EXPORT_USERS(
    p_export_format in VARCHAR2 DEFAULT 'UNIX')
```

Parameters

Table 15–9 describes the parameters available in the `EXPORT_USERS` procedure.

Table 15–9 EXPORT_USERS Parameters

Parameter	Description
`p_export_format`	Indicates how rows in the export file will be formatted. Specify `'UNIX'` to have the resulting file contain rows delimited by line feeds. Specify `'DOS'` to have the resulting file contain rows delimited by carriage returns and line feeds

Example

```
BEGIN
  HTMLDB_UTIL.EXPORT_USERS;
END;
```

FETCH_APP_ITEM Function

This function fetches session state for the current or specified application in the current or specified session.

Syntax

```
HTMLDB_UTIL.FETCH_APP_ITEM(
    p_item    IN VARCHAR2,
    p_app     IN NUMBER DEFAULT NULL,
    p_session IN NUMBER DEFAULT NULL)
RETURN VARCHAR2;
```

Parameters

Table 15–10 describes the parameters available in the FETCH_APP_ITEM function.

Table 15–10 FETCH_APP_ITEM Parameters

Parameter	Description
`p_item`	The name of an application-level item (not a page item) whose current value is to be fetched
`p_app`	The ID of the application that owns the item (leave null for the current application)
`p_session`	The session ID from which to obtain the value (leave null for the current session)

Example

```
DECLARE VAL VARCHAR2(30);
BEGIN
VAL := HTMLDB_UTIL.FETCH_APP_ITEM (p_item=>'F300_NAME',p_app=>300);
END;
```

FETCH_USER Procedure

This procedure fetches a user account record. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
FETCH_USER (
    p_user_id                      NUMBER                  IN
    p_workspace                    VARCHAR2                OUT
    p_user_name                 VARCHAR2              OUT
    p_first_name                VARCHAR2              OUT
    p_last_name                 VARCHAR2              OUT
    p_web_password              VARCHAR2              OUT
    p_email_address             VARCHAR2              OUT
    p_start_date                VARCHAR2              OUT
    p_end_date                  VARCHAR2              OUT
    p_employee_id               VARCHAR2              OUT
    p_allow_access_to_schemas   VARCHAR2              OUT
    p_person_type               VARCHAR2              OUT
    p_default_schema            VARCHAR2              OUT
    p_groups                    VARCHAR2              OUT
    p_developer_role            VARCHAR2              OUT);
```

Parameters

Table 15–11 describes the parameters available in the `FETCH_USER` procedure.

Table 15–11 Fetch_User Parameters

Parameter	Description
`p_user_id`	Numeric primary key of the user account
`p_workspace`	The name of the workspace
`p_user_name`	Alphanumeric name used for login
`p_first_name`	Informational
`p_last_name`	Informational
`p_description`	Informational
`p_email_address`	Email address
`p_start_date`	Unused
`p_end_date`	Unused
`p_employee_id`	Unused
`p_allow_access_to_schemas`	A list of schemas assigned to the user's workspace to which user is restricted
`p_person_type`	Unused
`p_default_schema`	A database schema assigned to the user's workspace, used by default for browsing
`p_groups`	Unused
`p_developer_role`	Unused

FIND_SECURITY_GROUP_ID Function

This function returns the numeric security group ID of the named workspace.

Syntax

```
HTMLDB_UTIL.FIND_SECURITY_GROUP_ID(
    p_workspace    IN VARCHAR2
RETURN NUMBER;
```

Parameters

Table 15–12 describes the parameters available in the `FIND_SECURITY_GROUP_ID` function.

Table 15–12 FIND_SECURITY_GROUP_ID Parameters

Parameter	Description
`p_workspace`	The name of the workspace

Example

```
DECLARE VAL NUMBER;
BEGIN
  VAL := HTMLDB_UTIL.FIND_SECURITY_GROUP_ID (p_workspace=>'DEMOS');
END;
```

FIND_WORKSPACE Function

This function returns the workspace name associated with a security group ID.

Syntax

```
HTMLDB_UTIL.FIND_WORKSPACE(
    p_security_group_id    IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 15–13 describes the parameters available in the `FIND_WORKSPACE` function.

Table 15–13 FIND_WORKSPACE Parameters

Parameter	Description
`p_security_group_id`	The security group ID of a workspace

Example

```
DECLARE VAL NUMBER;
BEGIN
  VAL := HTMLDB_UTIL.FIND_ FIND_WORKSPACE (p_security_group_id =>'20');
END;
```

GET_ATTRIBUTE Function

This function returns the value of one of the attribute values (1 through 10) of a named user in the Application Express accounts table.

Syntax

```
HTMLDB_UTIL.GET_ATTRIBUTE(
    p_username               IN VARCHAR2
    p_attribute_number       IN NUMBER)
RETURN VARCHAR2;
```

Parameters

Table 15–14 describes the parameters available in the GET_ATTRIBUTE function.

Table 15–14 GET_ATTRIBUTE Parameters

Parameter	Description
`p_username`	User name in the account.
`p_attribute_number`	Number of attributes in the user record (1 through 10)

Example

```
DECLARE VAL VARCHAR2(30);
BEGIN
  VAL := HTMLDB_UTIL.GET_ATTRIBUTE (
                        p_username => 'SCOTT',
                        p_attribute_number => 1);
END;
```

GET_CURRENT_USER_ID Function

This function returns the numeric user ID of the current user.

Syntax

```
HTMLDB_UTIL.GET_CURRENT_USER_ID;
RETURN NUMBER;
```

Parameters

None.

Example

```
DECLARE VAL NUMBER;
BEGIN
  VAL := HTMLDB_UTIL.GET_CURRENT_USER_ID;
END;
```

GET_DEFAULT_SCHEMA Function

This function returns the default schema name associated with the current user.

Syntax

```
HTMLDB_UTIL.GET_DEFAULT_SCHEMA;
RETURN VARCHAR2;
```

Parameters

None.

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := HTMLDB_UTIL. GET_DEFAULT_SCHEMA;
END;
```

GET_EMAIL Function

This function returns the email address associated with the named user.

Syntax

```
HTMLDB_UTIL.GET_EMAIL(
   p_username IN VARCHAR2);
RETURN VARCHAR2;
```

Parameters

Table 15–15 describes the parameters available in `GET_EMAIL` function.

Table 15–15 GET_EMAIL Parameters

Parameter	Description
`p_username`	The user name in the account

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := HTMLDB_UTIL.GET_EMAIL(p_username => 'SCOTT');
END;
```

GET_FILE Procedure

This procedure downloads files from the Oracle Application Express file repository.

Syntax

```
HTMLDB_UTIL.GET_FILE (
    p_file_id    IN   VARCHAR2,
    p_mime_type  IN   VARCHAR2 DEFAULT NULL,
    p_inline     IN   VARCHAR2 DEFAULT 'NO');
```

Parameters

Table 15–16 describes the parameters available in `GET_FILE` procedure.

Table 15–16 GET_FILE Parameters

Parameter	Description

p_file_id	ID in HTMLDB_APPLICATION_FILES of the file to be downloaded. HTMLDB_APPLICATION_FILES is a view on all files uploaded to your workspace. The following example demonstrates how to use HTMLDB_ APPLICATION_FILES:
	`DECLARE`
	`    l_file_id NUMBER;`
	`BEGIN`
	`        SELECT id INTO l_file_id FROM HTMLDB_APPLICATION_FILES`
	`WHERE filename = 'myxml';`
	`        --`
	`        HTMLDB_UTIL.GET_FILE(`
	`              p_file_id   => l_file_id,`
	`              p_mime_type => 'text/xml',`
	`              p_inline    => 'YES');`
	`END;`

p_mime_type	Mime type of the file to download
p_inline	Valid values include YES and NO. YES to display inline in a browser. NO to download as attachment

Example

```
BEGIN
        HTMLDB_UTIL.GET_FILE(
              p_file_id   => '8675309',
              p_mime_type => 'text/xml',
              p_inline    => 'YES');
END;
```

GET_FILE_ID Function

This function obtains the primary key of a file in the Oracle Application Express file repository.

Syntax

```
HTMLDB_UTIL.GET_FILE_ID (
    p_fname    IN   VARCHAR2)
RETURN NUMBER;
```

Parameters

Table 15–17 describes the parameters available in GET_FILE_ID function.

Table 15–17 GET_FILE_ID Parameters

Parameter	Description
p_fname	The NAME in HTMLDB_APPLICATION_FILES of the file to be downloaded. HTMLDB_APPLICATION_FILES is a view on all files uploaded to your workspace.

Example

```
DECLARE
```

```
        l_name VARCHAR2(255);
        l_file_id NUMBER;
BEGIN
        SELECT name INTO l_name FROM HTMLDB_APPLICATION_FILES
        WHERE filename = 'F125.sql';
--
        l_file_id := HTMLDB_UTIL.GET_FILE_ID(p_fname => );
END;
```

GET_FIRST_NAME Function

This function returns the `FIRST_NAME` field stored in the named user account record.

Syntax

```
HTMLDB_UTIL.GET_FIRST_NAME
   p_username IN VARCHAR2);
RETURN VARCHAR2;
```

Parameters

Table 15–18 describes the parameters available in `GET_FIRST_NAME` function.

Table 15–18 GET_FIRST_NAME Parameters

Parameter	Description
`p_username`	Identifies the user name in the account

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := HTMLDB_UTIL.GET_FIRST_NAME(p_username => 'SCOTT');
END;
```

GET_GROUPS_USER_BELONGS_TO Function

This function returns a colon separated list of group names to which the named user is a member.

Syntax

```
HTMLDB_UTIL.GET_GROUPS_USER_BELONGS_TO(
   p_username IN VARCHAR2);
RETURN VARCHAR2;
```

Parameters

Table 15–19 describes the parameters available in `GET_GROUPS_USER_BELONGS_TO` function.

Table 15–19 GET_GROUPS_USER_BELONGS_TO Parameters

Parameter	Description
`p_username`	Identifies the user name in the account

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := HTMLDB_UTIL.GET_GROUPS_USER_BELONGS_TO(p_username => 'SCOTT');
END;
```

GET_GROUP_ID Function

This function returns the numeric ID of a named group in the workspace.

Syntax

```
HTMLDB_UTIL.GET_GROUP_ID(
   p_group_name);
RETURN VARCHAR2;
```

Parameters

Table 15–20 describes the parameters available in `GET_GROUP_ID` function.

Table 15–20 GET_GROUP_ID Parameters

Parameter	Description
`p_group_name`	Identifies the user name in the account

Example

```
DECLARE VAL NUMBER;
BEGIN
  VAL := HTMLDB_UTIL.GET_GROUP_ID(p_group_name => 'Managers');
END;
```

GET_GROUP_NAME Function

This function returns the name of a group identified by a numeric ID.

Syntax

```
HTMLDB_UTIL.GET_GROUP_NAME(
   p_group_id);
RETURN NUMBER;
```

Parameters

Table 15–21 describes the parameters available in `GET_GROUP_NAME` function.

Table 15–21 GET_GROUP_NAME Parameters

Parameter	Description
`p_group_id`	Identifies a numeric ID of a group in the workspace

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := HTMLDB_UTIL.GET_GROUP_NAME(p_group_id => 8922003);
END;
```

GET_LAST_NAME Function

This function returns the `LAST_NAME` field stored in the named user account record.

Syntax

```
HTMLDB_UTIL.GET_LAST_NAME(
   p_username IN VARCHAR2);
RETURN VARCHAR2;
```

Parameters

Table 15–22 describes the parameters available in `GET_LAST_NAME` function.

Table 15–22 GET_LAST_NAME Parameters

Parameter	Description
`p_username`	The user name in the user account record

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := HTMLDB_UTIL.GET_LAST_NAME(p_username => 'SCOTT');
END;
```

GET_USERNAME Function

This function returns the user name of a user account identified by a numeric ID.

Syntax

```
HTMLDB_UTIL.GET_USERNAME(
   p_userid);
RETURN VARCHAR2;
```

Parameters

Table 15–23 describes the parameters available in `GET_USERNAME` function.

Table 15–23 GET_USERNAME Parameters

Parameter	Description
`p_userid`	Identifies the numeric ID of a user account in the workspace

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := HTMLDB_UTIL.GET_USERNAME(p_userid => 228922003);
END;
```

GET_NUMERIC_SESSION_STATE Function

This function returns a numeric value for a numeric item. You can use this function in Oracle Application Express applications wherever you can use PL/SQL or SQL. You can also use the shorthand, function `NV`, in place of `HTMLDB_UTIL.GET_NUMERIC_SESSION_STATE`.

Syntax

```
HTMLDB_UTIL.GET_NUMERIC_SESSION_STATE (
    p_item     IN VARCHAR2)
    RETURN NUMBER;
```

Parameters

Table 15–24 describes the parameters available in `GET_NUMERIC_SESSION_STATE` function.

Table 15–24 GET_NUMERIC_SESSION_STATE Parameters

Parameter	Description
`p_item`	The case insensitive name of the item for which you want to have the session state fetched

Example

```
DECLARE
      l_item_value    Number;
BEGIN
      l_item_value := HTMLDB_UTIL.GET_NUMERIC_SESSION_STATE('my_item');
END;
```

GET_PREFERENCE Function

This function retrieves the value of a previously saved preference for a given user.

Syntax

```
HTMLDB_UTIL.GET_PREFERENCE (
    p_preference  IN    VARCHAR2 DEFAULT NULL,
    p_user        IN    VARCHAR2 DEFAULT V('USER'))
    RETURN VARCHAR2;
```

Parameters

Table 15–25 describes the parameters available in the `GET_PREFERENCE` function.

Table 15–25 GET_PREFERENCE Parameters

Parameter	Description
`p_preference`	Name of the preference to retrieve the value
`p_value`	Value of the preference
`p_user`	User for whom the preference is being retrieved

Example

```
DECLARE
      l_default_view    VARCHAR2(255);
BEGIN
      l_default_view := HTMLDB_UTIL.GET_PREFERENCE(
                   p_preference => 'default_view',
                   p_user       => :APP_USER);
END;
```

GET_SESSION_STATE Function

This function returns the value for an item. You can use this function in your Oracle Application Express applications wherever you can use PL/SQL or SQL. You can also use the shorthand, function `V`, in place of `HTMLDB_UTIL.GET_SESSION_STATE`.

Syntax

```
HTMLDB_UTIL.GET_SESSION_STATE (
    p_item    IN   VARCHAR2)
    RETURN VARCHAR2;
```

Parameters

Table 15–26 describes the parameters available in `GET_SESSION_STATE` function.

Table 15–26 GET_SESSION_STATE Parameters

Parameter	Description
`p_item`	The case insensitive name of the item for which you want to have the session state fetched

Example

```
DECLARE
      l_item_value  VARCHAR2(255);
BEGIN
      l_item_value := HTMLDB_UTIL.GET_SESSION_STATE('my_item');
END;
```

GET_USER_ID Function

This function returns the numeric ID of a named user in the workspace.

Syntax

```
HTMLDB_UTIL.GET_USER_ID(
   p_username);
RETURN VARCHAR2;
```

Parameters

Table 15–27 describes the parameters available in `GET_USER_ID` function.

Table 15–27 GET_USER_ID Parameters

Parameter	Description
`p_username`	Identifies the name of a user in the workspace

Example

```
DECLARE VAL NUMBER;
BEGIN
  VAL := HTMLDB_UTIL.GET_USER_ID(p_username => 'Managers');
END;
```

GET_USER_ROLES Function

This function returns the DEVELOPER_ROLE field stored in the named user account record.

Syntax

```
HTMLDB_UTIL.GET_USER_ROLES(
   p_username IN VARCHAR2);
RETURN VARCHAR2;
```

Parameters

Table 15–28 describes the parameters available in GET_USER_ROLES function.

Table 15–28 GET_USER_ROLES Parameters

Parameter	Description
p_username	Identifies a user name in the account

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := HTMLDB_UTIL.GET_USER_ROLES(p_username=>'SCOTT');
END;
```

IS_LOGIN_PASSWORD_VALID Function

This function returns a Boolean result based on the validity of the password for a named user account in the current workspace. This function returns true if the password matches and it returns false if the password does not match.

Syntax

```
HTMLDB_UTIL.IS_LOGIN_PASSWORD_VALID(
   p_username IN VARCHAR2,
   p_password IN VARCHAR2);
RETURN BOOLEAN;
```

Parameters

Table 15–29 describes the parameters available in the IS_LOGIN_PASSWORD_VALID function.

Table 15–29 IS_LOGIN_PASSWORD_VALID Parameters

Parameter	Description
p_username	User name in account
p_password	Password to be compared with password stored in the account

Example

```
DECLARE VAL BOOLEAN;
BEGIN
  VAL := HTMLDB_UTIL. IS_LOGIN_PASSWORD_VALID (
             p_username=>'SCOTT'
             p_password=>'tiger');
END;
```

IS_USERNAME_UNIQUE Function

This function returns a Boolean result based on whether the named user account is unique in the workspace.

Syntax

```
HTMLDB_UTIL.IS_USERNAME_UNIQUE(
   P_username IN VARCHAR2);
RETURN BOOLEAN;
```

Parameters

Table 15–30 describes the parameters available in `IS_USERNAME_UNIQUE` function.

Table 15–30 IS_USERNAME_UNIQUE Parameters

Parameter	Description
`p_username`	Identifies the user name to be tested

Example

```
DECLARE VAL BOOLEAN;
BEGIN
  VAL := HTMLDB_UTIL.IS_USERNAME_UNIQUE(
             p_username=>'SCOTT');
END;
```

KEYVAL_NUM Function

This function gets the value of the package variable (`wwv_flow_utilities.g_val_num`) set by `HTMLDB_UTIL.SAVEKEY_NUM`.

Syntax

```
HTMLDB_UTIL.KEYVAL_NUM;
```

Parameters

Table 15–31 describes the parameters available in `KEYVAL_NUM` function.

Table 15–31 KEYVAL_NUM Parameters

Parameter	Description
`p_val`	The numeric value previously saved

Example

```
DECLARE
BEGIN
    VAL := HTMLDB_UTIL.KEYVAL_NUM;
END;
```

See Also: "SAVEKEY_NUM Function" on page 15-26

KEYVAL_VC2 Function

This function gets the value of the package variable (`wwv_flow_utilities.g_val_vc2`) set by `HTMLDB_UTIL.SAVEKEY_VC2`.

Syntax

```
HTMLDB_UTIL.KEYVAL_VC2;
```

Parameters

`p_val` is the VARCHAR2 value previously saved.

Example

```
DECLARE
VAL VARCHAR2(4000);
BEGIN
    VAL := HTMLDB_UTIL.KEYVAL_VC2;

END;
```

See Also: "SAVEKEY_VC2 Function" on page 15-26

PREPARE_URL Function

Given a ready-to-render `f?p` relative URL, this function adds a Session State Protection checksum argument (`&cs=`) if one is required.

Syntax

```
HTMLDB_UTIL.PREPARE_URL (
   p_url IN VARCHAR2
   p_url_charset IN VARCHAR2 default null,
   p_checksum_type IN VARCHAR2 default null)
RETURN VARCHAR2;
```

Parameters

Table 15–32 describes the parameters available in the PREPARE_URL function.

Table 15–32 PREPARE_URL Parameters

Parameter	Description
p_url	An f?p relative URL with all substitutions resolved
p_url_charset	The character set name (for example, `UTF-8`) to use when escaping special characters contained within argument values
p_checksum type	Null or any of the following six values, `SESSION` or `3`, `PRIVATE_BOOKMARK` or `2`, or `PUBLIC_BOOKMARK` or `1`

Example

```
DECLARE
l_url varchar2(2000);
l_session number := v('APP_SESSION');
BEGIN
l_url :=
HTMLDB_UTIL.PREPARE_URL('f?p=100:1:'||l_session||'::NO::P1_ITEM:xyz');
END;
```

PUBLIC_CHECK_AUTHORIZATION Function

Given the name of a security scheme, this function determines if the current user passes the security check.

Syntax

```
HTMLDB_UTIL.PUBLIC_CHECK_AUTHORIZATION (
    p_security_scheme    IN    VARCHAR2)
    RETURN BOOLEAN;
```

Parameters

Table 15–33 describes the parameters available in `PUBLIC_CHECK_AUTHORIZATION` function.

Table 15–33 PUBLIC_CHECK_AUTHORIZATION Parameters

Parameter	Description
`p_security_name`	The name of the security scheme that determines if the user passes the security check

Example

```
DECLARE
      l_check_security  BOOLEAN;
BEGIN
      l_check_security := HTMLDB_UTIL.PUBLIC_CHECK_AUTHORIZATION('my_auth_
scheme');
END;
```

REMOVE_PREFERENCE Procedure

This function removes the preference for the supplied user.

Syntax

```
HTMLDB_UTIL.REMOVE_PREFERENCE(
    p_preference    IN    VARCHAR2 DEFAULT NULL,
    p_user          IN    VARCHAR2 DEFAULT V('USER'));
```

Parameters

Table 15–34 describes the parameters available in the `REMOVE_PREFERENCE` procedure.

Table 15–34 REMOVE_PREFERENCE Parameters

Parameter	Description
`p_preference`	Name of the preference to remove
`p_user`	User for whom the preference is defined

Example

```
BEGIN
       HTMLDB_UTIL.REMOVE_PREFERENCE(
                    p_preference => 'default_view',
                    p_user       => :APP_USER);
```

```
END;
```

REMOVE_SORT_PREFERENCES Procedure

This procedure removes the user's column heading sorting preference value.

Syntax

```
HTMLDB_UTIL.REMOVE_SORT_PREFERENCES (
    p_user  IN VARCHAR2 DEFAULT V('USER'));
```

Parameters

Table 15–35 describes the parameters available in `REMOVE_SORT_PREFERENCES` function.

Table 15–35 REMOVE_SORT_PREFERENCES Parameters

Parameter	Description
`p_user`	Identifies the user for whom sorting preferences will be removed

Example

```
BEGIN
      HTMLDB_UTIL.REMOVE_SORT_PREFERENCES(:APP_USER);
END;
```

REMOVE_USER Procedure

This procedure removes the user account identified by the primary key or a user name. To execute this procedure, the current user must have administrative privilege in the workspace.

Syntax

```
HTMLDB_UTIL.REMOVE_USER(
    p_user_id IN NUMBER,
    p_user_name IN VARCHAR2);
```

Parameters

Table 15–36 describes the parameters available in the `REMOVE_USER` procedure.

Table 15–36 REMOVE_USER Parameters

Parameter	Description
`p_user_id`	The numeric primary key of the user account record
`p_user_name`	The the user name of the user account

Example

```
BEGIN
HTMLDB_UTIL.REMOVE_USER(p_user_id=>'99997');
END;

BEGIN
```

```
HTMLDB_UTIL.REMOVE_USER(p_user_name => 'SCOTT');
END;
```

RESET_PW Procedure

This procedure resets the password for a named user and emails it in a message to the email address located for the named account in the current workspace. To execute this procedure, the current user must have administrative privilege in the workspace.

Syntax

```
HTMLDB_UTIL.RESET_PW(
    p_user IN VARCHAR2,
    p_msg  IN VARCHAR2);
```

Parameters

Table 15–37 describes the parameters available in the `RESET_PW` procedure.

Table 15–37 RESET_PW Parameters

Parameter	Description
`p_user`	The user name of the user account
`p_msg`	Message text to be mailed to a user

Example

```
BEGIN
HTMLDB_UTIL.RESET_PW(
    p_user => 'SCOTT',
    p_msg => 'Contact help desk at 555-1212 with questions');
END;
```

RESET_AUTHORIZATIONS Procedure

To increase performance, Oracle Application Express caches the results of authorization schemes after they have been evaluated. You can use this procedure to undo caching, requiring each authorization scheme be revalidated when it is next encountered during page show or accept processing. You can use this procedure if you want users to have the ability to change their responsibilities (their authorization profile) within your application.

Syntax

```
HTMLDB_UTIL.RESET_AUTHORIZATIONS;
```

Parameters

None.

Example

```
BEGIN
HTMLDB_UTIL.RESET_AUTHORIZATIONS;
END;
```

SAVEKEY_NUM Function

This function sets a package variable (`wwv_flow_utilities.g_val_num`) so that it can be retrieved using the function `KEYVAL_NUM`.

Syntax

```
HTMLDB_UTIL.SAVEKEY_NUM(
    p_val IN NUMBER);
```

Parameters

Table 15–38 describes the parameters available in the `SAVEKEY_NUM` procedure.

Table 15–38 SAVEKEY_NUM Parameters

Parameter	Description
`p_val`	The numeric value to be saved

Example

```
DECLARE
VAL NUMBER;
BEGIN
    VAL := HTMLDB_UTIL.SAVEKEY_NUM(
        p_val => 10);
END;
```

See Also: "KEYVAL_NUM Function" on page 15-21

SAVEKEY_VC2 Function

This function sets a package variable (`wwv_flow_utilities.g_val_vc2`) so that it can be retrieved using the function `KEYVAL_VC2`.

Syntax

```
HTMLDB_UTIL.SAVEKEY_VC2
    p_val IN VARCHAR2);
```

Parameters

Table 15–39 describes the parameters available in the `SAVEKEY_VC2` procedure.

Table 15–39 SAVEKEY_VC2 Parameters

Parameter	Description
`p_val`	The is the VARCHAR2 value to be saved

Example

```
DECLARE
VAL VARCHAR2(4000);
BEGIN
    VAL := HTMLDB_UTIL.SAVEKEY_VC2(
        p_val => 'XXX');
END;
```

See Also: "KEYVAL_VC2 Function" on page 15-22

SET_ATTRIBUTE Procedure

This procedure sets the value of one of the attribute values (1 through 10) of a user in the Application Express accounts table.

Syntax

```
HTMLDB_UTIL.SET_ATTRIBUTE(
    p_userid IN NUMBER,
    p_attribute_number IN NUMBER,
    p_attribute_value  IN VARCHAR2);
```

Parameters

Table 15–40 describes the parameters available in the `SET_ATTRIBUTE` procedure.

Table 15–40 SET_ATTRIBUTE Parameters

Parameter	Description
`p_userid`	The numeric ID of the user account
`p_attribute_number`	Attribute number in the user record (1 through 10)
`p_attribute_value`	Value of the attribute located by `p_attribute_number` to be set in the user record

Example

```
DECLARE VAL VARCHAR2(30);
BEGIN
    HTMLDB_UTIL.SET_ATTRIBUTE (
        p_userid => htmldb_util.get_user_id(p_username => 'SCOTT'),
        p_attribute_number => 1,
        p_attribute_value => 'foo');
END;
```

SET_EMAIL Procedure

This procedure updates a user account with a new email address. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
HTMLDB_UTIL.SET_EMAIL(
    p_userid IN NUMBER,
    p_email  IN VARCHAR2);
```

Parameters

Table 15–41 describes the parameters available in the `SET_EMAIL` procedure.

Table 15–41 SET_EMAIL Parameters

Parameter	Description
`p_userid`	The numeric ID of the user account
`p_email`	The email address to be saved in user account

Example

```
BEGIN
HTMLDB_UTIL.SET_EMAIL(
    p_userid  => '888883232',
    P_email   => 'scott.scott@oracle.com');
END;
```

SET_FIRST_NAME Procedure

This procedure updates a user account with a new `FIRST_NAME` value. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
HTMLDB_UTIL.SET_FIRST_NAME(
    p_userid      IN NUMBER,
    p_first_name  IN VARCHAR2);
```

Parameters

Table 15–42 describes the parameters available in the `SET_FIRST_NAME` procedure.

Table 15–42 SET_FIRST_NAME Parameters

Parameter	Description
`p_userid`	The numeric ID of the user account
`p_first_name`	`FIRST_NAME` value to be saved in user account

Example

```
BEGIN
HTMLDB_UTIL.SET_FIRST_NAME(
    p_userid       => '888883232',
    P_first_name   => 'Scott');
END;
```

SET_LAST_NAME Procedure

This procedure updates a user account with a new `LAST_NAME` value. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
HTMLDB_UTIL.SET_LAST_NAME(
    p_userid      IN NUMBER,
    p_last_name  IN VARCHAR2);
```

Parameters

Table 15–43 describes the parameters available in the `SET_LAST_NAME` procedure.

Table 15–43 SET_LAST_NAME Parameters

Parameter	Description
`p_userid`	The numeric ID of the user account

Table 15–43 (Cont.) SET_LAST_NAME Parameters

Parameter	Description
`p_last_name`	`LAST_NAME` value to be saved in the user account

Example

```
BEGIN
HTMLDB_UTIL.SET_LAST_NAME(
    p_userid       => '888883232',
    p_last_name   => 'SMITH');
END;
```

SET_USERNAME Procedure

This procedure updates a user account with a new `USER_NAME` value. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
HTMLDB_UTIL.USERNAME(
    p_userid      IN NUMBER,
    p_username    IN VARCHAR2);
```

Parameters

Table 15–44 describes the parameters available in the `SET_USERNAME` procedure.

Table 15–44 SET_USERNAME Parameters

Parameter	Description
`p_userid`	The numeric ID of the user account
`p_username`	`USER_NAME` value to be saved in the user account

Example

```
BEGIN
HTMLDB_UTIL.SET_USERNAME(
    p_userid       => '888883232',
    P_username   => 'USER-XRAY');
END;
```

SET_PREFERENCE Procedure

This procedure sets a preference that will persist beyond the user's current session.

Syntax

```
HTMLDB_UTIL.SET_PREFERENCE (
    p_preference   IN    VARCHAR2 DEFAULT NULL,
    p_value        IN    VARCHAR2 DEFAULT NULL,
    p_user         IN    VARCHAR2 DEFAULT NULL);
```

Parameters

Table 15–45 describes the parameters available in the `SET_PREFERENCE` procedure.

Table 15–45 SET_PREFERENCE Parameters

Parameter	Description
`p_preference`	Name of the preference (case-sensitive)
`p_value`	Value of the preference
`p_user`	User for whom the preference is being set

Example

```
BEGIN
       HTMLDB_UTIL.SET_PREFERENCE(
             p_preference => 'default_view',
             p_value      => 'WEEKLY',
             p_user       => :APP_USER);
END;
```

SET_SESSION_STATE Procedure

This procedure sets session state for a current Oracle Application Express session.

Syntax

```
HTMLDB_UTIL.SET_SESSION_STATE (
    p_name     IN    VARCHAR2 DEFAULT NULL,
    p_value    IN    VARCHAR2 DEFAULT NULL);
```

Parameters

Table 15–46 describes the parameters available in the `SET_SESSION_STATE` procedure.

Table 15–46 SET_SESSION_STATE Parameters

Parameter	Description
`p_name`	Name of the application-level or page-level item for which you are setting sessions state
`p_value`	Value of session state to set

Example

```
BEGIN
HTMLDB_UTIL.SET_SESSION_STATE('my_item','myvalue');
END;
```

STRING_TO_TABLE Function

Given a string, this function returns a PL/SQL array of type `HTMLDB_APPLICATION_GLOBAL.VC_ARR2`. This array is a `VARCHAR2(32767)` table.

Syntax

```
HTMLDB_UTIL.STRING_TO_TABLE (
    p_string       IN VARCHAR2,
    p_separator    IN VARCHAR2 DEFAULT ':')
    RETURN HTMLDB_APPLICATION_GLOBAL.VC_ARR2;
```

Parameters

Table 15–47 describes the parameters available in the `STRING_TO_TABLE` function.

Table 15–47 STRING_TO_TABLE Parameters

Parameter	Description
`p_string`	String to be converted into a PL/SQL table of type `HTMLDB_APPLICATION_GLOBAL.VC_ARR2`
`p_separator`	String separator. The default is a colon

Example

```
DECLARE
       l_vc_arr2    HTMLDB_APPLICATION_GLOBAL.VC_ARR2;
BEGIN
       l_vc_arr2 := HTMLDB_UTIL.STRING_TO_TABLE('One:Two:Three');
       FOR z IN 1..l_vc_arr2.count LOOP
             htp.p(l_vc_arr2(z));
       END LOOP;
END;
```

TABLE_TO_STRING Function

Given a a PL/SQL table of type `HTMLDB_APPLICATION_GLOBAL.VC_ARR2`, this function returns a delimited string separated by the supplied separator, or by the default separator, a colon (:).

Syntax

```
HTMLDB_UTIL.TABLE_TO_STRING (
    p_table       IN     HTMLDB_APPLICATION_GLOBAL.VC_ARR2,
    p_string      IN     VARCHAR2 DEFAULT ':')
    RETURN VARCHAR2;
```

Parameters

Table 15–48 describes the parameters available in the `TABLE_TO_STRING` function.

Table 15–48 TABLE_TO_STRING Parameters

Parameter	Description
`p_string`	String separator. Default separator is a colon (:)
`p_table`	PL/SQL table that is to be converted into a delimited string

Example

```
DECLARE
       l_string     VARCHAR2(255);
       l_vc_arr2    HTMLDB_APPLICATION_GLOBAL.VC_ARR2;
BEGIN
       l_vc_arr2 := HTMLDB_UTIL.STRING_TO_TABLE('One:Two:Three');

       l_string := HTMLDB_UTIL.TABLE_TO_STRING(l_vc_arr2);
END;
```

URL_ENCODE Function

This function encodes (into hexadecimal) all special characters that include spaces, question marks, and ampersands.

Syntax

```
HTMLDB_UTIL.URL_ENCODE (
    p_url   IN    VARCHAR2)
    RETURN VARCHAR2;
```

Parameters

Table 15–49 describes the parameters available in the `URL_ENCODE` function.

Table 15–49 URL_ENCODE Parameters

Parameter	Description
`p_url`	The string to be encoded

Example

```
DECLARE
      l_url  VARCHAR2(255);
BEGIN
      l_url := HTMLDB_UTIL.URL_ENCODE('http://www.myurl.com?id=1&cat=foo');
END;
```

HTMLDB_MAIL

You can use the `HTMLDB_MAIL` package to send an email from an Oracle Application Express application. This package is built on top of the Oracle supplied `UTL_SMTP` package. Because of this dependence, the `UTL_SMTP` package must be installed and functioning in order to use HTMLDB_MAIL.

> **See Also:** *Oracle Database PL/SQL Packages and Types Reference* for more information about the UTL_SMTP package

`HTMLDB_MAIL` contains two procedures. Use `HTMLDB_MAIL.SEND` to send an outbound email message from your application. Use `HTMLDB_MAIL.PUSH_QUEUE` to deliver mail messages stored in `HTMLDB_MAIL_QUEUE`.

Topics in this section include:

- SEND Procedure
- PUSH_QUEUE Procedure

> **Note:** The most efficient approach to sending email is to create a background job (using a DBMS_JOB package) to periodically send all mail messages stored in the active mail queue.

> **See Also:** "Sending Email from an Application" on page 8-17

SEND Procedure

This procedure sends an outbound email message from an application. Although you can use this procedure to pass in either a VARCHAR2 or a CLOB to p_body and p_body_html, the data types must be the same. In other words, you cannot pass a CLOB to P_BODY and a VARCHAR2 to p_body_html.

When using HTMLDB_MAIL.SEND, remember the following:

- **No single line may exceed 1000 characters.** The SMTP/MIME specification dictates that no single line shall exceed 1000 characters. To comply with this restriction, you must add a carriage return or line feed characters to break up your p_body or p_body_html parameters into chunks of 1000 characters or less. Failing to do so will result in erroneous email messages, including partial messages or messages with extraneous exclamation points.
- **Plain text and HTML email content.** Passing a value to p_body, but not p_body_html results in a plain text message. Passing a value to p_body and p_body_html yields a multi-part message that includes both plain text and HTML content. The settings and capabilities of the recipient's email client determine what displays. Although most modern email clients can read a HTML formatted email, remember that some users disable this functionality to address security issues.
- **Avoid images.** When referencing images in p_body_html using the <img /> tag, remember that the images must be accessible to the recipient's email client in order for them to see the image.

  For example, suppose you reference an image on your network called hello.gif as follows:

  ```
  <img src="http://someserver.com/hello.gif" alt="Hello" />]
  ```

  In this example, the image is not attached to the email, but is referenced by the email. For the recipient to see it, they must be able to access the image using a Web browser. If the image is inside a firewall and the recipient is outside of the firewall, the image will not display. For this reason, avoid using images. If you must include images, be sure to include the ALT attribute to provide a textual description in the event the image is not accessible.

Syntax

```
HTMLDB_MAIL.SEND(
    p_to                        IN    VARCHAR2,
    p_from                      IN    VARCHAR2,
    p_body                      IN  [ VARCHAR2 | CLOB ],
    p_body_html                 IN  [ VARCHAR2 | CLOB ] DEFAULT,
    p_subj                      IN    VARCHAR2 DEFAULT)
    p_cc                        IN    VARCHAR2 DEFAULT)
    p_bcc                       IN    VARCHAR2 DEFAULT);
```

Parameters

Table 15–50 describes the parameters available in the SEND procedure.

Table 15–50 Send Parameters

Parameter	Description
p_to	Valid email address to which the email will be sent (required). For multiple email addresses, use a comma separated list

Table 15–50 (Cont.) Send Parameters

Parameter	Description
`p_from`	Email address from which the email will be sent (required). This email address must be a valid address. Otherwise, the message will not be sent
`p_body`	Body of the email in plain text, not HTML (required). If a value is passed to `p_body_html`, then this is the only text the recipient sees. If a value is not passed to `p_body_html`, then this text only displays for email clients that do not support HTML or have HTML disabled. A carriage return or line feed (CRLF) must be included every 1000 characters
`p_body_html`	Body of the email in HTML format. This must be a full HTML document including the `<html>` and `<body>` tags. A single line cannot exceed 1000 characters without a carriage return or line feed (CRLF)
`p_subj`	Subject of the email
`p_cc`	Valid email addresses to which the email is copied. For multiple email addresses, use a comma separated list
`p_bcc`	Valid email addresses to which the email is blind copied. For multiple email addresses, use a comma separated list

Examples

The following example demonstrates how to use `HTMLDB_MAIL.SEND` to send a plain text email message from an application.

```
-- Example One: Plain Text only message
DECLARE
    l_body      CLOB;
BEGIN
    l_body := 'Thank you for your interest in the HTMLDB_MAIL
package.'||utl_tcp.crlf||utl_tcp.crlf;
    l_body := l_body ||'  Sincerely,'||utl_tcp.crlf;
    l_body := l_body ||'  The HTMLDB Dev Team'||utl_tcp.crlf;
    htmldb_mail.send(
        p_to       => 'some_user@somewhere.com',   -- change to your email address
        p_from     => 'some_sender@somewhere.com', -- change to a real senders
email address
        p_body     => l_body,
        p_subj     => 'HTMLDB_MAIL Package - Plain Text message');
END;
/
```

The following example demonstrates how to use `HTMLDB_MAIL.SEND` to send a HTML email message from an application. Remember, you must include a carriage return or line feed (CRLF) every 1000 characters. The example that follows uses `utl_tcp.crlf`.

```
-- Example Two: Plain Text / HTML message
DECLARE
    l_body      CLOB;
    l_body_html CLOB;
BEGIN
    l_body := 'To view the content of this message, please use an HTML enabled
mail client.'||utl_tcp.crlf;

    l_body_html := '<html>
                  <head>
```

```
                        <style type="text/css">
                          body{font-family: Arial, Helvetica, sans-serif;
                               font-size:10pt;
                               margin:30px;
                               background-color:#ffffff;}

                          span.sig{font-style:italic;
                                   font-weight:bold;
                                   color:#811919;}
                        </style>
                      </head>
                      <body>'||utl_tcp.crlf;
    l_body_html := l_body_html ||'<p>Thank you for your interest in the
<strong>HTMLDB_MAIL</strong> package.</p>'||utl_tcp.crlf;
    l_body_html := l_body_html ||'  Sincerely,<br />'||utl_tcp.crlf;
    l_body_html := l_body_html ||'  <span class="sig">The HTMLDB Dev
Team</span><br />'||utl_tcp.crlf;
    htmldb_mail.send(
     p_to        => 'some_user@somewhere.com',   -- change to your email address
     p_from      => 'some_sender@somewhere.com', -- change to a real senders email
address
     p_body      => l_body,
     p_body_html => l_body_html,
     p_subj      => 'HTMLDB_MAIL Package - HTML formatted message');
END;
/
```

PUSH_QUEUE Procedure

Oracle Application Express stores unsent email messages in a table named HTMLDB_MAIL_QUEUE. You can manually deliver mail messages stored in this queue to the specified SMTP gateway by invoking the HTMLDB_MAIL.PUSH_QUEUE procedure.

Oracle Application Express logs successfully submitted message in the table HTMLDB_MAIL_LOG with the timestamp reflecting your server's local time. Keep in mind, the most efficient approach to sending email is to create a background job (using a DBMS_JOB package) to periodically send all mail messages stored in the active mail queue.

See Also: "Sending Email Using a Background Job" on page 8-17

Syntax

```
HTMLDB_MAIL.PUSH_QUEUE(
    p_smtp_hostname             IN    VARCHAR2 DEFAULT,
    p_smtp_portno               IN    NUMBER   DEFAULT;
```

Parameters

Table 15–51 describes the parameters available in the PUSH_QUEUE procedure.

Table 15–51 PUSH_QUEUE Parameters

Parameters	Description
p_smtp_hostname	SMTP gateway host name
p_smtp_portno	SMTP gateway port number

Note that these parameter values are provided for backward compatibility, but their respective values are ignored. The SMTP gateway hostname and SMTP gateway port number are exclusively derived from the Application Express environment settings when sending e-mail.

See Also: "Configuring Email Environment Settings" on page 8-16

Example

The following example demonstrates the use of the `HTMLDB_MAIL.PUSH_QUEUE` procedure using a shell script. This example only applies to UNIX/LINUX installations. In this example, the SMTP gateway host name is defined as `smtp01.oracle.com` and the SMTP gateway port number is `25`.

```
SQLPLUS / <<EOF
FLOWS_020100.HTMLDB_MAIL.PUSH_QUEUE('smtp01.oracle.com','25');
DISCONNECT
EXIT
EOF
```

See Also: "Sending Email from an Application" on page 8-17

HTMLDB_ITEM

You can use the `HTMLDB_ITEM` package to create form elements dynamically based on a SQL query instead of creating individual items page by page.

Topics in this section include:

- CHECKBOX Function
- DATE_POPUP Function
- DISPLAY_AND_SAVE Function
- HIDDEN Function
- MD5_CHECKSUM Function
- MD5_HIDDEN Function
- MULTI_ROW_UPDATE Procedure
- POPUP_FROM_LOV Function
- POPUP_FROM_QUERY Function
- POPUPKEY_FROM_LOV Function
- POPUPKEY_FROM_QUERY Function
- RADIOGROUP Function
- SELECT_LIST Function
- SELECT_LIST_FROM_LOV Function
- SELECT_LIST_FROM_LOV_XL Function
- SELECT_LIST_FROM_QUERY Function
- SELECT_LIST_FROM_QUERY_XL Function
- TEXTAREA Function
- TEXT Function

- TEXT_FROM_LOV Function
- TEXT_FROM_LOV_QUERY Function

CHECKBOX Function

This function creates check boxes.

Syntax

```
HTMLDB_ITEM.CHECKBOX(
    p_idx                       IN    NUMBER,
    p_value                     IN    VARCHAR2 DEFAULT,
    p_attributes                IN    VARCHAR2 DEFAULT,
    p_checked_values            IN    VARCHAR2 DEFAULT,
    p_checked_values_delimiter  IN    VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 15–52 describes the parameters available in the `CHECKBOX` function.

Table 15–52 CHECKBOX Parameters

Parameter	Description
`p_idx`	Number that determines which `HTMLDB_APPLICATION` global variable will be used. Valid range of values is 1 to 50. For example 1 creates `F01` and 2 creates `F02`
`p_value`	Value of a check box, hidden field, or input form item
`p_attributes`	Controls HTML tag attributes (such as disabled)
`p_checked_values`	Values to be checked by default
`p_checked_values_delimiter`	Delimits the values in the previous parameter, `p_checked_values`

Examples of Default Check Box Behavior

The following example demonstrates how to create a selected check box for each employee in the `emp` table.

```
SELECT HTMLDB_ITEM.CHECKBOX(1,empno,'CHECKED') " ",
       ename,
       job
FROM   emp
ORDER BY 1
```

The following example demonstrates how to have all check boxes for employees display without being selected.

```
SELECT HTMLDB_ITEM.CHECKBOX(1,empno) " ",
       ename,
       job
FROM   emp
ORDER BY 1
```

The following example demonstrates how to select the check boxes for employees who work in department 10.

```
SELECT HTMLDB_ITEM.CHECKBOX(1,empno,DECODE(deptno,10,'CHECKED',null)) " ",
       ename,
       job
FROM   emp
ORDER BY 1
```

The next example demonstrates how to select the check boxes for employees who work in department 10 or department 20.

```
SELECT HTMLDB_ITEM.CHECKBOX(1,deptno,NULL,'10:20',':') " ",
       ename,
       job
FROM   emp
ORDER BY 1
```

Creating an On-Submit Process

If you are using check boxes in your application, you might need to create an On Submit process to perform a specific type of action on the selected rows. For example, you could have a Delete button that utilizes the following logic:

```
SELECT HTMLDB_ITEM.CHECKBOX(1,empno) " ",
       ename,
       job
FROM   emp
ORDER  by 1
```

Consider the following sample on-submit process:

```
FOR I in 1..HTMLDB_APPLICATION.G_F01.COUNT LOOP
    DELETE FROM emp WHERE empno = to_number(HTMLDB_APPLICATION.G_F01(i));
END LOOP;
```

DATE_POPUP Function

Use this function with forms that include date fields. The `DATE_POPUP` function dynamically generates a date field that has a popup calendar button.

Syntax

```
HTMLDB_ITEM.DATE_POPUP(
    p_idx          IN    NUMBER,
    p_row          IN    NUMBER,
    p_value        IN    VARCHAR2 DEFAULT,
    p_date_format  IN    DATE DEFAULT,
    p_size         IN    NUMBER DEFAULT,
    p_maxlength    IN    NUMBER DEFAULT,
    p_attributes   IN    VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 15–53 describes the parameters available in the `DATE_POPUP` function.

Table 15–53 DATE_POPUP Parameters

Parameter	Description
`p_idx`	Number that determines which `HTMLDB_APPLICATION` global variable will be used.Valid range of values is 1 to 50. For example, 1 creates `F01` and 2 creates `F02`
`p_row`	This parameter is deprecated. Anything specified for this value will be ignored
`p_value`	Value of a field item
`p_date_format`	Valid database date format
`p_size`	Controls HTML tag attributes (such as disabled)
`p_maxlength`	Determines the maximum number of enterable characters. Becomes the maxlength attribute of the `<input>` HTML tag
`p_attributes`	Extra HTML parameters you want to add

See Also: *Oracle Database SQL Reference* for information about the `TO_CHAR` or `TO_DATE` functions

Example

The following example demonstrates how to use `HTMLDB_ITEM.DATE_POPUP` to create popup calendar buttons for the `hiredate` column.

```
SELECT
  empno,
  HTMLDB_ITEM.HIDDEN(1,empno)||
  HTMLDB_ITEM.TEXT(2,ename) ename,
  HTMLDB_ITEM.TEXT(3,job) job,
  mgr,
  HTMLDB_ITEM.DATE_POPUP(4,rownum,hiredate,'dd-mon-yyyy') hd,
  HTMLDB_ITEM.TEXT(5,sal) sal,
  HTMLDB_ITEM.TEXT(6,comm) comm,
  deptno
FROM emp
ORDER BY 1
```

DISPLAY_AND_SAVE Function

Use this function to display an item as text, but save its value to session state.

Syntax

```
HTMLDB_ITEM.DISPLAY_AND_SAVE(
    p_idx         IN    NUMBER,
    p_value       IN    VARCHAR2 DEFAULT NULL
    p_item_id     IN    VARCHAR2 DEFAULT NULL,
    p_item_label  IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 15–54 describes the parameters available in the `DISPLAY_AND_SAVE` function.

Table 15–54 DISPLAY_AND_SAVE Parameters

Parameter	Description
`p_idx`	Number that determines which `HTMLDB_APPLICATION` global variable will be used.Valid range of values is 1 to 50. For example, 1 creates `F01` and 2 creates `F02`
`p_value`	Current value
`p_item_id`	HTML attribute ID for the `<input>` tag
`p_item_label`	Label of the text field item

Example

The following example demonstrates how to use the `HTMLDB_ITEM.DISPLAY_AND_SAVE` function.

```
SELECT HTMLDB_ITEM.DISPLAY_AND_SAVE(10,empno) c FROM emp
```

HIDDEN Function

This function dynamically generates hidden form items.

Syntax

```
HTMLDB_ITEM.HIDDEN(
    p_idx     IN    NUMBER,
    p_value   IN    VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 15–55 describes the parameters available in the `HIDDEN` function.

Table 15–55 HIDDEN Parameters

Parameter	Description
`p_idx`	Number to identify the item you want to generate. The number will determine which `G_FXX` global is populated **See Also:** "HTMLDB_APPLICATION" on page 15-59
`p_value`	Value of the hidden input form item

Example

Typically, the primary key of a table is stored as a hidden column and used for subsequent update processing, for example:

```
SELECT
  empno,
  HTMLDB_ITEM.HIDDEN(1,empno)||
  HTMLDB_ITEM.TEXT(2,ename) ename,
  HTMLDB_ITEM.TEXT(3,job) job,
  mgr,
  HTMLDB_ITEM.DATE_POPUP(4,rownum,hiredate,'dd-mon-yyyy') hiredate,
  HTMLDB_ITEM.TEXT(5,sal) sal,
  HTMLDB_ITEM.TEXT(6,comm) comm,
  deptno
FROM emp
```

```
ORDER BY 1
```

The previous query could use the following page process to process the results:

```
BEGIN
  FOR i IN 1..HTMLDB_APPLICATION.G_F01.COUNT LOOP
    UPDATE emp
    SET
      ename=HTMLDB_APPLICATION.G_F02(i),
      job=HTMLDB_APPLICATION.G_F03(i),
      hiredate=to_date(HTMLDB_APPLICATION.G_F04(i),'dd-mon-yyyy'),
      sal=HTMLDB_APPLICATION.G_F05(i),
      comm=HTMLDB_APPLICATION.G_F06(i)
    WHERE empno=to_number(HTMLDB_APPLICATION.G_F01(i));
  END LOOP;
END;
```

Note that the `G_F01` column (which corresponds to the hidden `EMPNO`) is used as the key to update each row.

MD5_CHECKSUM Function

This function passes values to `HTMLDB_ITEM.MULTI_ROW_UPDATE` and is used for lost update detection. Lost update detection ensures data integrity in applications where data can be accessed concurrently.

Syntax

```
HTMLDB_ITEM.MD5_CHECKSUM(
    p_value01   IN    VARCHAR2 DEFAULT,
    p_value02   IN    VARCHAR2 DEFAULT,
    p_value03   IN    VARCHAR2 DEFAULT,
    ...
    p_value50   IN    VARCHAR2 DEFAULT,
    p_col_sep   IN    VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 15–56 describes the parameters available in the `MD5_CHECKSUM` function.

Table 15–56 MD5_CHECKSUM Parameters

Parameter	Description
`p_value01`	Fifty available inputs. If no parameters are supplied, the default to null
...	
`p_value50`	
`p_col_sep`	String used to separate `p_value` inputs. Defaults to the pipe symbol (`|`)

Example

```
SELECT HTMLDB_ITEM.MD5_CHECKSUM(ename,job,sal)
FROM emp
```

MD5_HIDDEN Function

This function is used for lost update detection. Lost update detection ensures data integrity in applications where data can be accessed concurrently.

This function produces a hidden form field and includes 50 inputs. `HTMLDB_ITEM.MD5_HIDDEN` also produces an MD5 checksum using the Oracle database `DBMS_OBFUSCATION_TOOLKIT`:

```
UTL_RAW.CAST_TO_RAW(DBMS_OBFUSCATION_TOOLKIT.MD5())
```

An MD5 checksum provides data integrity through hashing and sequencing to ensure that data is not altered or stolen as it is transmitted over a network

Syntax

```
HTMLDB_ITEM.MD5_HIDDEN(
    p_idx       IN    NUMBER,
    p_value01   IN    VARCHAR2 DEFAULT,
    p_value02   IN    VARCHAR2 DEFAULT,
    p_value03   IN    VARCHAR2 DEFAULT,
    ...
    p_value50   IN    VARCHAR2 DEFAULT,
    p_col_sep   IN    VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 15–57 describes the parameters available in the `MD5_HIDDEN` function.

Table 15–57 MD5_HIDDEN Parameters

Parameter	Description
`p_idx`	Indicates the form element to be generated. For example, 1 equals `F01` and 2 equals `F02`. Typically the `p_idx` parameter is constant for a given column

`p_value01`
...
`p_value50`
	Fifty available inputs. Parameters not supplied default to null

`p_col_sep`	String used to separate `p_value` inputs. Defaults to the pipe symbol (`|`)

Example

The `p_idx` parameter specifies the FXX form element to be generated. In the following example, 7 generates `F07`. Also note that an HTML hidden form element will be generated.

```
SELECT HTMLDB_ITEM.MD5_HIDDEN(7,ename,job,sal), ename, job, sal FROM emp
```

MULTI_ROW_UPDATE Procedure

Use this procedure within a Multi Row Update process type. This procedure takes a string containing a multiple row update definition in the following format:

```
OWNER:TABLE:pk_column1,pk_idx:pk_column2,pk_idx2|col,idx:col,idx...
```

Syntax

```
HTMLDB_ITEM.MULTI_ROW_UPDATE(
    p_mru_string    IN    VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Example

To use this procedure indirectly within an application-level process, you need to create a query to generate a form of database data. The following example demonstrates how to create a multiple row update on the `emp` table.

```
SELECT
empno,
HTMLDB_ITEM.HIDDEN(1,empno),
HTMLDB_ITEM.HIDDEN(2,deptno),
HTMLDB_ITEM.TEXT(3,ename),
HTMLDB_ITEM.SELECT_LIST_FROM_QUERY(4,job,'SELECT DISTINCT job FROM emp'),
HTMLDB_ITEM.TEXT(5,sal),
HTMLDB_ITEM.TEXT(7,comm),
HTMLDB_ITEM.MD5_CHECKSUM(ename,job,sal,comm),
deptno
FROM emp
WHERE deptno = 20
```

Note the call to `HTMLDB_ITEM.MD5_CHECKSUM`, instead of `HTMLDB_ITEM.MD5_HIDDEN`. Since `HTMLDB_ITEM.MULTI_ROW_UPDATE` gets the checksum from `HTMLDB_APPLICATION.G_FCS`, you need to call `HTMLDB_ITEM.MD5_CHECKSUM` in order to populate `HTMLDB_APPLICATION.G_FCS` when the page is submitted. Additionally, the columns in `HTMLDB_ITEM.MD5_CHECKSUM` must be in the same order those in the `MULTI_ROW_UPDATE` process. These updates can then processed (or applied to the database) using an after submit page process of Multi Row Update in a string similar to the following:

```
SCOTT:emp:empno,1:deptno,2|ename,3:job,4:sal,5:comm,7:,:,:,:,
```

POPUP_FROM_LOV Function

This function generates an HTML popup select list from an application list of values (LOV). Similar from other available functions in the `HTMLDB_ITEM` package, `POPUP_FROM_LOV` function is designed to generate forms with `F01` to `F50` form array elements.

Syntax

```
HTMLDB_ITEM.POPUP_FROM_LOV(

    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT,
    p_lov_name         IN    VARCHAR2,
    p_width            IN    VARCHAR2 DEFAULT,
    p_max_length       IN    VARCHAR2 DEFAULT,
    p_form_index       IN    VARCHAR2 DEFAULT,
    p_escape_html      IN    VARCHAR2 DEFAULT,
    p_max_elements     IN    VARCHAR2 DEFAULT,
    p_attributes       IN    VARCHAR2 DEFAULT,
    p_ok_to_query      IN    VARCHAR2 DEFAULT,
    p_item_id          IN    VARCHAR2 DEFAULT NULL,
    p_item_label       IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 15–58 describes the some parameters in the `POPUP_FROM_LOV` function.

Table 15–58 POPUP_FROM_LOV Parameters

Parameter	Description
`p_idx`	Form element name. For example, `1` equals `F01` and `2` equals `F02`. Typically, `p_idx` is a constant for a given column
`p_value`	Form element current value. This value should be one of the values in the `p_lov_name` parameter
`p_lov_name`	Named LOV used for this popup
`p_width`	Width of the text box
`p_max_length`	Maximum number of characters that can be entered in the text box
`p_form_index`	HTML form on the page in which an item is contained. Defaults to 0 and rarely used. Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field that posts to a different Web site). If this form comes before the `#FORM_OPEN#` substitution string, then its index is zero and the form opened automatically by Oracle Application Express must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV that passes a value back to a form element.
`p_escape_html`	Replacements for special characters that require an escaped equivalent: ■ `&lt;` for < ■ `&gt;` for > ■ `&amp;` for & Range of values is `YES` and `NO`. If `YES`, special characters will be escaped. This parameter is useful if you know your query will return illegal HTML.
`p_max_elements`	Limit on the number of rows that can be returned by your query. Limits the performance impact of user searches. By entering a value in this parameter, you force the user to search for a narrower set of results.
`p_attributes`	Additional HTML attributes to use for the form item.
`p_ok_to_query`	Range of values is `YES` and `NO`. If `YES`, a popup returns first set of rows for the LOV. If `NO`, a search is initiated to return rows.
`p_item_id`	ID attribute of the form element.
`p_item_label`	Invisible label created for the item.

Example

The following example demonstrates a sample query the generates a popup from a LOV named `DEPT`.

```
SELECT HTMLDB_ITEM.POPUP_FROM_LOV (1,deptno,'DEPT_LOV') dt
FROM emp
```

POPUP_FROM_QUERY Function

This function generates an HTML popup select list from a query. Like other available functions in the HTMLDB_ITEM package, the POPUP_FROM_QUERY function is designed to generate forms with F01 to F50 form array elements.

Syntax

```
HTMLDB_ITEM.POPUP_FROM_QUERY(

    p_idx             IN    NUMBER,
    p_value           IN    VARCHAR2 DEFAULT,
    p_lov_query       IN    VARCHAR2,
    p_width           IN    VARCHAR2 DEFAULT,
    p_max_length      IN    VARCHAR2 DEFAULT,
    p_form_index      IN    VARCHAR2 DEFAULT,
    p_escape_html     IN    VARCHAR2 DEFAULT,
    p_max_elements    IN    VARCHAR2 DEFAULT,
    p_attributes      IN    VARCHAR2 DEFAULT,
    p_ok_to_query     IN    VARCHAR2 DEFAULT,
    p_item_id         IN    VARCHAR2 DEFAULT NULL,
    p_item_label      IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 15–59 describes the parameters in the POPUP_FROM_QUERY function.

Table 15–59 POPUP_FROM_QUERY Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, p_idx is a constant for a given column.
p_value	Form element current value. This value should be one of the values in the p_lov_query parameter.
p_lov_query	SQL query that is expected to select two columns (a display column and a return column). For example: `SELECT dname, deptno FROM dept`
p_width	Width of the text box.
p_max_length	Maximum number of characters that can be entered in the text box.
p_form_index	HTML form on the page in which an item is contained. Defaults to 0 and rarely used. Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field that posts to a different Web site). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by Oracle Application Express must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV that passes a value back to a form element.

Table 15–59 (Cont.) POPUP_FROM_QUERY Parameters

Parameter	Description
`p_escape_html`	Replacements for special characters that require an escaped equivalent. ■ `&lt;` for `<` ■ `&gt;` for `>` ■ `&amp;` for `&` Range of values is `YES` and `NO`. If `YES`, special characters will be escaped. This parameter is useful if you know your query will return illegal HTML.
`p_max_elements`	Limit on the number of rows that can be returned by your query. Limits the performance impact of user searches. By entering a value in this parameter, you force the user to search for a narrower set of results.
`p_attributes`	Additional HTML attributes to use for the form item.
`p_ok_to_query`	Range of values is `YES` and `NO`. If `YES`, a popup returns the first set of rows for the LOV. If `NO`, a search is initiated to return rows.
`p_item_id`	ID attribute of the form element.
`p_item_label`	Invisible label created for the item.

Example

The following example demonstrates a sample query the generates a popup select list from the `emp` table.

```
SELECT HTMLDB_ITEM.POPUP_FROM_QUERY (1,deptno,'SELECT dname, deptno FROM dept') dt
FROM emp
```

POPUPKEY_FROM_LOV Function

This function generates a popup key select list from a shared list of values (LOV). Similar to other available functions in the `HTMLDB_ITEM` package, the `POPUPKEY_FROM_LOV` function is designed to generate forms with `F01` to `F50` form array elements.

Syntax

```
HTMLDB_ITEM.POPUPKEY_FROM_LOV(
    p_idx             IN    NUMBER,
    p_value           IN    VARCHAR2 DEFAULT,
    p_lov_name        IN    VARCHAR2,
    p_width           IN    VARCHAR2 DEFAULT,
    p_max_length      IN    VARCHAR2 DEFAULT,
    p_form_index      IN    VARCHAR2 DEFAULT,
    p_escape_html     IN    VARCHAR2 DEFAULT,
    p_max_elements    IN    VARCHAR2 DEFAULT,
    p_attributes      IN    VARCHAR2 DEFAULT,
    p_ok_to_query     IN    VARCHAR2 DEFAULT,
    RETURN VARCHAR2;
```

Although the text field associated with the popup displays in the first column in the LOV query, the actual value is specified in the second column in the query.

Parameters

Table 15–60 describes the some parameters in the POPUPKEY_FROM_LOV function.

Table 15–60 POPUPKEY_FROM_LOV Parameters

Parameter	Description

p_idx	Identifies a form element name. For example, 1 equals F01 and 2 equals F02. Typically, p_idx is a constant for a given column
	Because of the behavior of POPUPKEY_FROM_QUERY, the next index value should be p_idx + 1. For example:
	`SELECT HTMLDB_ITEM.POPUPKEY_FROM_LOV (1,deptno,'DEPT') dt,`
	`HTMLDB_ITEM.HIDDEN(3,empno) eno`

p_value	Indicates the current value. This value should be one of the values in the P_LOV_NAME parameter.
p_lov_name	Identifies a named LOV used for this popup.
p_width	Width of the text box.
p_max_length	Maximum number of characters that can be entered in the text box.
p_form_index	HTML form on the page in which an item is contained. Defaults to 0 and rarely used. Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field that posts to a different Web site). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by Oracle Application Express must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV that passes a value back to a form element.
p_escape_html	Replacements for special characters that require an escaped equivalent. ■ &lt; for < ■ &gt; for > ■ &amp; for & This parameter is useful if you know your query will return illegal HTML.
p_max_elements	Limit on the number of rows that can be returned by your query. Limits the performance impact of user searches. By entering a value in this parameter, you force the user to search for a narrower set of results.
p_attributes	Additional HTML attributes to use for the form item.
p_ok_to_query	Range of values is YES and NO. If YES, a popup returns the first set of rows for the LOV. If NO, a search is initiated to return rows.

Example

The following example demonstrates how to generate a popup key select list from a shared list of values (LOV).

```
SELECT HTMLDB_ITEM.POPUPKEY_FROM_LOV (1,deptno,'DEPT') dt
FROM emp
```

POPUPKEY_FROM_QUERY Function

This function generates a popup key select list from a SQL query. Similar to other available functions in the HTMLDB_ITEM package, the POPUPKEY_FROM_QUERY function is designed to generate forms with F01 to F50 form array elements.

Syntax

```
HTMLDB_ITEM.POPUPKEY_FROM_QUERY(
    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT,
    p_lov_query        IN    VARCHAR2,
    p_width            IN    VARCHAR2 DEFAULT,
    p_max_length       IN    VARCHAR2 DEFAULT,
    p_form_index       IN    VARCHAR2 DEFAULT,
    p_escape_html      IN    VARCHAR2 DEFAULT,
    p_max_elements     IN    VARCHAR2 DEFAULT,
    p_attributes       IN    VARCHAR2 DEFAULT,
    p_ok_to_query      IN    VARCHAR2 DEFAULT,
    p_item_id          IN    VARCHAR2 DEFAULT NULL,
    p_item_label       IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 15–61 describes the some parameters in the POPUPKEY_FROM_QUERY function.

Table 15–61 POPUPKEY_FROM_QUERY Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, p_idx is a constant for a given column. Because of the behavior of POPUPKEY_FROM_QUERY, the next index value should be p_idx + 1. For example: `SELECT HTMLDB_ITEM.POPUPKEY_FROM_QUERY (1,deptno,'SELECT dname, deptno FROM dept') dt, HTMLDB_ITEM.HIDDEN(3,empno) eno`
p_value	Form element current value. This value should be one of the values in the P_LOV_QUERY parameter.
p_lov_query	LOV query used for this popup.
p_width	Width of the text box.
p_max_length	Maximum number of characters that can be entered in the text box.
p_form_index	HTML form on the page in which an item is contained. Defaults to 0 and rarely used. Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field that posts to a different Web site). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by Oracle Application Express must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV that passes a value back to a form element.

Table 15–61 (Cont.) POPUPKEY_FROM_QUERY Parameters

Parameter	Description
`p_escape_html`	Replacements for special characters that require an escaped equivalent. ▪ `&lt;` for < ▪ `&gt;` for > ▪ `&amp;` for & This parameter is useful if you know your query will return illegal HTML.
`p_max_elements`	Limit on the number of rows that can be returned by your query. Limits the performance impact of user searches. By entering a value in this parameter, you force the user to search for a narrower set of results.
`p_attributes`	Additional HTML attributes to use for the form item.
`p_ok_to_query`	Range of values is `YES` and `NO`. If `YES`, a popup returns first set of rows for the LOV. If `NO`, a search is initiated to return rows.
`p_item_id`	ID attribute of the form element.
`p_item_label`	Invisible label created for the item.

Example

The following example demonstrates how to generate a popup select list from a SQL query.

```
SELECT HTMLDB_ITEM.POPUPKEY_FROM_QUERY (1,deptno,'SELECT dname, deptno FROM dept')
dt
FROM emp
```

RADIOGROUP Function

This function generates a radio group from a SQL query.

Syntax

```
HTMLDB_ITEM.RADIOGROUP(
    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT,
    p_selected_value   IN    VARCHAR2 DEFAULT,
    p_display          IN    VARCHAR2 DEFAULT,
    p_attributes       IN    VARCHAR2 DEFAULT,
    p_onblur           IN    VARCHAR2 DEFAULT,
    p_onchange         IN    VARCHAR2 DEFAULT,
    p_onfocus          IN    VARCHAR2 DEFAULT,)
    RETURN VARCHAR2;
```

Parameters

Table 15–62 describes the parameters available in the `RADIOGROUP` function.

Table 15–62 RADIOGROUP Parameters

Parameter	Description
`p_idx`	Number that determines which `HTMLDB_APPLICATION` global variable will be used. Valid range of values is `1` to `50`.For example `1` creates `F01` and `2` creates `F02`.

Table 15–62 (Cont.) RADIOGROUP Parameters

Parameter	Description
`p_value`	Value of the radio group.
`p_selected_value`	Value that should be selected.
`p_display`	Text to display next to the radio option.
`p_attributes`	Extra HTML parameters you want to add.
`p_onblur`	JavaScript to execute in the onBlur event.
`p_onchange`	JavaScript to execute in the onChange event.
`p_onfocus`	JavaScript to execute in the onFocus event.

Example

The following example demonstrates how to select department 20 from the `emp` table as a default in a radio group.

```
SELECT HTMLDB_ITEM.CHECKBOX(1,deptno,'20',dname) dt
FROM   dept
ORDER  BY 1
```

SELECT_LIST Function

This function dynamically generates a static select list. Similar to other functions available in the `HTMLDB_ITEM` package, these select list functions are designed to generate forms with `F01` to `F50` form array elements.

Syntax

```
HTMLDB_ITEM.SELECT_LIST(
    p_idx           IN   NUMBER,
    p_value         IN   VARCHAR2 DEFAULT,
    p_list_values   IN   VARCHAR2 DEFAULT,
    p_attributes    IN   VARCHAR2 DEFAULT,
    p_show_null     IN   VARCHAR2 DEFAULT,
    p_null_value    IN   VARCHAR2 DEFAULT,
    p_null_text     IN   VARCHAR2 DEFAULT,
    p_item_id       IN   VARCHAR2 DEFAULT,
    p_item_label    IN   VARCHAR2 DEFAULT,
    p_show_extra    IN   VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 15–63 describes the parameters available in the `SELECT_LIST` function.

Table 15–63 SELECT_LIST Parameters

Parameter	Description
`p_idx`	Form element name. For example, `1` equals `F01` and `2` equals `F02`. Typically the `P_IDX` parameter is constant for a given column.
`p_value`	Current value. This value should be a value in the `P_LIST_VALUES` parameter.

Table 15–63 (Cont.) SELECT_LIST Parameters

Parameter	Description
`p_list_values`	List of static values separated by commas. Displays values and returns values that are separated by semicolons. Note that this is only available in the `SELECT_LIST` function.
`p_attributes`	Extra HTML parameters you want to add.
`p_show_null`	Extra select option to enable the NULL selection. Range of values is `YES` and `NO`.
`p_null_value`	Value to be returned when a user selects the null option. Only relevant when `p_show_null` equals `YES`.
`p_null_text`	Value to be displayed when a user selects the null option. Only relevant when `p_show_null` equals YES.
`p_item_id`	HTML attribute ID for the <input> tag.
`p_item_label`	Label of the select list.
`p_show_extra`	Shows the current value even if the value of p_value is not located in the select list.

Example

The following example demonstrates a static select list that displays `Yes`, returns `Y`, defaults to `Y`, and generates a `F01` form item.

```
SELECT HTMLDB_ITEM.SELECT_LIST(1,'Y','Yes;Y,No;N')
FROM emp
```

SELECT_LIST_FROM_LOV Function

This function dynamically generates select lists from a shared list of values (LOV). Similar to other functions available in the `HTMLDB_ITEM` package, these select list functions are designed to generate forms with `F01` to `F50` form array elements.

Syntax

```
HTMLDB_ITEM.SELECT_LIST_FROM_LOV(
    p_idx           IN   NUMBER,
    p_value         IN   VARCHAR2 DEFAULT,
    p_lov           IN   VARCHAR2,
    p_attributes    IN   VARCHAR2 DEFAULT,
    p_show_null     IN   VARCHAR2 DEFAULT,
    p_null_value    IN   VARCHAR2 DEFAULT,
    p_null_text     IN   VARCHAR2 DEFAULT,
    p_item_id       IN   VARCHAR2 DEFAULT,
    p_item_label    IN   VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 15–64 describes the parameters available in the `SELECT_LIST_FROM_LOV` function.

Table 15–64 SELECT_LIST_FROM_LOV Parameters

Parameter	Description
`p_idx`	Form element name. For example, `1` equals `F01` and `2` equals `F02`. Typically, the `p_idx` parameter is constant for a given column.
`p_value`	Current value. This value should be a value in the `p_list_values` parameter.
`p_lov`	Text name of an application list of values. This list of values must be defined in your application. This parameter is used only by the `select_list_from_lov` function.
`p_attributes`	Extra HTML parameters you want to add.
`p_show_null`	Extra select option to enable the NULL selection. Range of values is `YES` and `NO`.
`p_null_value`	Value to be returned when a user selects the null option. Only relevant when `p_show_null` equals `YES`.
`p_null_text`	Value to be displayed when a user selects the null option. Only relevant when `p_show_null` equals `YES`.
`p_item_id`	HTML attribute ID for the `<input>` tag.
`p_item_label`	Label of the select list.

Example

The following example demonstrates a select list based on an LOV defined in the application.

```
SELECT HTMLDB_ITEM.SELECT_LIST_FROM_LOV(2,job,'JOB_FLOW_LOV')
FROM emp
```

SELECT_LIST_FROM_LOV_XL Function

This function dynamically generates very large select lists (greater than 32K) from a shared list of values (LOV). Similar to other functions available in the `HTMLDB_ITEM` package, these select list functions are designed to generate forms with `F01` to `F50` form array elements. This function is the same as `SELECT_LIST_FROM_LOV`, but its return value is CLOB. This enables you to use it in SQL queries where you need to handle a column value longer than 4000 characters.

Syntax

```
HTMLDB_ITEM.SELECT_LIST_FROM_LOV_XL(
    p_idx          IN   NUMBER,
    p_value        IN   VARCHAR2 DEFAULT,
    p_lov          IN   VARCHAR2,
    p_attributes   IN   VARCHAR2 DEFAULT,
    p_show_null    IN   VARCHAR2 DEFAULT,
    p_null_value   IN   VARCHAR2 DEFAULT,
    p_null_text    IN   VARCHAR2 DEFAULT,
    p_item_id      IN   VARCHAR2 DEFAULT,
    p_item_label   IN   VARCHAR2 DEFAULT)
    RETURN CLOB;
```

Parameters

Table 15–65 describes the parameters available in the SELECT_LIST_FROM_LOV_XL function.

Table 15–65 SELECT_LIST_FROM_LOV_XL Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, the p_idx parameter is constant for a given column.
p_value	Current value. This value should be a value in the p_list_values parameter.
p_lov	Text name of a list of values. This list of values must be defined in your application. This parameter is used only by the select_list_from_lov function.
p_attributes	Extra HTML parameters you want to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the null option. Only relevant when p_show_null equals YES.
p_null_text	Value to be displayed when a user selects the null option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Label of the select list.

Example

The following demonstrates a select list based on a LOV defined in the application.

```
SELECT HTMLDB_ITEM.SELECT_LIST_FROM_LOV_XL(2,job,'JOB_FLOW_LOV')
FROM emp
```

SELECT_LIST_FROM_QUERY Function

This function is the same as This function dynamically generates a select list from a query. Similar to other functions available in the HTMLDB_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements.

Syntax

```
HTMLDB_ITEM.SELECT_LIST_FROM_QUERY(
    p_idx           IN    NUMBER,
    p_value         IN    VARCHAR2 DEFAULT,
    p_query         IN    VARCHAR2,
    p_attributes    IN    VARCHAR2 DEFAULT,
    p_show_null     IN    VARCHAR2 DEFAULT,
    p_null_value    IN    VARCHAR2 DEFAULT,
    p_null_text     IN    VARCHAR2 DEFAULT,
    p_item_id       IN    VARCHAR2 DEFAULT,
    p_item_label    IN    VARCHAR2 DEFAULT,
    p_show_extra    IN    VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 15–66 describes the parameters available in the SELECT_LIST_FROM_QUERY function.

Table 15–66 SELECT_LIST_FROM_QUERY Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, the p_idx parameter is constant for a given column.
p_value	Current value. This value should be a value in the p_list_values parameter.
p_query	SQL query that is expected to select two columns, a display column, and a return column. For example: `SELECT dname, deptno FROM dept` Note that this is used only by the SELECT_LIST_FROM_QUERY function.
p_attributes	Extra HTML parameters you want to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the null option. Only relevant when p_show_null equals YES.
p_null_text	Value to be displayed when a user selects the null option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Label of the select list.
p_show_extra	Show the current value even if the value of p_value is not located in the select list.

Example

The following example demonstrates a select list based on a SQL query.

```
SELECT HTMLDB_ITEM.SELECT_LIST_FROM_QUERY(3,job,'SELECT DISTINCT job FROM emp')
FROM emp
```

SELECT_LIST_FROM_QUERY_XL Function

This function is the same as SELECT_LIST_FROM_QUERY, but its return value is a CLOB. This allows its use in SQL queries where you need to handle a column value longer than 4000 characters. Similar to other functions available in the HTMLDB_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements.

Syntax

```
HTMLDB_ITEM.SELECT_LIST_FROM_QUERY_XL(
    p_idx           IN    NUMBER,
    p_value         IN    VARCHAR2 DEFAULT,
    p_query         IN    VARCHAR2,
    p_attributes    IN    VARCHAR2 DEFAULT,
    p_show_null     IN    VARCHAR2 DEFAULT,
    p_null_value    IN    VARCHAR2 DEFAULT,
    p_null_text     IN    VARCHAR2 DEFAULT,
    p_item_id       IN    VARCHAR2 DEFAULT,
    p_item_label    IN    VARCHAR2 DEFAULT,
    p_show_extra    IN    VARCHAR2 DEFAULT)
    RETURN CLOB;
```

Parameters

Table 15–67 describes the parameters available in the `SELECT_LIST_FROM_QUERY_XL` function.

Table 15–67 SELECT_LIST_FROM_QUERY_XL Parameters

Parameter	Description
`p_idx`	Form element name. For example, `1` equals `F01` and `2` equals `F02`. Typically the `p_idx` parameter is constant for a given column.
`p_value`	Current value. This value should be a value in the `p_list_values` parameter.
`p_query`	SQL query that is expected to select two columns, a display column, and a return column. For example: `SELECT dname, deptno FROM dept` Note that this is used only by the `SELECT_LIST_FROM_QUERY_XL` function.
`p_attributes`	Extra HTML parameters you want to add.
`p_show_null`	Extra select option to enable the NULL selection. Range of values is `YES` and `NO`.
`p_null_value`	Value to be returned when a user selects the null option. Only relevant when `p_show_null` equals `YES`.
`p_null_text`	Value to be displayed when a user selects the null option. Only relevant when `p_show_null` equals `YES`.
`p_item_id`	HTML attribute ID for the `<input>` tag.
`p_item_label`	Label of the select list.
`p_show_extra`	Show the current value even if the value of `p_value` is not located in the select list.

Example

The following example demonstrates a select list based on a SQL query.

```
SELECT HTMLDB_ITEM.SELECT_LIST_FROM_QUERY_XL(3,job,'SELECT DISTINCT job FROM emp')
FROM emp
```

TEXTAREA Function

This function creates text areas.

Syntax

```
HTMLDB_ITEM.TEXTAREA(
    p_idx          IN    NUMBER,
    p_value        IN    VARCHAR2 DEFAULT NULL,
    p_rows         IN    NUMBER DEAULT 40,
    p_cols         IN    NUMBER DEFAULT 4
    p_attributes   IN    VARCHAR2 DEFAULT,
    p_item_id     IN    VARCHAR2 DEFAULT NULL,
    p_item_label  IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 15–68 describes the parameters available in the TEXTAREA function.

Table 15–68 TEXTAREA Parameters

Parameter	Description
p_idx	Number to identify the item you want to generate. The number will determine which G_FXX global is populated. **See Also:** "HTMLDB_APPLICATION" on page 15-59
p_value	Value of the text area item.
p_rows	Height of the text area (HTML rows attribute)
p_cols	Width of the text area (HTML column attribute).
p_attributes	Extra HTML parameters you want to add.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Label of the text area item.

Example

The following example demonstrates how to create a text area based on a SQL query.

```
SELECT HTMLDB_ITEM.TEXTAREA(3,ename,5,80) a
FROM emp
```

TEXT Function

This function generates text fields (or text input form items) from a SQL query.

Syntax

```
HTMLDB_ITEM.TEXT(
    p_idx         IN    NUMBER,
    p_value       IN    VARCHAR2 DEFAULT NULL,
    p_size        IN    NUMBER DEFAULT NULL,
    p_maxlength   IN    NUMBER DEFAULT NULL,
    p_attributes  IN    VARCHAR2 DEFAULT NULL,
    p_item_id     IN    VARCHAR2 DEFAULT NULL,
    p_item_label  IN    VARCHAR2 DEFAULT NULL)
```

Parameters

Table 15–69 describes the parameters available in the TEXT function.

Table 15–69 TEXT Parameters

Parameter	Description
`p_idx`	Number to identify the item you want to generate. The number will determine which `G_FXX` global is populated. **See Also:** "HTMLDB_APPLICATION" on page 15-59
`p_value`	Value of a text field item.
`p_size`	Controls HTML tag attributes (such as disabled).
`p_maxlength`	Maximum number of characters that can be entered in the text box.
`p_attributes`	Extra HTML parameters you want to add.
`p_item_id`	HTML attribute ID for the `<input>` tag.
`p_item_label`	Label of the text field item.

Example

The following sample query demonstrates how to generate one update field for each row. Note that the `ename`, `sal`, and `comm` columns use the `HTMLDB_ITEM.TEXT` function to generate an HTML text field for each row. Also, notice that each item in the query is passed a unique `p_idx` parameter to ensure that each column is stored in its own array.

```
SELECT
  empno,
  HTMLDB_ITEM.HIDDEN(1,empno)||
  HTMLDB_ITEM.TEXT(2,ename) ename,
  HTMLDB_ITEM.TEXT(3,job) job,
  mgr,
  HTMLDB_ITEM.DATE_POPUP(4,rownum,hiredate,'dd-mon-yyyy') hiredate,
  HTMLDB_ITEM.TEXT(5,sal) sal,
  HTMLDB_ITEM.TEXT(6,comm) comm,
  deptno
FROM emp
ORDER BY 1
```

TEXT_FROM_LOV Function

Use this function to display an item as text, deriving the display value of the named LOV.

Syntax

```
HTMLDB_ITEM.TEXT_FROM_LOV (
    p_value       IN    VARCHAR2 DEFAULT NULL,
    p_lov         IN    VARCHAR2,
    p_null_text   IN    VARCHAR2 DEFAULT '%')
    RETURN VARCHAR2;
```

Parameters

Table 15–70 describes the parameters available in the `TEXT_FROM_LOV` function.

Table 15–70 TEXT_FROM_LOV Parameters

Parameter	Description
`p_value`	Value of a field item.
`p_lov`	Text name of a shared list of values. This list of values must be defined in your application.
`p_null_text`	Value to be displayed when the value of the field item is null or a corresponding entry is not located for the value `p_value` in the list of values.

Example

The following example demonstrates how to derive the display value from a named LOV (`EMPNO_ENAME_LOV`).

```
SELECT HTMLDB_ITEM.TEXT_FROM_LOV(empno,'EMPNO_ENAME_LOV') c FROM emp
```

TEXT_FROM_LOV_QUERY Function

Use this function to display an item as text, deriving the display value from a list of values query.

Syntax

```
HTMLDB_ITEM.TEXT_FROM_LOV_QUERY (
    p_value       IN    VARCHAR2 DEFAULT NULL,
    p_query       IN    VARCHAR2,
    p_null_text   IN    VARCHAR2 DEFAULT '%')
    RETURN VARCHAR2;
```

Parameters

Table 15–71 describes the parameters available in the `TEXT_FROM_LOV_QUERY` function.

Table 15–71 TEXT_FROM_LOV_QUERY Parameters

Parameter	Description
`p_value`	Value of a field item.
`p_query`	SQL query that is expected to select two columns, a display column and a return column. For example: `SELECT dname, deptno FROM dept`
`p_null_text`	Value to be displayed when the value of the field item is null or a corresponding entry is not located for the value `p_value` in the list of values query.

Example

The following example demonstrates how to derive the display value from a query.

```
SELECT HTMLDB_ITEM.TEXT_FROM_LOV_QUERY(empno,'SELECT ename, empno FROM emp') c
from emp
```

HTMLDB_APPLICATION

The HTMLDB_APPLICATION package is a PL/SQL package that implements the Oracle Application Express rendering engine. You can use this package to take advantage of a number of global variables. Table 15–72 describes the global variables available in the HTMLDB_APPLICATION package.

Table 15–72 Global Variables Available in HTMLDB_APPLICATION

Global Variable	Description
G_USER	Specifies the currently logged in user.
G_FLOW_ID	Specifies the ID of the currently running application.
G_FLOW_STEP_ID	Specifies the ID of the currently running page.
G_FLOW_OWNER	Specifies the schema to parse for the currently running application.
G_REQUEST	Specifies the value of the request variable most recently passed to or set within the show or accept modules.

Topics in this section include:

- Referencing Arrays
- Referencing Values Within an On Submit Process
- Converting an Array to a Single Value

Referencing Arrays

Items are typically HTML form elements such as text fields, select lists, and check boxes. When you create a new form item using a wizard, the wizard uses a standard naming format. The naming format provides a handle so you can retrieve the value of the item later on.

If you need to create your own items, you can access them after a page is submitted by referencing HTMLDB_APPLICATION.G_F01 to HTMLDB_APPLICATION.G_F50 arrays. You can create your own HTML form fields by providing the input parameters using the format F01, F02, F03 and so on. You can create up to 50 input parameters ranging from F01 to F50, for example:

```
<INPUT TYPE="text" NAME="F01" SIZE="32" MAXLENGTH="32" VALUE="some value">

<TEXTAREA NAME="F02" ROWS=4 COLS=90 WRAP="VIRTUAL">this is the example of a text
area.</TEXTAREA>

<SELECT NAME="F03" SIZE="1">
<OPTION VALUE="abc">abc
<OPTION VALUE="123">123
</SELECT>
```

Because the F01 to F50 input items are declared as PL/SQL arrays, you can have multiple items named the same value. For example:

```
<INPUT TYPE="text" NAME="F01" SIZE="32" MAXLENGTH="32" VALUE="array element 1">
<INPUT TYPE="text" NAME="F01" SIZE="32" MAXLENGTH="32" VALUE="array element 2">
<INPUT TYPE="text" NAME="F01" SIZE="32" MAXLENGTH="32" VALUE="array element 3">
```

Note that following PL/SQL code produces the same HTML as show in the previous example.

```
FOR i IN 1..3 LOOP
HTMLDB_ITEM.TEXT(P_IDX        => 1,
 p_value      =>'array element '||i ,
 p_size       =>32,
 p_maxlength  =>32);
END LOOP;
```

Referencing Values Within an On Submit Process

You can reference the values posted by an HTML form using the PL/SQL variable `HTMLDB_APPLICATION.G_F01` to `HTMLDB_APPLICATION.G_F50`. Because this element is an array, you can reference values directly, for example:

```
FOR i IN 1.. HTMLDB_APPLICATION.G_F01.COUNT LOOP
    htp.p('element '||I||' has a value of '||HTMLDB_APPLICATION.G_F01(i));
END LOOP;
```

Note that check boxes displayed using `HTMLDB_ITEM.CHECKBOX` will only contain values in the `HTMLDB_APPLICATION` arrays for those rows which are checked. Unlike other items (TEXT, TEXTAREA, DATE_POPUP) which can contain an entry in the corresponding `HTMLDB_APPLICATION` array for every row submitted, a check box will only have an entry in the `HTMLDB_APPLICATION` array if it is selected.

Converting an Array to a Single Value

You can also use Oracle Application Express public utility functions to convert an array into a single value. The resulting string value is a colon-separated list of the array element values. The resulting string value is a colon-separated list of the array element values. For example:

```
htp.p(HTMLDB_UTIL.TABLE_TO_STRING(HTMLDB_APPLICATION.G_F01));
```

This function enables you to reference `G_F01` to `G_F50` values in an application process that performs actions on data. The following sample process demonstrates how values are inserted into a table:

```
FOR i IN 1..HTMLDB_APPLICATION.G_F01.COUNT LOOP
    INSERT INTO my_table (my_column) VALUES HTMLDB_APPLICATION.G_F01(i);
END LOOP;
```

HTMLDB_CUSTOM_AUTH

You can use the `HTMLDB_CUSTOM_AUTH` package to perform various operations related to authentication and session management.

Topics in this section include:

- APPLICATION_PAGE_ITEM_EXISTS Function
- CURRENT_PAGE_IS_PUBLIC Function
- DEFINE_USER_SESSION Procedure
- GET_COOKIE_PROPS Procedure
- GET_LDAP_PROPS Procedure
- GET_NEXT_SESSION_ID Function
- GET_SESSION_ID_FROM_COOKIE Function

- GET_USERNAME Function
- GET_SECURITY_GROUP_ID Function
- GET_SESSION_ID Function
- GET_USER Function
- IS_SESSION_VALID Function
- LOGIN Procedure
- LOGOUT Procedure
- POST_LOGIN Procedure
- SESSION_ID_EXISTS Function
- SET_USER Procedure
- SET_SESSION_ID Procedure
- SET_SESSION_ID_TO_NEXT_VALUE Procedure

APPLICATION_PAGE_ITEM_EXISTS Function

This function checks for the existence of page-level item within an application. This function requires the parameter `p_item_name`. This function returns a Boolean value (true or false).

Syntax

```
FUNCTION APPLICATION_PAGE_ITEM_EXISTS(
    p_item_name    IN    VARCHAR2)
RETURN BOOLEAN;
```

CURRENT_PAGE_IS_PUBLIC Function

This function checks whether the current page's authentication attribute is set to **Page Is Public** and returns a Boolean value (true or false)

See Also: "Editing Page Attributes" on page 4-20 and "Security" on page 4-22 for information about setting this page attribute

Syntax

```
FUNCTION CURRENT_PAGE_IS_PUBLIC
RETURN BOOLEAN;
```

DEFINE_USER_SESSION Procedure

This procedure combines the `SET_USER` and `SET_SESSION_ID` functions to create one call.

Syntax

```
PROCEDURE DEFINE_USER_SESSION(
    p_user         IN    VARCHAR2)
    p_session_id   IN    NUMBER);
```

GET_COOKIE_PROPS Procedure

This procedure obtains the properties of the session cookie used in the current authentication scheme for the specified application. These properties can be viewed directly in the Application Builder by viewing the authentication scheme attributes.

Syntax

```
HTMLDB_CUSTOM_AUTH.GET_COOKIE_PROPS(
 p_app_id                       IN  NUMBER,
 p_cookie_name                  OUT VARCHAR2,
 p_cookie_path                  OUT VARCHAR2,
 p_cookie_domain                OUT VARCHAR2);
```

Parameters

Table 15–73 describes the parameters available in the `GET_COOKIE_PROPS` procedure.

Table 15–73 GET_COOKIE_PROPS Parameters

Parameter	Description
`p_app_id`	An application ID in the current workspace.
`p_cookie_name`	The cookie name.
`p_cookie_path`	The cookie path.
`p_cookie_domain`	The cookie domain.

Example

```
DECLARE
    l_cookie_name   varchar2(256);
    l_cookie_path   varchar2(256);
    l_cookie_domain varchar2(256);
BEGIN
HTMLDB_CUSTOM_AUTH.GET_COOKIE_PROPS (
    p _cookie_name   => l_cookie_name,
    p _cookie_path   => l_cookie_path,
    p _cookie_domain => l_cookie_domain);
END;
```

GET_LDAP_PROPS Procedure

This procedure obtains the LDAP attributes of the current authentication scheme for the current application. These properties can be viewed directly in Application Builder by viewing the authentication scheme attributes.

Syntax

```
HTMLDB_CUSTOM_AUTH.GET_LDAP_PROPS(
 p_ldap_host                OUT VARCHAR2,
 p_ldap_port                OUT NUMBER,
 p_ldap_dn                  OUT VARCHAR2,
 p_ldap_edit_function       OUT VARCHAR2);
```

Parameters

Table 15–74 describes the parameters available in the GET_LDAP_PROPS procedure.

Table 15–74 GET_LDAP_PROPS Parameters

Parameter	Description
`p_ldap_host`	LDAP host name.
`p_ldap_port`	LDAP port number.
`p_ldap_dn`	LDAP DN string.
`p_ldap_edit_function`	LDAP edit function name.

Example

```
DECLARE
    l_ldap_host          varchar2(256);
    l_ldap_port          number;
    l_ldap_dn            varchar2(256);
    l_ldap_edit_function varchar2(256);
BEGIN
HTMLDB_CUSTOM_AUTH.GET_LDAP_PROPS (
    p_ldap_host       => l_ldap_host,
    p_ldap_port       => l_ldap_port,
    p_ldap_dn         => l_ldap_dn,'
    p_ldap_edit_function => l_ldap_edit_function);
END;
```

GET_NEXT_SESSION_ID Function

This function generates the next session ID from the Oracle Application Express sequence generator. This function returns a number.

Syntax

```
FUNCTION GET_NEXT_SESSION_ID
RETURN NUMBER;
```

GET_SESSION_ID_FROM_COOKIE Function

This function returns the Oracle Application Express session ID located by the session cookie in the context of a page request in the current browser session.

Syntax

```
HTMLDB_CUSTOM_AUTH.GET_SESSION_ID_FROM_COOKIE;
RETURN NUMBER;
```

Example

```
DECLARE VAL NUMBER;
BEGIN
  VAL := HTMLDB_CUSTOM_AUTH.GET_SESSION_ID_FROM_COOKIE;
END;
```

GET_USERNAME Function

This function returns user name registered with the current Oracle Application Express session in the internal sessions table. This user name is usually the same as the authenticated user running the current page.

Syntax

```
HTMLDB_CUSTOM_AUTH.GET_USERNAME;
RETURN VARCHAR2;
```

Example

```
DECLARE VAL VARCHAR2(256);
BEGIN
  VAL := HTMLDB_CUSTOM_AUTH.GET_USERNAME;
END;
```

GET_SECURITY_GROUP_ID Function

This function returns a number with the value of the security group ID that identifies the workspace of the current user.

Syntax

```
FUNCTION GET_SECURITY_GROUP_ID
RETURN NUMBER;
```

GET_SESSION_ID Function

This function returns `HTMLDB_APPLICATION.G_INSTANCE` global variable. `GET_SESSION_ID` returns a number.

Syntax

```
PROCEDURE GET_SESSION_ID
RETURN NUMBER;
```

GET_USER Function

This function returns the `HTMLDB_APPLICATION.G_USER` global variable (`VARCHAR2`).

Syntax

```
FUNCTION GET_USER
RETURN VARCHAR2;
```

IS_SESSION_VALID Function

This function is a Boolean result obtained from executing the current application's authentication scheme to determine if a valid session exists. This function returns the Boolean result of the authentication scheme's page sentry.

Syntax

```
HTMLDB_CUSTOM_AUTH.IS_SESSION_VALID;
RETURN BOOLEAN;
```

Example

```
DECLARE VAL VARCHAR2(256);
BEGIN
  VAL := HTMLDB_CUSTOM_AUTH.IS_SESSION_VALID;
END;
```

LOGIN Procedure

Also referred to as the "Login API," this procedure performs authentication and session registration.

Syntax

```
HTMLDB_CUSTOM_AUTH.LOGIN(
 p_uname                     IN  VARCHAR2,
 p_password                  IN  VARCHAR2,
 p_session_id                IN  VARCHAR2,
 p_app_page                  IN  VARCHAR2,
 p_entry_point               IN  VARCHAR2,
 p_preserve_case             IN  BOOLEAN);
```

Parameter

Table 15–75 describes the parameters available in the `LOGIN` procedure.

Table 15–75 LOGIN Parameters

Parameter	Description
`p_uname`	Login name of the user.
`p_password`	Clear text user password.
`p_session_id`	Current Oracle Application Express session ID.
`p_app_page`	Current application ID. After login page separated by a colon (:).
`p_entry_point`	Internal use only.
`p_preserve_case`	If true, do not upper `p_uname` during session registration

Example

```
BEGIN
HTMLDB_CUSTOM_AUTH.LOGIN (
    p_uname       => 'SCOTT',
    p_password    => 'secret99',
    p_session_id  => V('APP_SESSION'),
    p_app_page    => :APP_ID||':1');
END;
```

Note: :Do not use bind variable notations for `p_session_id` argument.

LOGOUT Procedure

This procedure effects a logout from the current session by unsetting the session cookie and redirecting to a new location.

Syntax

```
HTMLDB_CUSTOM_AUTH.LOGOUT(
 p_this_app                  IN VARCHAR2,
 p_next_app_page_sess        IN VARCHAR2,
 p_next_url                  IN VARCHAR2);
```

Parameter

Table 15–76 describes the parameters available in the `LOGOUT` procedure.

Table 15–76 LOGOUT Parameters

Parameter	Description
`p_this_app`	Current application ID.
`p_next_app_page_sess`	Application and page ID to redirect to. Separate multiple pages using a colon (:) and optionally followed by a colon (:) and the session ID (if control over the session ID is desired).
`p_next_url`	URL to redirect to (use this instead of `p_next_app_page_sess`).

Example

```
BEGIN
HTMLDB_CUSTOM_AUTH.LOGOUT (
    p_this_app           => '1000',
    p_next_app_page_sess  => '1000:99');
END;
```

POST_LOGIN Procedure

This procedure performs session registration, assuming the authentication step has been completed. It can be called only from within an Oracle Application Express application page context.

Syntax

```
HTMLDB_CUSTOM_AUTH.POST_LOGIN(
 p_uname                    IN  VARCHAR2,
 p_session_id               IN  VARCHAR2,
 p_app_page                 IN  VARCHAR2,
 p_preserve_case            IN  BOOLEAN);
```

Parameter

Table 15–77 describes the parameters available in the `POST_LOGIN` procedure.

Table 15–77 POST_LOGIN Parameters

Parameter	Description
`p_uname`	Login name of user.

Table 15–77 (Cont.) POST_LOGIN Parameters

Parameter	Description
`p_session_id`	Current Oracle Application Express session ID.
`p_app_page`	Current application ID and after login page separated by a colon (:).
`p_preserve_case`	If true, do not include `p_uname` in uppercase during session registration.

Example

```
BEGIN
HTMLDB_CUSTOM_AUTH.POST_LOGIN (
    p_uname       => 'SCOTT',
    p_session_id  => V('APP_SESSION'),
    p_app_page    => :APP_ID||':1');
END;
```

SESSION_ID_EXISTS Function

This function returns a Boolean result based on the global package variable containing the current Oracle Application Express session ID. Returns true if the result is a positive number. returns false if the result is a negative number.

Syntax

```
FUNCTION SESSION_ID_EXISTS
RETURN BOOLEAN;
```

Example

```
DECLARE VAL BOOLEAN;
BEGIN
  VAL := HTMLDB_CUSTOM_AUTH.SESSION_ID_EXISTS;
END;
```

SET_USER Procedure

This procedure sets the `HTMLDB_APPLICATION.G_USER` global variable. `SET_USER` requires the parameter `P_USER` (`VARCHAR2`) which defines a user ID.

Syntax

```
PROCEDURE SET_USER(
    p_user   IN    VARCHAR2)
```

SET_SESSION_ID Procedure

This procedure sets `HTMLDB_APPLICATION.G_INSTANCE` global variable. `SET_SESSION_ID` returns a number. This procedure requires the parameter `P_SESSION_ID` (`NUMBER`) which specifies a session ID.

Syntax

```
PROCEDURE SET_SESSION_ID(
    p_session_id    IN    NUMBER)
```

SET_SESSION_ID_TO_NEXT_VALUE Procedure

This procedure combines the operation of `GET_NEXT_SESSION_ID` and `SET_SESSION_ID` in one call.

Syntax

```
PROCEDURE SET_SESSION_ID_TO_NEXT_VALUE;
```

HTMLDB_LDAP

You can use `HTMLDB_LDAP` to perform various operations related to Lightweight Directory Access Protocol (LDAP) authentication.

Topics in this section include:

- AUTHENTICATE Function
- IS_MEMBER Function
- MEMBER_OF Function
- MEMBER_OF2 Function
- GET_USER_ATTRIBUTES Procedure
- GET_ALL_USER_ATTRIBUTES Procedure

AUTHENTICATE Function

The `AUTHENTICATE` function returns a boolean true if the username and password can be used to perform a `SIMPLE_BIND_S` call using the provided search base, host, and port.

Syntax

```
FUNCTION AUTHENTICATE(
    p_username     in VARCHAR2 DEFAULT NULL,
    p_password     in VARCHAR2 DEFAULT NULL,
    p_search_base  in VARCHAR2,
    p_host         in VARCHAR2,
    p_port         in VARCHAR2 DEFAULT 389)
RETURN BOOLEAN;
```

Parameters

Table 15–78 describes the parameters available in the `AUTHENTICATE` function.

Table 15–78 AUTHENTICATE Parameters

Parameter	Description
`p_username`	Login name of the user.
`p_password`	Password for `p_username`.
`p_search_base`	LDAP search base, for example, `dc=users,dc=my,dc=org`.
`p_host`	LDAP server host name.
`p_port`	LDAP server port number.

IS_MEMBER Function

The IS_MEMBER function returns a boolean true if the user named by p_username (with password if required) is a member of the group specified by the p_group and p_group_base parameters using the provided auth base, host, and port.

Syntax

```
FUNCTION IS_MEMBER(
    p_username     in VARCHAR2 DEFAULT NULL,
    p_pass         in VARCHAR2 DEFAULT NULL,
    p_auth_base    in VARCHAR2,
    p_host         in VARCHAR2,
    p_port         in VARCHAR2 DEFAULT 389,
    p_group        in VARCHAR2,
    p_group_base   in VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 15–79 describes the parameters available in the IS_MEMBER function.

Table 15–79 IS_MEMBER Parameters

Parameter	Description
p_username	Login name of the user.
p_pass	Password for p_username.
p_auth_base	LDAP search base, for example, dc=users,dc=my,dc=org.
p_host	LDAP server host name.
p_port	LDAP server port number.
p_group	Name of the group to be search for membership.
p_group_base	The base from which the search should be started.

MEMBER_OF Function

The MEMBER_OF function returns an array of groups the username designated by p_username (with password if required) belongs to, using the provided auth base, host, and port.

Syntax

```
FUNCTION MEMBER_OF(
    p_username     in VARCHAR2 DEFAULT NULL,
    p_pass         in VARCHAR2 DEFAULT NULL,
    p_auth_base    in VARCHAR2,
    p_host         in VARCHAR2,
    p_port         in VARCHAR2 DEFAULT 389)
RETURN wwv_flow_global.vc_arr2;
```

Parameters

Table 15–80 describes the parameters available in the MEMBER_OF function.

Table 15–80 MEMBER_OF Parameters

Parameter	Description
`p_username`	Login name of the user.
`p_pass`	Password for `p_username`.
`p_auth_base`	LDAP search base, for example, `dc=users,dc=my,dc=org`.
`p_host`	LDAP server host name.
`p_port`	LDAP server port number.

MEMBER_OF2 Function

The `MEMBER_OF2` function returns an `VARCHAR2` list of groups the username designated by `p_username` (with password if required) belongs to, using the provided auth base, host, and port.

Syntax

```
FUNCTION MEMBER_OF2(
    p_username     in VARCHAR2 DEFAULT NULL,
    p_pass         in VARCHAR2 DEFAULT NULL,
    p_auth_base    in VARCHAR2,
    p_host         in VARCHAR2,
    p_port         in VARCHAR2 DEFAULT 389)
RETURN VARCHAR2;
```

Parameters

Table 15–81 describes the parameters available in the `MEMBER_OF2` function.

Table 15–81 MEMBER_OF2 Parameters

Parameter	Description
`p_username`	Login name of the user.
`p_pass`	Password for `p_username`.
`p_auth_base`	LDAP search base, for example, `dc=users,dc=my,dc=org`.
`p_host`	LDAP server host name.
`p_port`	LDAP server port number.

GET_USER_ATTRIBUTES Procedure

The `GET_USER_ATTRIBUTES` procedure returns an OUT array of user_attribute values for the username designated by `p_username` (with password if required) corresponding to the attribute names passed in `p_attributes`, using the provided auth base, host, and port.

Syntax

```
PROCEDURE GET_USER_ATTRIBUTES(
    p_username         in VARCHAR2 DEFAULT NULL,
    p_pass             in VARCHAR2 DEFAULT NULL,
    p_auth_base        in VARCHAR2,
    p_host             in VARCHAR2,
    p_port             in VARCHAR2 DEFAULT 389,
    p_attributes       in  wwv_flow_global.vc_arr2,
    p_attribute_values  out wwv_flow_global.vc_arr2);
```

Parameters

Table 15–82 describes the parameters available in the GET_USER_ATTRIBUTES procedure.

Table 15–82 GET_USER_ATTRIBUTES Parameters

Parameter	Description
p_username	Login name of the user.
p_pass	Password for p_username.
p_auth_base	LDAP search base, for example, dc=users,dc=my,dc=org.
p_host	LDAP server host name.
p_port	LDAP server port number.
p_attributes	An array of attribute names for which values are to be returned.
p_attribute_values	An array of values returned for each corresponding attribute name in p_attributes.

GET_ALL_USER_ATTRIBUTES Procedure

The GET_ALL_USER_ATTRIBUTES procedure returns two OUT arrays of user_attribute names and values for the username designated by p_username (with password if required) using the provided auth base, host, and port.

Syntax

```
PROCEDURE GET_ALL_USER_ATTRIBUTES(
    p_username          in VARCHAR2 DEFAULT NULL,
    p_pass              in VARCHAR2 DEFAULT NULL,
    p_auth_base         in VARCHAR2,
    p_host              in VARCHAR2,
    p_port              in VARCHAR2 DEFAULT 389,
    p_attributes        out  wwv_flow_global.vc_arr2,
    p_attribute_values  out wwv_flow_global.vc_arr2);
```

Parameters

Table 15–83 describes the parameters available in the GET_ALL_USER_ATTRIBUTES procedure.

Table 15–83 GET_ALL_USER_ATTRIBUTES Parameters

Parameter	Description
p_username	Login name of the user.
p_pass	Password for p_username.
p_auth_base	LDAP search base, for example, dc=users,dc=my,dc=org.
p_host	LDAP server host name.
p_port	LDAP server port number.
p_attributes	An array of attribute names returned.
p_attribute_values	An array of values returned for each corresponding attribute name returned in p_attributes.

Part III

Database Tools

Part III explains how to use view and manage database objects from a Web browser.

Part III contains the following chapters:

16 Managing Database Objects with Object Browser

Object Browser enables developers to browse, create, and edit objects in an Oracle Database Express Edition (Oracle Database XE) database.

This section contains the following topics:

See Also:

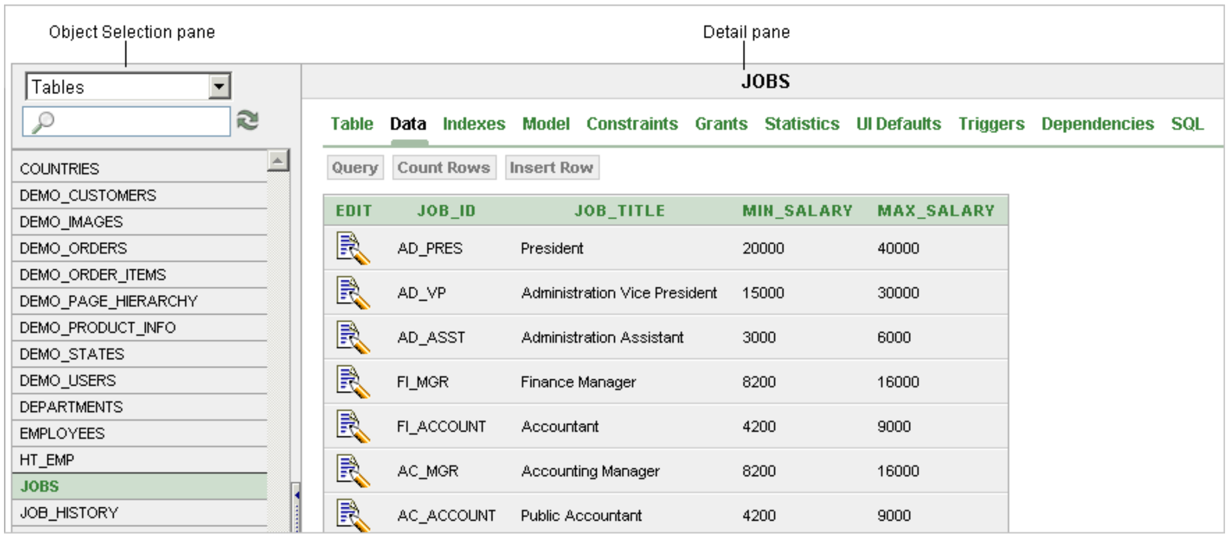
- "Managing Database Objects" in *Oracle Database Express Edition 2 Day Developer Guide* and *Oracle Database SQL Reference*

About Object Browser

The Object Browser page is divided into two sections:

- **Object Selection pane** displays on the left side of the Object Browser page and lists database objects of a selected type within the current schema. You can further narrow the results by filtering on the object name.

- **Detail pane** displays to the right of the page and displays detailed information about the selected object. To view object details, select an object in the Object Selection pane. Click the tabs at the top of the Detail pane to view additional details about the current object. To edit an object, click the appropriate button.

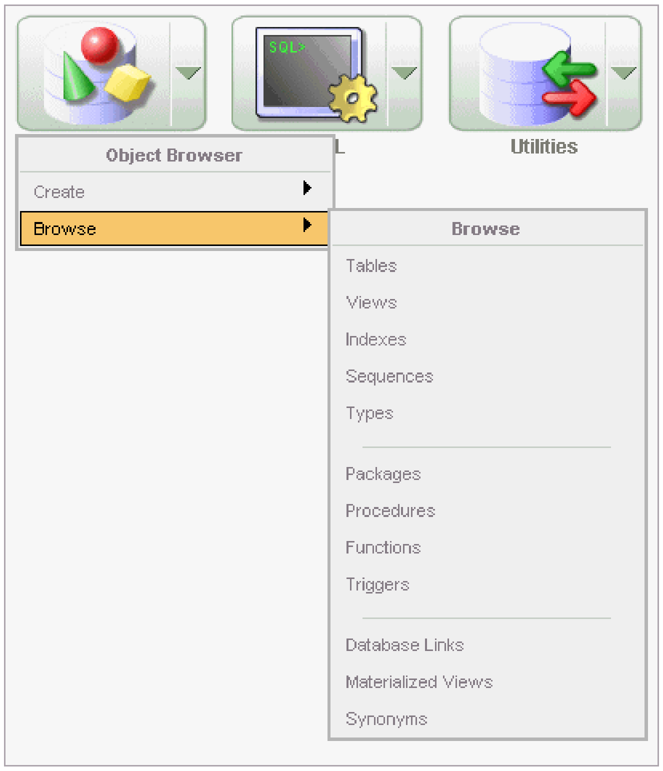


EDIT	JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
	AD_PRES	President	20000	40000
	AD_VP	Administration Vice President	15000	30000
	AD_ASST	Administration Assistant	3000	6000
	FI_MGR	Finance Manager	8200	16000
	FI_ACCOUNT	Accountant	4200	9000
	AC_MGR	Accounting Manager	8200	16000
	AC_ACCOUNT	Public Accountant	4200	9000

Accessing Object Browser

To access Object Browser:

1. Log in to the Database Home Page.
2. To view Object Browser you can either:
   - Click the **Object Browser** icon.
   - Click the down arrow on the right side of the icon to view a drop down menu. Then, select the appropriate menu option.



Note: For the purposes of consistency, this document uses the primary navigation path (or drill-down approach) when explaining navigation.

Searching For and Browsing Database Objects

The Object Selection pane displays on the left side of the Object Browser page and lists database objects by type with the current schema. You can filter the view by selecting an object type or entering a case insensitive search term.

Topics in this section include:

- Searching For and Selecting Database Objects
- Hiding the Object Selection Pane

Searching For and Selecting Database Objects

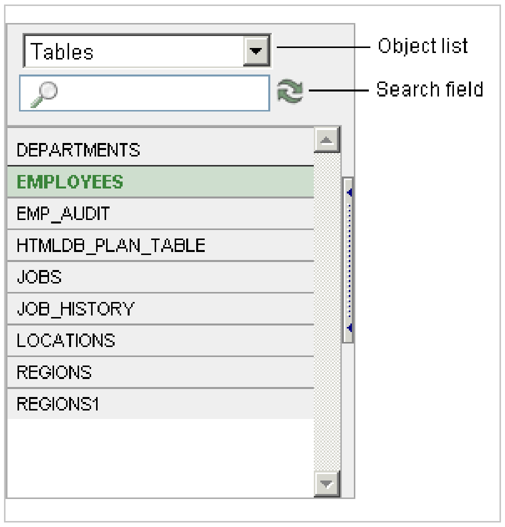
To search for a database object in the Object Selection pane:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. Select an object type from the Object list.

   The list of objects that appears depends upon the available objects in the current schema. Note that any object having a red bar adjacent to it is invalid.

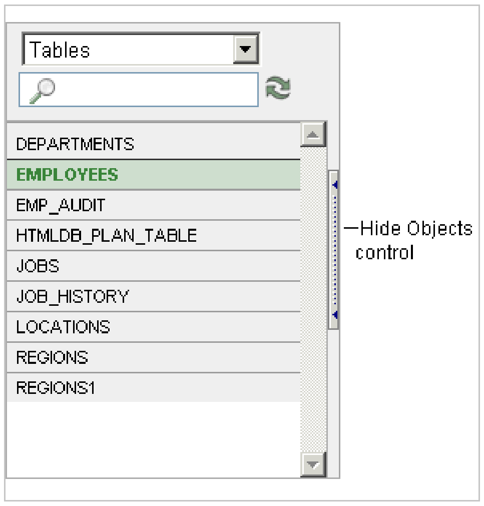


3. To search for an object name, enter a case insensitive search term in the search field.
4. To view all objects, leave the search field blank.

Once you locate the database object you want to view, simply select it. The selected object displays in the Detail pane. If no object is selected, the Detail pane is blank.

Hiding the Object Selection Pane

You can hide the Object Selection pane by selecting the **Hide Objects** control. This control displays on the right side of the Object Selection pane. If the Object Selection pane appears, selecting this control hides it. Similarly, if the Object Selection pane is hidden, selecting this control causes the pane to reappear.



About Creating New Database Objects

You can create new database objects using the Create Database Object Wizard. Once you select an object, a set of tabs and buttons appears at the top of the Detail pane. Use the tabs to view different aspects of the current items (for example, a tables indexes). Use the buttons to modify the current object.

To create a new object:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. Click **Create**, located in the upper right corner of the Detail pane.
3. From the list of object types, select the type of object you want to create.
4. Follow the on-screen instructions.

Managing Tables

A table is a unit of data storage in an Oracle database, containing rows and columns. When you view a table in Object Browser, a table description appears that describes each column in the table.

See Also:

- "Managing Tables" in *Oracle Database Express Edition 2 Day Developer Guide*
- *Oracle Database Administrator's Guide* for information on managing tables.
- *Oracle Database Concepts* for conceptual information on tables types.
- *Oracle Database SQL Reference* for the syntax required to create and alter tables.

Topics in this section include:

- Creating a Table
- Browsing a Table
- Editing a Table
- Dropping a Table

Creating a Table

To create a new table:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. Click **Create**.
3. From the list of object types, select **Table**.
4. Enter a table name.

   Table names must conform to Oracle naming conventions and not contain spaces or start with a number or underscore.

5. To have the final table name match the case entered in the Table Name field, click **Preserve Case**.
6. Enter details for each column. For each column:

   a. Enter the column name.

b. Select the column type. Available types include NUMBER, VARCHAR2, DATE, TIMESTAMP, CHAR, CLOB, BLOB, NVARCHAR2, BINARY_FLOAT, and BINARY_DOUBLE

c. Enter the following additional information as appropriate:

   – Precision

   – Scale

d. To specify a column should not be NULL, select the check box in the **Not Null** column.

   To change the order of previously entered columns, click the **Up** and **Down** arrows in the Move column. To add additional columns, click **Add Column**.

e. Click **Next**.

Next, define the primary key for this table (optional). A primary key is a single field or combination of fields that uniquely identifies a record.

7. For Primary Key, select one of the following and click **Next**:

   - **No Primary Key** - No primary key is created.
   - **Populated from a new sequence** - Creates a primary key and creates a trigger and a new sequence. The new sequence is used in the trigger to populated the selected primary key column. The primary key can only be a single column.
   - **Populated from an existing sequence** - Creates a primary key and creates a trigger. The selected sequence is used in the trigger to populate the selected primary key column. The primary key can only be a single column.
   - **Not populated** - Defines a primary key but does not have the value automatically populated with a sequence within a trigger. You can also select this option to define a composite primary key (that is, a primary key made up of more than one column).

   Next, add foreign keys (optional). A foreign key establishes a relationship between a column (or columns) in one table and a primary or unique key in another table.

8. To add a foreign key:

   a. Name - Enter a name of the foreign key constraint that you are defining.

   b. Select Key Column(s) - Select the columns that are part of the foreign key. Once selected, click the **Add** icon to move them to Key Column(s).

   c. References Table - Select the table which will be referenced by this foreign key. Then, select the columns to be referenced by this foreign key. Once selected, click the **Add** icon to move the selected columns to Referenced Column(s).

   d. Select one of the following:

      – **Disallow Delete** - Blocks the delete of rows from the referenced table when there are dependent rows in this table.

      – **Cascade Delete** - Deletes the dependent rows from this table when the corresponding parent table row is deleted.

      – **Set to Null on Delete** - Sets the foreign key column values in this table to null when the corresponding parent table row is deleted.

   e. Click **Add**.

   f. Click **Next**.

Next, add a constraint (optional). You can create multiple constraints, but you must add each constraint separately.

9. To add a constraint:

   a. Specify the type of constraint (Check or Unique).

      A **check constraint** is a validation check on one or more columns within the table. No records can be inserted or updated in a table which violates an enabled check constraint. A **unique constraint** designates a column or a combination of columns as a unique key. To satisfy a unique constraint, no two rows in the table can have the same values for the specified columns.

   b. Enter the constraint in the field provided. For unique constraints, select the column(s) that are to be unique. For check constraints, enter the expression that should be checked such as, `flag in ('Y','N')`.

   c. Click **Add**.

10. Click **Finish**.

    A confirmation page appears. To view the SQL used to create the table, click **SQL Syntax.**

11. Click **Create**.

    Note that you do not need to follow the steps for creating a table in the order described in the previous procedure. Instead of navigating through the wizard by clicking the Next and Previous button, you can also access a specific step by selecting it in the progress indicator on the left side of the page.

    See Also: "Overview of Tables" in *Oracle Database Concepts* for information about tables

Browsing a Table

When you view a table in Object Browser, the table description appears. While viewing this description, you can add a column, modify a column, rename a column, drop a column, rename the table, copy the table, drop the table, truncate the table, or create a lookup table based upon a column in the current table. Additionally, you have access other reports that offer related information including the table data, indexes, data model, constraints, grants, statistics, user interface defaults, triggers, dependencies, and SQL to produce the selected table.

To view a table description:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. From the Object list, select **Tables**.

3. From the Object Selection pane, select a table.

   The table description appears.

Summary of Available Views

Click the tabs at the top of the page to view different reports about the table. Table 16–1 describes all available views.

Table 16–1 Available Views for Tables

View	Description
Table	Displays details of the first 31 columns including the column name, data type, nullable status, default value, and primary key. While viewing table details you can add, modify, delete, or rename a column. Additionally, you can drop, rename, copy, or truncate the table as long as the referencing table has no records as well as create a lookup table. **See Also:** "Editing a Table" on page 16-9
Data	Displays a report of the data in the current table. Actions you can perform include: ■ **Query** - Enables you to sort by column. To restrict specific rows, enter a condition in the Column Condition field. Use the percent sign (%) for wildcards. From Order by, select the columns you want to review and click **Query**. ■ **Count Rows** - Displays a report of the number of rows in the current table. ■ **Insert Row** - Enables you to insert a new row into the table. ■ **Download** - Exports all data in the table to a spreadsheet. Click the download link at the bottom of the page to export all data in the selected table.
Indexes	Displays indexes associated with this table. Actions you can perform include **Create** and **Drop**. **See Also:** "Managing Indexes" on page 16-13
Model	Displays a graphical representation of the selected table along with all related tables. Related tables are those that reference the current table in a foreign key and those tables referenced by foreign keys within the current table. You can position the cursor over an underlined table name to view the relationship between that table and the current table. Click an underlined table name to view the model of the related table.
Constraints	Displays a list of constraints for the current table. Actions you can perform include **Create**, **Drop**, **Enable**, and **Disable**.
Grants	Displays a list of grants on the current table, including the grantee, the privilege, and grant options. Actions you can perform in this view include **Grant** and **Revoke**.
Statistics	Displays collected statistics about the current table, including the number of rows and blocks, the average row length, sample size, when the data was last analyzed, and the compression status (enabled or disabled). Click **Analyze** to access the Analyze Table Wizard.
UI Defaults	Displays user interface defaults for forms and reports. User interface defaults enable developers to assign default user interface properties to a table, column, or view within a specified schema. Click **Edit** to edit defined user interface defaults. Click **Create** to initialize user interface defaults for tables that do not currently have user interface defaults defined. **See Also:** "Managing User Interface Defaults" on page 9-1

Table 16–1 (Cont.) Available Views for Tables

View	Description
Triggers	Displays a list of triggers associated with the current table. Actions you can perform include **Create**, **Drop**, **Enable**, and **Disable**. To view trigger details, click the trigger name. **See Also**: "Managing Triggers" on page 16-26
Dependencies	Displays report showing objects referenced by this table, objects this table references, and synonyms for this table.
SQL	Displays the SQL necessary to re-create this table, including keys, indexes, triggers and table definition.

Editing a Table

While viewing a table description, you can edit it by clicking the buttons above the table description.

To edit a table:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Tables**.
3. From the Object Selection pane, select a table.

   The table description appears.
4. Click the appropriate button described in Table 16–2.

Table 16–2 Edit Table Buttons

Button	Description
Add Column	Adds a new column to the table. Enter a column name and select a type. Depending upon the column type, specify whether the column requires a value as well as the column length, precision, and scale.
Modify Column	Modifies the selected column.
Rename Column	Renames the selected column.
Drop Column	Drops the selected column.
Rename	Renames the selected table.
Copy	Copies the selected table.
Drop	Drops the selected table. **See Also:** "Using the Recycle Bin to View and Restore Dropped Objects" on page 20-8
Truncate	Removes all rows from the selected table. Truncating a table can be more efficient than dropping and re-creating a table. Dropping and re-creating a table may invalidate dependent objects, requiring you to regrant object privileges or re-create indexes, integrity constraints, and triggers.
Create Lookup Table	Creates a lookup table based on the column you select. That column becomes a foreign key to the lookup table.

Dropping a Table

To drop a table:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. From the Object list, select **Tables**.
3. From the Object Selection pane, select a table.

   The table description appears.

4. Click **Drop**.

> **See Also:** "Using the Recycle Bin to View and Restore Dropped Objects" on page 20-8

Managing Views

A view is a logical representation of another table or combination of tables. A view derives its data from the tables on which it is based. These tables are called **base tables**. Base tables might in turn be actual tables or might be views themselves. All operations performed on a view actually affect the base table of the view. You can use views in almost the same way as tables. You can query, update, insert into, and delete from views, just as you can standard tables.

Topics in this section include:

- Creating a View
- Browsing a View
- Editing a View
- Compiling a View
- Dropping a View

> **See Also:** "Managing Views" in *Oracle Database Express Edition 2 Day Developer Guide* and *Oracle Database Administrator's Guide*

Creating a View

To create a new view:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. Click **Create**.
3. From the list of object types, select **View**.
4. Define the view:
   - **View Name** - Enter a name for the View.
   - **Query** - Specify a query to define the view.

     To access Query Builder or SQL Command Processor, click the appropriate link at the bottom of the page. The selected tool appears in a pop-up window. Once you create the appropriate SQL, click **Return** to automatically close the popup window and return to the wizard with the SQL.

5. Click **Next**.

A confirmation page appears. To view the SQL used to create the view, click **SQL**.

6. Click **Create**.

See Also: "Building Queries with Query Builder" on page 17-1 and "Using SQL Commands" on page 19-1

Browsing a View

When you access a view in Object Browser, the Detail pane displays a report listing the columns in that view.

To browse a view:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. From the Object list, select **Views**.

3. From the Object Selection pane, select a view.

   The view definition appears displaying the appropriate columns.

Summary of Available Views

Click the tabs at the top of the page to view different reports. Table 16–3 describes all available views.

Table 16–3 Available Views for Views

View	Description
View	(Default) Displays the columns in the current view. Actions you can perform include: ■ Compile ■ Drop **See Also:** "Editing a View" on page 16-12, "Compiling a View" on page 16-12, and "Dropping a View" on page 16-12
Data	Displays a report of the data in the columns in the view. Actions you can perform include: ■ **Query** - Enables you to sort by column. To restrict specific rows, enter a condition in the Column Condition field. Use the percent sign (%) for wildcards. From Order by, select the columns you want to review and click **Query**. ■ **Count Rows** - Enables you to insert a new row into the table. ■ **Insert Row** - Enables you to insert a new row into the table.
Grants	Displays a list of grants associated with the columns in the view. Grant details include grantee, privilege, and grant options. Actions you can perform include **Grant** and **Revoke**.
UI Defaults	Displays user interface defaults for forms and reports. User interface defaults enable developers to assign default user interface properties to a table, column, or view within a specified schema. Click **Edit** to edit existing user interface defaults. Click **Create** to initialize user interface defaults for views that do not currently have user interface defaults defined. **See Also:** "Managing User Interface Defaults" on page 9-1

Table 16–3 (Cont.) Available Views for Views

View	Description
Dependencies	Displays a report showing objects referenced by this view, objects this view references, and synonyms for this view.
SQL	Displays the SQL necessary to re-create this view.

Editing a View

When you edit a view you can edit the code manually, perform a search and replace, and compile the view. Additionally, you can save the view as a file or drop it.

Editing a View Manually

To edit a view manually:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Views**.
3. From the Object Selection pane, select a view.
4. Select the Code tab.
5. Click **Edit** to activate manual edit mode.

   If you edit and make changes to a view, you need to compile. See "Compiling a View" on page 16-12.

Note: You can expand the Edit pane by clicking the Full Screen icon in the upper right of the pane, beneath the Create button.

Using Find and Replace

Click **Find** to perform a basic search and replace.

Downloading a View

Click **Download** to save the current view as a file.

Compiling a View

If you edit and make changes to a view, you need to compile to save your changes. Note that there is no save function since this is just a view of the object within the database.

Click **Compile** to re-create the current view.

Dropping a View

To drop a view:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Views**.
3. From the Object Selection pane, select a view.

4. Select the **View** tab or the **Code** tab.
5. Click **Drop** to delete the current view.

Managing Indexes

An index is an optional structure associated with tables and clusters. You can create indexes on one or more columns of a table to speed access to data on that table.

When you view an index in Object Browser, the Detail pane displays a report containing the index name, index type, table owner, table type, and a listing of the indexed columns.

Topics in this section include:

- Creating an Index
- Browsing an Index
- Dropping an Index

See Also: "Managing Indexes" in *Oracle Database Express Edition 2 Day Developer Guide*

Creating an Index

To create an index:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. Click **Create**.
3. From the list of object types, select **Index**.
4. Select a table and select the type of index you want to create. Available index types include:
   - **Normal** - Indexes one or more scalar typed object attributes of a table
   - **Text** - Creates a text index (Oracle Text)
5. Click **Next**.
6. Create the index definition. Specify an index name, select one or more columns to be indexed, and click **Next**.

   A confirmation page appears. To view the SQL used to create the index, click **SQL.**
7. Click **Finish**.

Browsing an Index

To browse an index:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Indexes**.
3. From the Object Selection pane, select an index.

   The index appears displaying the index name, type, table owner, and table type as well as a listing of indexed columns.

Summary of Available Views

Click the tabs at the top of the page to view different reports about the index. Table 16–4 describes all available views.

Table 16–4 Available Views for Indexes

View	Description
Object Details	(Default) Displays the index name, index type, table owner, and table type as well as a listing of the indexed columns. Actions you can perform while viewing Object Details include: ■ Disable - Disables the current index ■ Drop - Drops the current index.x ■ Rebuild - Rebuilds the current index
Statistics	Displays collected statistics about the current view, including the number of rows, sample size, when the data was last analyzed, and the compression status (enabled or disabled). Click **Analyze** to refresh the displayed statistics.
SQL	Displays the SQL necessary to re-create this index.

Dropping an Index

To drop an index:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Indexes**.
3. From the Object Selection pane, select an index.
4. Under Object Details, click the **Drop** tab.

Managing Sequences

A sequence generates a serial list of unique numbers for numeric columns of a database table. Database sequences are generally used to populate table primary keys.

Topics in this section include:

- Creating a Sequence
- Browsing a Sequence
- Dropping a Sequence

See Also: "Managing Sequences" in *Oracle Database Express Edition 2 Day Developer Guide*

Creating a Sequence

To create a sequence:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. Click **Create**.
3. From the list of object types, select **Sequence**.

4. Define the sequence, specify a sequence name, and click **Next**.

   A confirmation page appears. To view the SQL used to create the sequence, click **Show SQL.**

5. Click **Create**.

Browsing a Sequence

To browse a sequence:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. From the Object list, select **Sequences**.
3. From the Object Selection pane, select a sequence.

   The Object Details view appears.

Summary of Available Views

Click the tabs at the top of the page to view different reports about the sequence. Table 16–5 describes all available views.

Table 16–5 Available Views for Sequences

View	Description
Object Details	(Default) Displays details about the current sequence. Actions you can perform in this view include **Alter** and **Drop**.
Grant	Displays a list of grants associated with the sequence. Grant details include grantee, privilege, and grant options. Actions you can perform include **Grant** and **Revoke**.
Dependencies	Displays a list of objects that use (or depend) upon this sequence.
SQL	Displays the SQL necessary to re-create this sequence.

Dropping a Sequence

To drop a sequence:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. From the Object list, select **Sequences**.
3. From the Object Selection pane, select a sequence.

   The Object Details view appears.

4. Click **Drop**.

Managing Types

A type is a user-specified object or collection definition. Oracle Database Express Edition (Oracle Database XE) currently only supports collection definitions. There are two categories of Oracle collections (SQL collections):

- Variable-length arrays (`VARRAY` types)
- Nested tables (`TABLE` types)

`VARRAY` types are used for one-dimensional arrays, while nested table types are used for single-column tables within an outer table.

Topics in this section include:

- Creating a Type
- Browsing a Type
- Dropping a Type

> **See Also:** *Oracle Database Concepts* and *Oracle Database PL/SQL User's Guide and Reference* for information about collection types

Creating a Type

To create a collection type:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. Click **Create**.
3. From the list of object types, select **Type**.
4. Specify a name and click **Next**.
5. Select a type, data type, limit, and click **Next**.

   A confirmation page appears. To view the SQL used to create the collection type, click **Show SQL**.
6. Click **Finish**.

Browsing a Type

To browse a collection type:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Type**.
3. From the Object Selection pane, select a type.

   The Object Details view appears.

Summary of Available Views

Click the tabs at the top of the page to view different reports. Table 16–6 describes all available views.

Table 16–6 Available Views for Types

View	Description
Object Details	(Default) Displays details about the selected type. To drop a type, click **Drop**.
Synonyms	Displays a list of synonyms for the current type.
Grants	Displays a list of grants associated with the type. Grant details include grantee, privilege, and grant options. Actions you can perform include **Grant** and **Revoke**.
SQL	Displays the SQL necessary to re-create this type.

Dropping a Type

To drop a collection type:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Type**.
3. From the Object Selection pane, select a type.

   The Object Details view appears.
4. Click **Drop**.

Managing Packages

A package is a database object that groups logically related PL/SQL types, items, functions and procedures. Packages usually have two parts, a specification and a body. The **specification** is the interface to your application. The **body** implements the specification.

Topics in this section include:

- Creating a Package
- Viewing a Package
- Editing a Package
- Compiling a Package
- Downloading a Package
- Dropping a Package

See Also:

- "Managing Packages" in *Oracle Database Express Edition 2 Day Developer Guide*
- "Using PL/SQL Packages" in *Oracle Database PL/SQL User's Guide and Reference* for additional information on PL/SQL packages.
- "Using PL/SQL Subprograms" in *Oracle Database PL/SQL User's Guide and Reference* for information on PL/SQL subprograms

Creating a Package

To create a package:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. Click **Create**.
3. From the list of object types, select **Package**.
4. Select the type of package you want to create:
   - Specification
   - Body
   - Package with methods on database tables

5. If you select **Specification**:
   a. Enter a name and click **Next**.

      The wizard creates a dummy package specification and displays it for editing.
   b. Edit the specification and click **Finish**.
6. If you select **Body**:
   a. Select the package you want to create the body for and click **Next**

      The wizard creates a package body with stubbed out calls identified in the specification and displays it for editing.
   b. Edit the package body and click **Finish**.
7. If you select **Package with methods on database tables**:
   a. Enter a name and click **Next**.
   b. Select up to ten tables and click **Next**.

      The wizard creates a specification and body with insert, update, delete, and GET APIs for the selected tables. Note that you have the option to show or download the specification or body.
   c. Click **Finish**.

Viewing a Package

When you access a package in Object Browser the specification appears.

To view a specification:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Packages**.
3. From the Object Selection pane, select a package.

   The Specification appears. You can copy the code in this view for use in other tools.

Summary of Available Views

Click the tabs at the top of the page to view different reports about the package. Table 16–7 describes all available views.

Table 16–7 Available Views for Packages

View	Description
Specification	(Default) Displays the package specification. This define the interface to your application. Actions you can perform include: ■ **Edit** ■ **Compile** ■ **Download** ■ **Drop** ■ **Find**

Table 16–7 (Cont.) Available Views for Packages

View	Description
Body	Displays the package body, if one exists, for the selected package. Actions you can perform include: ■ **Edit** ■ **Compile** ■ **Download** ■ **Drop** ■ **Find**
Dependencies	Displays objects that use (or depend on) on the current package and objects the package depends on.
Errors	Displays errors related to the current package.
Grants	Lists details of grants for the current package, including privilege, grantee, grantable, grantor, and object name.

Editing a Package

When you edit a package, you can edit the code manually, perform a search and replace, and compile the package.

Editing a Package Manually

To edit a package manually:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Packages**.
3. From the Object Selection pane, select a package.

   The Specification appears. You can copy the code in this view for use in other tools. Note you can edit both the specification and the body fromObject Browser.
4. Click **Edit** to activate edit mode.
5. Click **Find** to perform a basic search and replace.

Note: You can expand the Edit pane by clicking the Full Screen icon in the upper right of the pane, beneath the Create button.

Compiling a Package

If you edit and make changes to a package, you need to compile in order to save your changes. There is no save function because this is just a view of the object within the database.

To compile a package:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Packages**.
3. From the Object Selection pane, select a package.

The Specification appears.

4. Click **Compile** to compile the current package.

   Compiling re-creates the object in the database. If the compile fails, an error message display above the code.

Downloading a Package

To download a package:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. From the Object list, select **Packages**.
3. From the Object Selection pane, select a package.

   The Specification appears.

4. Click **Download** to save the current package as a file.

Dropping a Package

To drop a package while viewing the Specification:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. From the Object list, select **Packages**.
3. From the Object Selection pane, select a package.

   The Specification appears.

4. Click **Drop** to delete the current package.

To drop a package while viewing the Body:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. From the Object list, select **Packages**.
3. From the Object Selection pane, select a package.

   The Specification appears.

4. Click the **Body** tab.
5. Click **Drop**.

Managing Procedures

A procedure is a subprogram that performs a specific action. You can use Object Browser to view, create, edit, download, and drop procedures.

Topics in this section include:

- Creating a Procedure
- Browsing a Procedure
- Editing a Procedure

- Compiling a Procedure
- Downloading a Procedure
- Dropping a Procedure

> **See Also:** "Using Subprograms and Packages" in *Oracle Database Express Edition 2 Day Developer Guide* and "Understanding PL/SQL Procedures" in *Oracle Database PL/SQL User's Guide and Reference*

Creating a Procedure

To create a procedure:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. Click **Create**.
3. From the list of object types, select **Procedures**.
4. Enter a procedure name and click **Next**.
5. Define the arguments by specifying the following information (optional):
   - Argument Name
   - In/Out (the parameter mode)
   - Argument Type (datatype)
   - Default (value)

   To add additional arguments, click **Add Argument**.
6. Click **Next**.
7. Enter PL/SQL block you want to use as the procedure body and click **Next**.

   To view the previously defined arguments, click **Defined Arguments**.

   A confirmation page appears. To view the SQL used to create the procedure, click **Show SQL**.
8. Click **Finish**.

Browsing a Procedure

To browse a procedure:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Procedures**.
3. From the Object Selection pane, select a procedure.

   The Code view appears, displaying the source code for the procedure. You can copy the code in this view for use in other tools.

Summary of Available Views

Click the tabs at the top of the page to view different reports about the procedure. Table 16–8 describes all available views.

Table 16–8 Available Views for Procedures

View	Description
Code	(Default) Displays the source code for the procedure. You can copy the code in this view for use in other tools. Actions you can perform in this view include: ■ **Edit** ■ **Compile** ■ **Download** ■ **Drop** ■ **Find** **See Also:** "Editing a Procedure" on page 16-22, "Compiling a Procedure" on page 16-22, "Downloading a Procedure" on page 16-23, and "Dropping a Procedure" on page 16-23
Dependencies	Displays objects that use (or depend) on the current procedure and objects the procedure depends on.
Errors	Lists errors related to the current procedure.
Grants	Lists details of grants for the current procedure, including privilege, grantee, grantable, grantor, and object name.

Editing a Procedure

When you edit a procedure you can edit the code manually or perform a search and replace.

Editing a Procedure Manually

To edit a procedure manually:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. From the Object list, select **Procedures**.
3. From the Object Selection pane, select a procedure.

   The Code view appears. Be default, you can copy the code in this view for use in other tools.

4. Click **Edit** to activate edit mode.
5. Click **Find** to perform a basic search and replace.

Note: You can expand the Edit pane by clicking the Full Screen icon in the upper right of the pane, beneath the Create button.

Compiling a Procedure

If you edit and make changes to a procedure, you need to compile in order to save your changes. There is no save function because this is just a view of the object within the database.

To compile a procedure:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. From the Object list, select **Procedures**.
3. From the Object Selection pane, select a procedure.
4. Click **Compile** to compile the current procedure.

   Compiling re-creates the object in the database. If the compile fails, an error message display above the code.

Downloading a Procedure

To download a procedure:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Procedures**.
3. From the Object Selection pane, select a procedure.
4. Click **Download** to save the current procedure as a file.

Dropping a Procedure

To drop a procedure:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Procedures**.
3. From the Object Selection pane, select a procedure.
4. Click **Drop** to delete the current procedure.

Managing Functions

A function is a subprogram that can take parameters and return a single value.

Topics in this section include:

- Creating a Function
- Browsing a Function
- Editing a Function
- Compiling a Function
- Downloading a Function
- Dropping a Function

See Also: *Oracle Database SQL Reference* for information about PL/SQL functions and "Understanding PL/SQL Functions" in *Oracle Database PL/SQL User's Guide and Reference*

Creating a Function

To create a function:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. Click **Create**.
3. From the list of object types, select **Functions**.
4. Enter a function name, specify the return datatype, and click **Next**.
5. Define the arguments and click **Next** (optional):
   - Argument Name
   - Argument Type (datatype)
   - Default (value)

   To add additional arguments, click **Add Argument**.
6. Enter P/LSQL block you want to use as the function body and click **Next**.

   To link to SQL Commands, click **Command Processor.** To view the previously defined arguments, click **Defined Arguments**.

   A confirmation page appears. To view the SQL used to create the function, click **Show SQL**.
7. Click **Finish**.

Browsing a Function

To view a function in Object Browser:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Functions**.
3. From the Object Selection pane, select a function.

   The Code view appears. You can copy the code in this view for use in other tools.

Summary of Available Views

Click the tabs at the top of the page to view different reports about the function. Table 16–9 describes all available views.

Table 16–9 Available Views for Functions

View	Description
Code	(Default) Displays the source code for the function. You can copy the code in this view for use in other tools. Actions you can perform in this view include: ▪ **Edit** ▪ **Compile** ▪ **Download** ▪ **Drop** ▪ **Find** **See Also:** "Editing a Function" on page 16-25, "Compiling a Function" on page 16-25, "Downloading a Function" on page 16-25, and "Dropping a Function" on page 16-26
Dependencies	Displays objects that use (or depend) on the current function and objects the function depends on.
Errors	Displays errors related to the current function.

Table 16–9 (Cont.) Available Views for Functions

View	Description
Grants	Lists details of grants for the current function, including privilege, grantee, grantable, grantor, and object name.

Editing a Function

When you edit a function you can edit the code manually, perform a search and replace, and compile the function.

Editing a Function Manually

To edit a function manually:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Functions**.
3. From the Object Selection pane, select a function.

   The Code view appears. Be default, you can copy code from this view for use in other tools.
4. Click **Edit** to activate manual edit mode.
5. Click **Find** to perform a basic search and replace.

Note: You can expand the Edit pane by clicking the Full Screen icon in the upper right of the pane, beneath the Create button.

Compiling a Function

If you edit and make changes to a function, you need to compile in order to save your changes. There is no save function because this is just a view of the object within the database.

To compile a function in Object Browser:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Functions**.
3. From the Object Selection pane, select a function.

   The Code view appears.
4. Click **Compile** to compile the current function.

   Compiling re-creates the object in the database. If the compile fails, an error message display above the code.

Downloading a Function

To save a function to a file in Object Browser:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. From the Object list, select **Functions**.
3. From the Object Selection pane, select a function.

   The Code view appears.
4. Click **Download** to save the current function as a file.

Dropping a Function

To drop a function in Object Browser:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Functions**.
3. From the Object Selection pane, select a function.

   The Code view appears.
4. Click **Drop** to delete the current function.

Managing Triggers

A database trigger is a stored subprogram associated with a database table, view, or event. The trigger can be called once, for example when an event occurs, or many times, for example for each row affected by an INSERT, UPDATE, or DELETE statement.

Topics in this section include:

- Creating Triggers
- Browsing a Trigger
- Editing a Trigger
- Compiling a Trigger
- Downloading a Trigger
- Dropping a Trigger

See Also:

- *Oracle Database Concepts*
- *Oracle Database Application Developer's Guide - Fundamentals*
- "Using Triggers" in *Oracle Database Express Edition 2 Day Developer Guide*

Creating Triggers

To create a trigger in Object Browser:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. Click **Create**.
3. From the list of object types, select **Trigger**.
4. Select a table name and click **Next**.

5. Select the appropriate trigger attributes, enter the trigger body, and click **Next**.

   A confirmation page appears. To view the SQL used to create the trigger, click **SQL**.

6. Click **Finish**.

Browsing a Trigger

To browse a trigger in Object Browser:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. From the Object list, select **Triggers**.
3. From the Object Selection pane, select a trigger.

   The Details view appears.

Summary of Available Views

Click the tabs at the top of the page to view different reports about the trigger. Table 16–10 describes all available views.

Table 16–10 Available Views for Triggers

View	Description
Object Details	(Default) Lists of the details about the current trigger. Actions you can perform include: ■ **Compile** ■ **Disable** ■ **Download** ■ **Drop** ■ **Code** **See Also:** "Editing a Trigger" on page 16-27, "Compiling a Trigger" on page 16-28, "Downloading a Trigger" on page 16-28, and "Dropping a Trigger" on page 16-28
Errors	Displays errors related to the current trigger.
SQL	Displays the SQL necessary to re-create the trigger.

Editing a Trigger

When you edit a trigger you can edit the code manually, perform a search and replace, and compile the trigger.

Editing a Trigger Manually

To edit a trigger manually:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. From the Object list, select **Triggers**.
3. From the Object Selection pane, select a trigger.
4. Select the **Code** tab and then click **Edit** to activate manual edit mode.

5. Click **Find** to perform a basic search and replace.

Note: You can expand the Edit pane by clicking the Full Screen icon in the upper right of the pane, beneath the Create button.

Compiling a Trigger

If you edit and make changes to a function, you need to compile in order to save your changes. There is no save function because this is just a view of the object within the database.

To compile a trigger in Object Browser:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Triggers**.
3. From the Object Selection pane, select a trigger.

   The Details view appears.
4. Click **Compile** to compile the current trigger.

   Compiling re-creates the object in the database. If the compile fails, an error message display above the code.

Downloading a Trigger

To save the current trigger as a file:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Triggers**.
3. From the Object Selection pane, select a trigger.

   The Details view appears.
4. Click **Download** to save the current trigger as a file.

Dropping a Trigger

To save drop a trigger in Object Browser:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Triggers**.
3. From the Object Selection pane, select a trigger.

   The Details view appears.
4. Click **Drop** to delete the current trigger.

Managing Database Links

A database link is a schema object in one database that enables you to access objects in another database. Once you create a database link, you can access the remote objects by appending `@dblink` to the table or view name, where `dblink` is the name of the database link.

Topics in this section include:

- Creating a Database Link
- Browsing a Database Link
- Dropping a Database Link

Creating a Database Link

To create a database link:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. Click **Create**.
3. From the list of object types, select **Database Link**.
4. Specify the following information and click **Next**.
   - Database Link Name
   - Connect To Schema
   - Password
   - Remote Hostname or IP
   - Remove Host Port
   - SID or Service Name

   A confirmation page appears.
5. To view the SQL used to create the database link, click **Show SQL**.
6. Click **Create**.

Browsing a Database Link

To browse a database link:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Database Links**.
3. From the Object Selection pane, select a database link.

   The Object Details view appears.

Summary of Available Views

Click the tabs at the top of the page to view different reports about the database link. Table 16–11 describes all available views.

Table 16–11 Available Views for Database Links

View	Description
Object Details	(Default) Displays details about the database link. Actions you can perform include: ■ **Drop** - Deletes the database link ■ **Test** - Tests the database link
Dependencies	Displays a list of objects that use (or depend) upon this database link.
SQL	Displays the SQL necessary to re-create this database link.

Dropping a Database Link

To drop a database link in Object Browser:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Database Links**.
3. From the Object Selection pane, select a database link.

   The Object Details View appears.
4. Click **Drop**.

Managing Materialized Views

A materialized view provides indirect access to table data by storing the results of a query in a separate schema object. Unlike an ordinary view, which does not take up any storage space or contain any data, a materialized view contains the rows resulting from a query against one or more base tables or views. A materialized view can be stored in the same database as its base tables or in a different database.

Topics in this section include:

- Creating a Materialized View
- Browsing a Materialized View
- Dropping a Materialized View

See Also: *Oracle Database Concepts* for information about materialized views

Creating a Materialized View

To create a materialized view:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. Click **Create**.
3. From the list of object types, select **Materialized View**.
4. Define the materialized view:
   a. **Materialized View Name** - Enter a name.
   b. **Query** - Specify a query to define the view.

To access Query Builder or SQL Command Processor, click the appropriate link at the bottom of the page. The selected tool appears in a pop-up window. Once you generate the appropriate SQL, click **Return** to automatically close the popup window and return to the wizard with the SQL.

c. Click **Next**.

   A confirmation page appears. To view the SQL used to create the materialized view, click **SQL**.

5. Click **Create**.

   See Also: "Building Queries with Query Builder" on page 17-1 and "Using SQL Commands" on page 19-1

Browsing a Materialized View

To view a materialized view:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.

2. From the Object list, select **Materialized Views**.
3. From the Object Selection pane, select a view.

   The Materialized View appears.

Summary of Available Views

Click the tabs at the top of the page to view different reports about the materialized view. Table 16–12 describes all available views.

Table 16–12 Available Views for Materialized View

View	Description
Materialized View	(Default) Displays details about the columns in the materialized view, including: ■ Column Name ■ Data type ■ Nullable flag ■ Default value ■ Primary key Click **Drop** to delete the current materialized view.
Data	Displays a report of the data in the columns. Actions you can perform include: ■ **Query** - Enables you to sort by column. To restrict specific rows, enter a condition in the Column Condition field. Use the percent sign (%) for wildcards. From Order by, select the columns you want to review and click Query. ■ **Count Rows** - Displays a report of the data in the current table.
Details	Displays object details stored in `DBA_SNAPSHOTS` such as updatable and status.
Grants	Displays a list of grants on the current view, including grantee, privilege, and grant options. Actions you can perform in this view include **Grant** and **Revoke**.

Table 16–12 (Cont.) Available Views for Materialized View

View	Description
Dependencies	Displays a list of objects that use (or depend) upon this materialized view.
SQL	Displays the SQL necessary to re-create this materialized view.

Dropping a Materialized View

To drop a materialized view in Object Browser:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Materialized Views**.
3. From the Object Selection pane, select a view.

   The Materialized View appears.
4. Click **Drop**.

Managing Synonyms

A synonym is an alias for a schema object. Synonyms can provide a level of security by masking the name and owner of an object and by providing location transparency for remote objects of a distributed database. Also, they are convenient to use and reduce the complexity of SQL statements for database users.

Topics in this section include:

- Creating Synonyms
- Viewing a Synonym
- Dropping a Synonym

See Also: *Oracle Database Administrator's Guide* for information about synonyms

Creating Synonyms

To create a synonym:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. Click **Create**.
3. From the list of object types, select **Synonym**.
4. Define the synonym:
   a. **Synonym Name** - Enter a name.
   b. **Public or Private** - Specify whether the synonym should be public or private.
   c. **Schema** - Select the database schema (or username) which owns the object upon which you want to create your synonym.
   d. **Object** - Enter the name of the object upon which you want to create a synonym.

   e. **Database Link** - Enter the name of the database link to use if the synonym is to be create on a remote object.
   f. Click **Next**.

   A confirmation page appears. To view the SQL used to create the synonym, click **Show SQL**.
5. Click **Finish**.

See Also: "Managing Synonyms" on page 16-32

Viewing a Synonym

To view a synonym:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Synonyms**.
3. From the Object Selection pane, select a synonym.

   The Object Details view appears displaying the following:
   - Synonym owner
   - Synonym name
   - Object owner
   - Object Name
   - Object Status
   - Status

Dropping a Synonym

To drop a synonym in Object Browser:

1. On the Database Home Page, click the **Object Browser** icon.

   Object Browser appears.
2. From the Object list, select **Synonyms**.
3. From the Object Selection pane, select a synonym.
4. Click **Drop**.

17 Building Queries with Query Builder

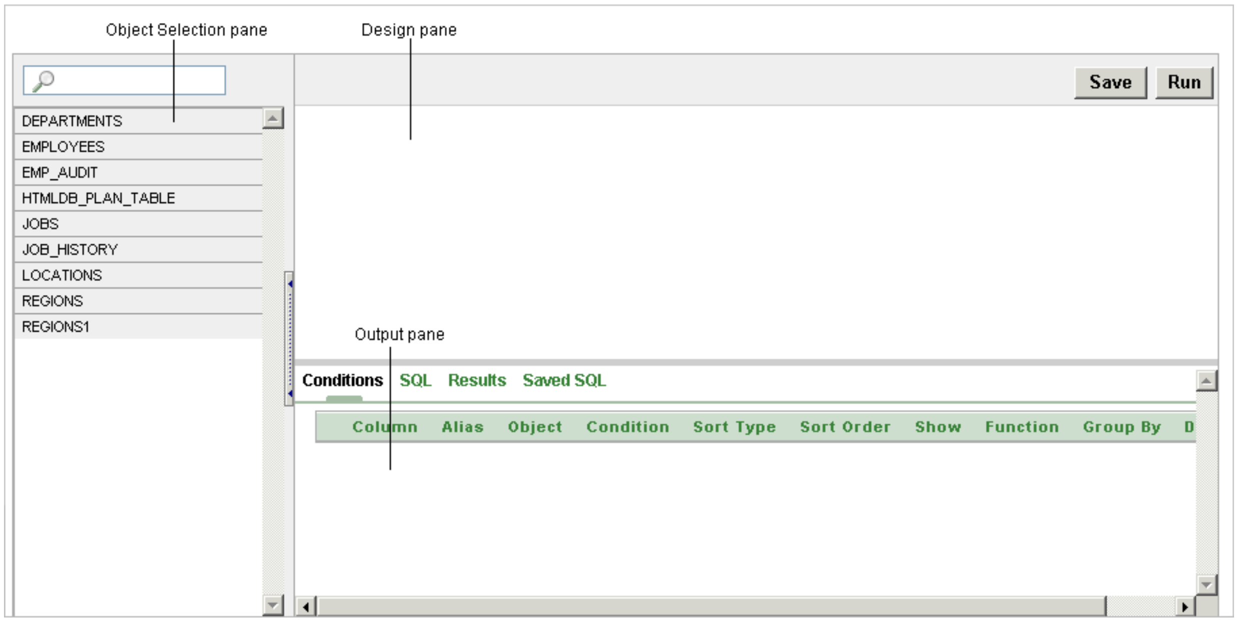
Query Builder's graphical user interface enables database developers to build SQL queries without the need for manual SQL coding. Using Query Builder, you can search and filter database objects, select objects and columns, create relationships between objects, view formatted query results, and save queries with little or no SQL knowledge.

This section contains the following topics:

About Query Builder

The Query Builder page is divided into three sections:

- **Object Selection pane** displays on the left side of the page and contains a list objects from which you can build queries. Only objects in the current schema display.
- **Design pane** displays to the right of the Object Selection pane and above the Conditions, SQL, Results, and Saved SQL tabs. When you select an object from the Object Selection pane, it appears in the Design pane.
- **Output pane** displays below the Design pane. Once you select objects and columns, you can create conditions, view the generated SQL, or view query results.



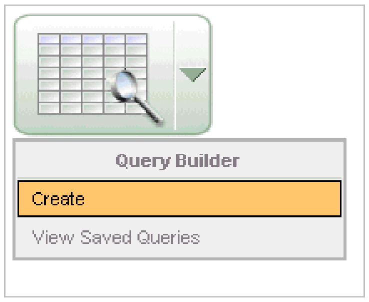
Topics in this section include:

- Accessing Query Builder
- Understanding the Query Building Process

Accessing Query Builder

To access Query Builder:

1. On the Database Home Page, click the **SQL** icon.
2. To view Query Builder you can either:
   - Click the **Query Builder** icon.
   - Click the down arrow on the right side of the icon to view a drop down menu. Then, select the appropriate menu option.



Note: For the purposes of consistency, this document uses the primary navigation path (or drill-down approach) when explaining navigation.

Understanding the Query Building Process

To build a a query in Query Builder, you perform the following steps:

1. Select objects from the Object Selection pane. See "Using the Object Selection Pane" on page 17-3.
2. Add objects to the Design pane and select columns. See "Selecting Objects" on page 17-4.
3. **Optional**: Establish relationships between objects. See "Creating Relationships Between Objects" on page 17-7.
4. **Optional**: Create query conditions. See "Specifying Query Conditions" on page 17-6.
5. Execute the query and view results. See "Viewing Query Results" on page 17-11.

See Also: "Viewing Generated SQL" on page 17-11 and "Working with Saved Queries" on page 17-9

Using the Object Selection Pane

The Object Selection pane displays on the left side of the Query Builder page and lists tables, views, and materialized views within the current schema.

Topics in this section include:

- Searching and Filtering Objects
- Hiding the Object Selection Pane

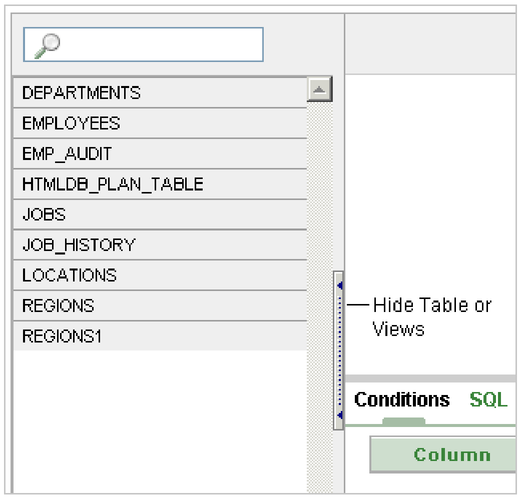
Searching and Filtering Objects

Use the Object Selection pane to search for and view tables, views, and materialized views within the current schema.

To search or filter objects:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **Query Builder** icon.

   Query Builder appears.
3. In the search field at the top of the pane, enter a case insensitive query.
4. To view all tables or views within the currently selected schema, leave the search field blank.



Hiding the Object Selection Pane

You can hide the Object Selection pane by selecting the **Hide Table or Views** control. By hiding the Object Selection pane, you can increase the size of the Design and Result panes.

The Hide Table or Views control displays on the right side of the Object Selection pane. If the Object list appears, selecting this control hides it. Similarly, if the Object list is hidden, selecting this control causes the pane to reappear.

Selecting Objects

The Design pane displays to the right of the Object Selection pane. When you select an object from the Object Selection pane, it appears in the Design pane. You use the Object Selection pane to select objects (that is, tables, views, and materialized views) and the Design pane to identify how those selected objects will be used in a query.

Topics in this section include:

- About Supported Column Types
- Adding an Object to the Design Pane
- Removing or Hiding Objects in the Design Pane

See Also: "Creating Relationships Between Objects" on page 17-7

About Supported Column Types

Columns of all types available in Oracle Database 10g Release (10.2) display as objects in the Design pane. Note the following column restrictions:

- You may only select a maximum of 60 columns for each query.
- The following column types are not selectable and cannot be included in a generated query:
  - `BLOB`
  - `NCLOB`
  - `RAW`
  - `LONG`
  - `LONG RAW`
  - `XMLType`
  - Any other nonscalar column types

Adding an Object to the Design Pane

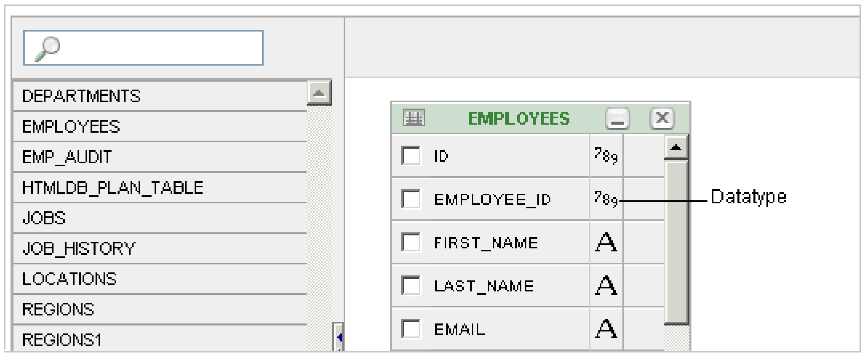
You add an object to the Design pane by selecting it from the Object Selection pane.

To add an object to the Design pane:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **Query Builder** icon.

   Query Builder appears.
3. Select an object from the Object Selection pane.

The selected object appears in the Design Pane. Note that a graphical representation of the datatype displays to the right of the column name.



4. Select the columns to be included in your query by clicking the check box to the left of the column name.

   When you select a column you are indicating it will be used in the query. As you select a column, it appears on the Conditions tab. Note that the Show check box on the Conditions tab controls whether a column is included in query results. Be default, this check box is selected.

   To select the first twenty columns, click the small icon in the upper left corner of the object and then select **Check All**.

5. To execute the query and view results, click **Run**.

   Tip: You can also execute a query by pressing **CTRL + ENTER**.

   The Results pane displays the query results.

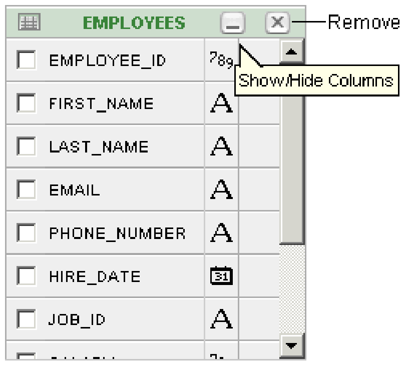
   See Also: "Specifying Query Conditions" on page 17-6

Resizing the Design and Results Panes

As you select objects, you can resize Design and Results panes by selecting the grey horizontal rule in the center of the page. Moving the rule up, shrinks the Design pane. Moving the rule down expands the Design pane.

Removing or Hiding Objects in the Design Pane

You remove or hide objects in the Design pane by selecting controls at the top of the object. To remove an object, select the **Remove** icon in the upper right corner. To temporarily hide the columns within an object, click the **Show/Hide Columns** icon.



Specifying Query Conditions

Conditions enable you to filter and identify the data you want to work with. As you select columns within an object, you can specify conditions on the Conditions tab. You can use these attributes to modify the column alias, apply column conditions, sort columns, or apply functions.

To specify query conditions:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **Query Builder** icon.

   Query Builder appears.
3. Select an object from the Object Selection pane.

   The selected object appears in the Design Pane.
4. Select the columns to be included in your query by clicking the box to the left of the column name.

   When you select a column, you are indicating you want to include it in your query. As you select each column, it appears as a separate row in the Conditions view. Table 17–1 describes the attributes available on the Conditions tab.

Table 17–1 Conditions Tab

Condition Attribute	Description
Up and Down Arrows	Controls the order that the columns to be displayed in the resulting query. Click the arrow buttons to move columns up and down. **See Also:** "Viewing Query Results" on page 17-11
Column	Displays the column name.
Alias	Specify an optional column alias. An **alias** is an alternative column name. Aliases are used to make a column name more descriptive, to shorten the column name, or prevent possible ambiguous references.

Condition	Specify a condition for the column.
	The condition you enter modifies the query's `WHERE` clause. When specifying a column condition, you must include the appropriate operator and operand. Consider the following examples:
	`>=10`
	`='VA'`
	`IN (SELECT dept_no FROM dept)`
	`BETWEEN SYSDATE AND SYSDATE + 15`

Sort Type	Select a sort type. Options include: ■ Ascending (`Asc`) ■ Descending (`Desc`)
Sort Order	Enter a number (1, 2, 3, and so on) to specify the order in which selected columns should display.

Table 17–1 (Cont.) Conditions Tab

Condition Attribute	Description
Show	Select this check box to include the column in your query results. You do not need to select Show if you need to add a column to the query for filtering only. For example, suppose you wish to create following query: `SELECT ename FROM emp WHERE deptno = 10` To create this query in Query Builder: **1.** From the Object list, select EMP. **2.** In the Design Pane, select `ename` and `deptno`. **3.** For the deptno column, in Condition enter `=10` and uncheck the Show check box.
Function	Select an argument function. Available functions include: ■ **NUMBER columns** - `COUNT`, `COUNT DISTINCT`, `AVG`, `MAXIMUM`,. `MINIMUM`, `SUM` ■ **VARCHAR2, CHAR columns** - `COUNT`, `COUNT DISTINCT`, `INITCAP`, `LENGTH`, `LOWER`, `LTRIM`, `RTRIM`, `TRIM`, `UPPER` ■ **DATE, TIMESTAMP columns** - `COUNT`, `COUNT DISTINCT`
Group By	Specify columns to be used for grouping when an aggregate function is used. Only applicable for columns included in output.
Delete	Deselect the column, excluding it from the query.

As you select columns and define conditions, Query Builder writes the SQL for you.

5. To view the underlying SQL, click the **SQL** tab.

Creating Relationships Between Objects

You can create relationships between objects by creating a join. A **join** identifies a relationship between two or more tables, views, or materialized views.

Topics in this section include:

- About Join Conditions
- Joining Objects Manually
- Joining Objects Automatically

About Join Conditions

When you write a join query, you specify a condition that conveys a relationship between two objects. This condition is called a **join condition**. A join condition determines how the rows from one object will combine with the rows from another object.

Query Builder supports inner, outer, left, and right joins. An **inner join** (also called a **simple join**) returns the rows that satisfy the join condition. An outer join extends the result of a simple join. An **outer join** returns all rows that satisfy the join condition and returns some or all of those rows from one table for which no rows from the other satisfy the join condition.

> **See Also:** *Oracle Database SQL Reference* for information about join conditions

Joining Objects Manually

You can create a join manually by selecting the Join column in the Design pane.

To join two objects manually:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **Query Builder** icon.

   Query Builder appears.
3. From the Object Selection pane, select the objects you want to join.

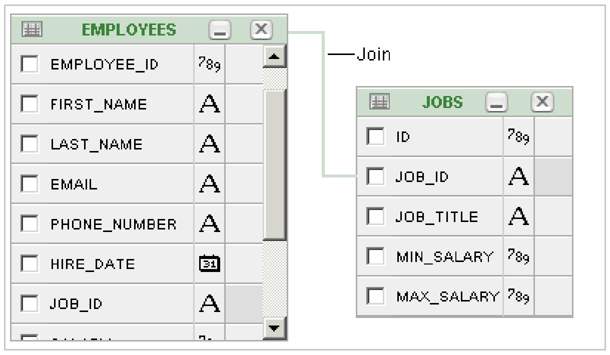
   The objects display in the Design pane.
4. Identify the columns you want to join.

   You create a join by selecting the Join column adjacent to the column name. The Join column displays to the right of the datatype, beneath the Remove icon. When your cursor is in the appropriate position, the following help tip displays:

   ```
   Click here to select column for join
   ```
5. Select the appropriate Join column for the first object.

   When selected, the Join column displays as a dark gray. To deselect a Join column, simply select is again or press **ESC**.
6. Select the appropriate Join column for the second object.

   > **Tip:** You can also join two objects by dragging and dropping. Select a column in the first table and then drag and drop it onto a column in another table.



   When joined, a green line connects the two columns.
7. Select the columns to be included in your query. You can view the SQL statement resulting from the join by positioning the cursor over the green line.
8. Click **Run** to execute the query.

   The Results pane displays the query results.

Joining Objects Automatically

When you join objects automatically, the Query Builder suggests logical, existing parent and child relationships between existing columns.

To join objects automatically:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **Query Builder** icon.

   Query Builder appears.
3. From the Object Selection pane, select an object.

   The object displays in the Design pane.
4. Click the small icon in the upper left corner of the object. Depending upon the selected object, the icon label displays as **Table Actions** or **View Actions**.

   The Actions window appears. Use the Actions window to select all columns within the current object or objects related to the current object.
5. In the Actions window, select the appropriate options:
   - **Check All** - Select this option to select the first twenty columns in the current object.
   - **Add Parent** - Displays tables that are referenced as a foreign key to the current object.
   - **Add Child** - Displays tables that reference the current object in a foreign key.

   If using Add Parent or Add child, the selected object appears and a green line connects the foreign key columns.
6. Select additional columns to be included in your query.

   You can view the SQL statement resulting from the join by positioning the cursor over the green line.
7. Click **Run** to execute the query.

   The Results pane displays the query results.

Working with Saved Queries

As you create new queries, you can save them by clicking the Save button in the Design pane. Once you save a query, you can access it later in the Saved SQL view.

Topics in this section include:

- Saving a Query
- Editing a Saved Query
- Deleting a Saved Query

Saving a Query

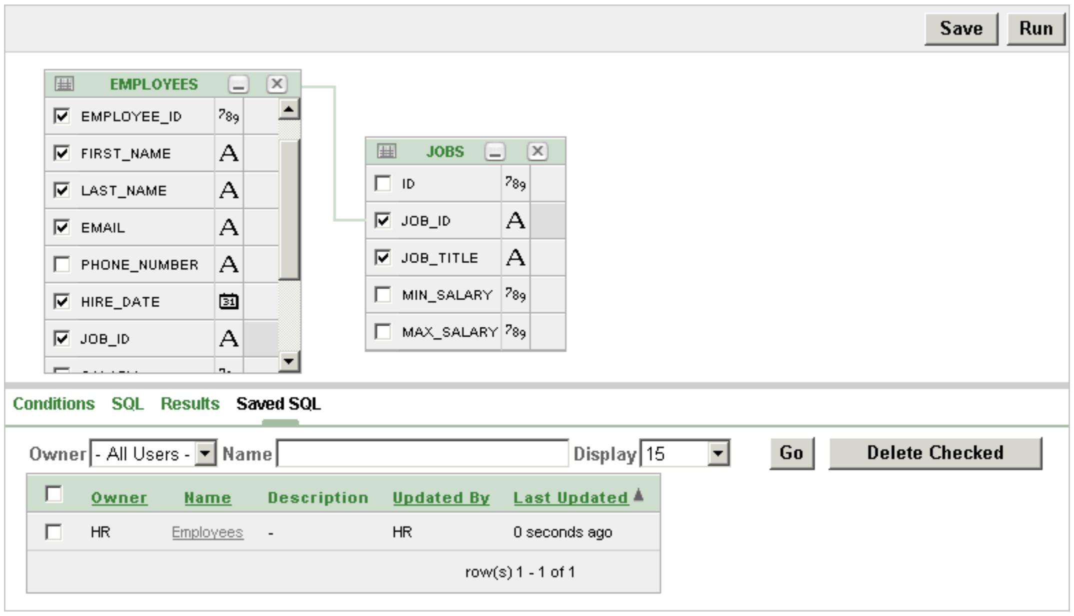
To save a query:

1. Build a query:
   a. On the Database Home Page, click the **SQL** icon.
   b. Click the **Query Builder** icon.

   c. Select objects from the Object Selection pane.
   d. Add objects to the Design pane and select columns.
   e. Execute the query.

2. Click **Save**.
3. Enter a name and description and click **Save**.

   The saved query displays in the Saved SQL view.



	Owner	Name	Description	Updated By	Last Updated
	HR	Employees	-	HR	0 seconds ago

row(s) 1 - 1 of 1

Note that Query Builder does not support duplicate query names. If you open an existing query, keep the existing name, and save it again, Query Builder over-writes the existing query. If you change the name of an existing query and save it again,Query Builder saves the query again under the new name.

Editing a Saved Query

Once you save a query, you can access it in the Saved SQL view.

To edit a Saved SQL query:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **Query Builder** icon.

   Query Builder appears.
3. Select the **Saved SQL** tab.
4. To filter the display, you can:
   - Make a selection from the Owner list and click **Go**.
   - Enter a search query in the Name field and click **Go**.
5. To edit a query, select the appropriate name.

   The saved query appears. The selected objects display in the Design pane and the Conditions view appears.

Deleting a Saved Query

To delete a Saved SQL query:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **Query Builder** icon.

   Query Builder appears.
3. Select the **Saved SQL** tab.
4. Select the queries to be deleted and click **Delete Checked**.

Viewing Generated SQL

The SQL view presents a read-only, formatted representation of the SQL generated by Query Builder. You can copy the SQL code that appears in the SQL View for use in other tools such as SQL Command Processor or Application Builder.

See Also: "Using SQL Commands" on page 19-1

To access the SQL view:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **Query Builder** icon.

   Query Builder appears.
3. Select an object from the Object Selection pane.

   The selected object appears in the Design Pane.
4. Select the columns to be included in your query.
5. Click the **SQL** tab.

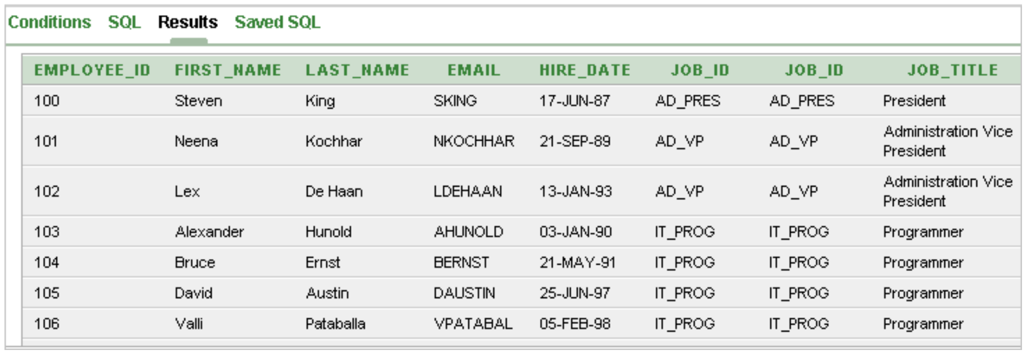
   The SQL code generated by Query Builder appears.

Viewing Query Results

Once you select objects and determine what columns to include in your query, you execute a query by:

- Clicking the **Run** button (or pressing **CTRL + ENTER**)
- Selecting the **Results** tab

The Results view appears, displaying formatted query results.



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	JOB_ID	JOB_TITLE
100	Steven	King	SKING	17-JUN-87	AD_PRES	AD_PRES	President
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	AD_VP	Administration Vice President
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	AD_VP	Administration Vice President
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	IT_PROG	Programmer
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	IT_PROG	Programmer
105	David	Austin	DAUSTIN	25-JUN-97	IT_PROG	IT_PROG	Programmer
106	Valli	Pataballa	VPATABAL	05-FEB-98	IT_PROG	IT_PROG	Programmer

18 Using SQL Scripts

This section provides information on how to use SQL Scripts within Oracle Database Express Edition (Oracle Database XE).

This section contains the following topics:

What is SQL Scripts?

A SQL script is a set of SQL commands saved as a file in SQL Scripts. A SQL script can contain one or more SQL statements or PL/SQL blocks. You can use SQL Scripts to create, edit, view, run, and delete script files.

When using SQL Scripts, remember the following:

- SQL Command Line (SQL*Plus) commands in a SQL script are ignored at runtime.
- There is no interaction between SQL Commands and SQL Scripts.
- You can cut and paste a SQL command from the SQL Script editor to run it in SQL Commands.
- SQL Scripts does not support bind variables.

See Also: "Using SQL Commands" on page 19-1 and "About Long Operations" on page 21-18

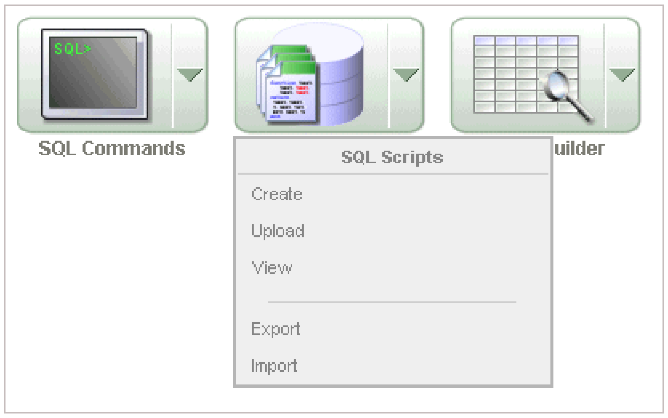
Accessing SQL Scripts

To access SQL Scripts:

1. Log in to the Database Home Page.

2. To view SQL Scripts page you can either:

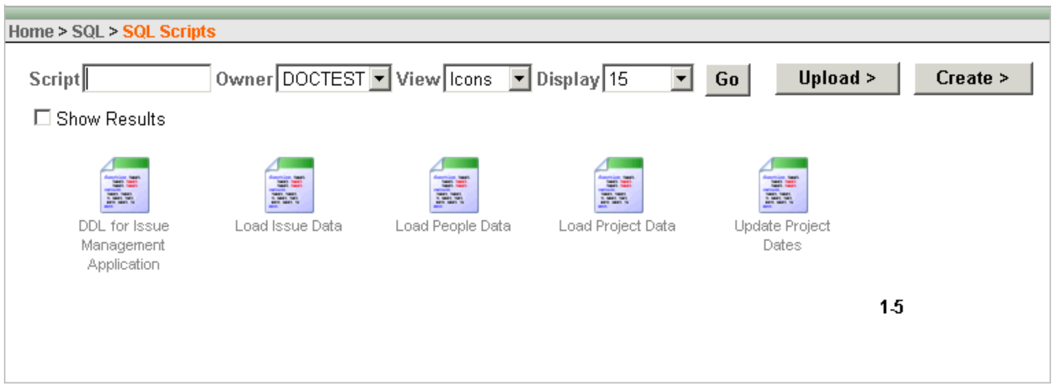
   - Click the **SQL** icon and then **SQL Scripts** to drill-down to the SQL Scripts page.
   - Click the down arrow on the right side of the SQL icon to view a drop down menu. Then select the **SQL Scripts** menu option.



Note: For the purposes of consistency, this document uses the primary navigation path (or drill-down approach) when explaining navigation.

About the SQL Scripts Page

The SQL Scripts page display all SQL scripts created by the current user. You can control the appearance of the page by making a selection from the View list. The default view, Icons, displays each script as an icon. Details view displays each script as a row in a report.



Script | Owner DOCTEST | View Icons | Display 15

DDL for Issue Management Application
Load Issue Data
Load People Data
Load Project Data
Update Project Dates

The SQL Scripts page features the following controls:

- **Script.** Search for a script by entering the script name, or a partial name, in the Script field and clicking **Go**. You control how many rows display by making a selection from the Display list.
- **Owner.** Search for the owner of the script you want to view by entering the username in the Owner field and clicking **Go**.
- **View.** Change the appearance of the SQL Scripts page by making a selection from the View list and clicking **Go**. Available View options include:

- **Icons** (default) displays each script as an icon identified by the script name. Click the **Show Results** check box to additionally display run results as icons identified by the script name.
- **Details** displays each script as a line in a report. Each line includes a check box to enable the selection of scripts for deletion, an edit icon to enable the script to be loaded into the script editor, the script name, the script owner, when the script was last updated and by who, the size in bytes, the number of times the script has been run linked to the run results, and an icon to enable the script to be run.

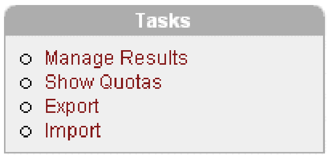
  Details view offers the following additional controls:

  - **Delete Checked.** In Details view, select the check box associated with the script you want to delete and click **Delete Checked**. See "Deleting a SQL Script" on page 18-6.
  - **Sort.** In Details view, click a column heading to sort the listed scripts by that column.

- **Upload.** Click **Upload** to upload a script from your local file system into SQL Scripts. See "Creating a SQL Script" on page 18-3.
- **Create.** Click **Create** to create a new script in the Script Editor. See "Creating a SQL Script" on page 18-3.

About the Tasks List

A Tasks list displays on the right side of the SQL Scripts page.



The Task list contains the following links:

- **Manage Results** enables you to view, search, and display results. See "Viewing SQL Script Results" on page 18-9.
- **Show Quotas** displays the Script Quotas page. The Script Quotas page shows the maximum size of a single result, the maximum size of all results, the quota used and the quota free. It also shows the maximum size of a SQL Script.
- **Export** enables you to export multiple scripts from the current SQL Script Repository for import into SQL Scripts in a different database user account. The scripts you select to export are encoded in a single export script written to your local file system. The export script is named *workspace_name*_script.sql by default. See "Exporting and Importing SQL Scripts" on page 18-11.
- **Upload** enables you to import a script exported by this, or a different workspace. **Import** only imports scripts encoded in an export script created using **Export**. The export script to import must be accessible on your local file system. See "Exporting and Importing SQL Scripts" on page 18-11.

Creating a SQL Script

You can create a new script in the Script Repository by:

- Creating a new script in the Script Editor

- Uploading a script from your local file system

Topics in this section include:

- Creating a SQL Script in the Script Editor
- Uploading a SQL Script

Creating a SQL Script in the Script Editor

To create a new SQL script in the Script Editor:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **SQL Scripts** icon.

   The SQL Scripts page appears.
3. Click the **Create** button.

   The Script Editor appears.
4. Enter a name for the script in the Script Name field.

   Script name extensions are optional.
5. Enter the SQL statements, PL/SQL blocks and SQL Command Line (SQL*Plus) commands you want to include in your script.

   Remember that SQL Command Line commands are ignored at runtime.
6. Click **Save** to save your script to the repository.

   The SQL Scripts page appears listing your newly saved script.

Uploading a SQL Script

To upload a script from your local file system:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **SQL Scripts** icon.

   The SQL Scripts page appears.
3. Click the **Upload** button.

   The Upload Script dialog appears.
4. To upload a script you can either:
   - Enter the name and path to the script you want to upload to the Script Repository.
   - Click **Browse** to locate the script you want to upload.
5. Optionally rename the script by entering the new name in the Script Name field.

   This is the name given to the script in the Script Repository.
6. Click **Upload** to add the script to the Script Repository.

   The SQL Scripts page appears listing your newly uploaded script.

   The script is parsed during upload. If it has a syntax error, an error icon appears in place of the run icon in the SQL Scripts page Details view.

   If a script of the same name exists in the Script Repository, you are prompted to rename it.

Using the Script Editor

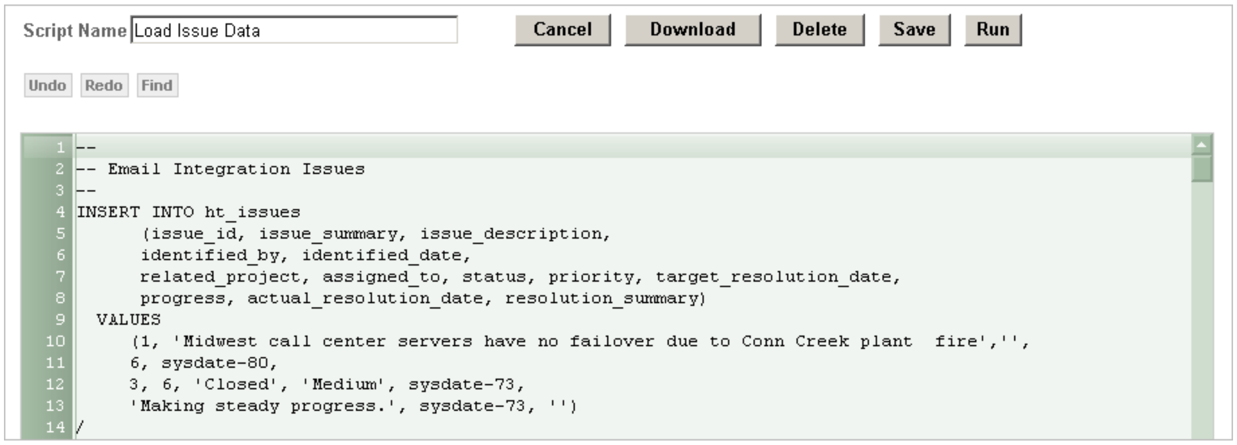
You use the Script Editor to add content to a new script, to edit existing scripts, and to run and delete scripts in the script repository.

To edit a SQL script:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **SQL Scripts** icon.

   The SQL Scripts page appears.
3. You can load a script into the editor as follows:
   - In Icons view, click the script icon.
   - In Details view, click the **Edit** icon.

   The Script Editor appears.



```
--
-- Email Integration Issues
--
INSERT INTO ht_issues
      (issue_id, issue_summary, issue_description,
      identified_by, identified_date,
      related_project, assigned_to, status, priority, target_resolution_date,
      progress, actual_resolution_date, resolution_summary)
  VALUES
     (1, 'Midwest call center servers have no failover due to Conn Creek plant  fire','',
     6, sysdate-80,
     3, 6, 'Closed', 'Medium', sysdate-73,
     'Making steady progress.', sysdate-73, '')
/
```

4. Edit the script. See "About the Script Editor" on page 18-5.

   You can test your script during editing by running the script to reveal errors. The Run Script dialog and the Script Results pages enable you to resume editing the script. See "Executing a SQL Script" on page 18-7, and "Viewing SQL Script Results" on page 18-9.
5. Click **Save** to save your script to the Script Repository,

   The SQL Scripts page appears.

About the Script Editor

You can perform the following actions in the Script Editor:

- **Cancel the editing session.** Click **Cancel** to exit the Script Editor without saving changes made since you last saved. The SQL Scripts page appears.
- **Save the script to your local file system.** Click **Download** to save a copy of the current script to your local file system. Enter a name for the script on your local file system and a directory path.
- **Delete the script from the Script Repository.** Click **Delete** to remove the current script from the Script Repository. See "Deleting a SQL Script" on page 18-6.
- **Save the script to the Script Repository.** Click **Save** to save your changes to the script to the Script Repository. The SQL Scripts page appears.

- **Execute the script.** Click **Run** to submit the script for execution. See "Executing a SQL Script" on page 18-7.
- **Undoing/redoing edits.** Click **Undo** (Ctrl+Z) and **Redo** (Ctrl+Y) to undo or redo line edits in the Script Editor.
- **Searching in your script.** Click **Find** to display the text and JavaScript regular expression find and replace options. Click **Find** again to hide the options.
- **Selecting a line.** Click the line number on the left side of the Script Editor to select the associated line of your script for copying or deleting.
- **Cutting and Pasting.** Use standard edit controls to cut, copy and paste content in the Script Editor.
- **Auto indenting lines.** New lines automatically indent to the previous line start column.

Deleting a SQL Script

You can delete scripts from the Script Repository by:

- Deleting selected scripts from the SQL Scripts page
- Deleting the current script in the Script Editor

Topics in this section include:

- Deleting Scripts from the SQL Scripts Page
- Deleting a Script in the Script Editor

Deleting Scripts from the SQL Scripts Page

To delete scripts from the SQL Scripts page.

1. On the Database Home Page, click the **SQL** icon.
2. Click the **SQL Scripts** icon.

   The SQL Scripts page appears. Change the page view to Details.
3. From the View list, select **Details** and click **Go**.

   Details view appears.
4. Select the scripts to be deleted.

   To select individual scripts, click the check box to the left of the Edit icon. To select all scripts visible in the current page, click the check box in the column heading.
5. Click **Delete Checked** to permanently remove the selected scripts from the Script Repository. You are prompted to confirm this action before the script is deleted.

   The message `"Script(s) deleted"` appears above the updated list of Scripts.

Deleting a Script in the Script Editor

To delete a script in the Script Editor:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **SQL Scripts** icon.

   The SQL Scripts page appears.

3. Open the script you want to delete in the Script Editor.
4. Click **Delete** to permanently remove the script from the Script Repository. You are prompted to confirm this action before the script is deleted.

   The SQL Scripts page appears. The message `"Script(s) deleted"` appears above the updated list of scripts.

Copying a SQL Script

You can copy a script in the Script Repository by saving it with a new name.

To copy a script:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **SQL Scripts** icon.

   The SQL Scripts page appears.
3. Load the script to copy into the editor.
4. Enter a name for the copied script in the Script Name field.
5. Click **Save** to save a copy of the script in the Script Repository.

   The SQL Scripts page appears listing the newly copied script.

Executing a SQL Script

You can execute scripts stored in the Script Repository. You can submit a script for execution either from the Script Editor, or from the SQL Scripts page.

When you submit a script for execution, the Run Script page appears. It displays the script name, when it was created and by who, when it was last updated and by who, the number of statements it contains, and its size in bytes. It also lists unknown statements such as SQL Command Line (SQL*Plus) commands that it will ignore during execution.

Finally, it lists statements with errors. If there are errors, the **Run** control does not appear.

Topics in this section include:

- Executing a SQL Script in the Script Editor
- Executing a SQL Script from the SQL Scripts Page
- About the Run Script Page

See Also: "About Long Operations" on page 21-18

Executing a SQL Script in the Script Editor

To execute a script in the Script Editor:

1. Open the script you want to execute in the Script Editor. See "Using the Script Editor" on page 18-5.
2. Click **Run** in the Script Editor.
3. The Run Script page appears.

The Run Script page displays information about the script and lists statements in error preventing execution, or statements such as SQL Command Line (SQL*Plus) commands that will be ignored when the script is executed.

The Run Script page has three controls:

- **Cancel** returns you to the SQL Scripts page without executing the script.
- **Edit Script** loads the script into the Script Editor. Note that **Edit Script** appears instead of **Run** when a script has errors.
- **Run** to submit the script for execution. Note that **Run** is not available if there are script errors.

4. Click **Run** to submit the script for execution.

   The Manage Script Results page appears listing script results.

5. To view script results, click the **View** icon under View Results.

   See Also: "Viewing SQL Script Results" on page 18-9

Executing a SQL Script from the SQL Scripts Page

To execute a script from the SQL Scripts page:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **SQL Scripts** icon.

   The SQL Scripts page appears. Change the page view to Details.

3. From the View list, select **Details** and click **Go**.

   Details view appears.

4. Click the **Run** icon for the script you want to execute. The Run icon is located on the far right side adjacent to the script name.
5. The Run Script page appears.

   The Run Script page displays information about the script and lists statements in error preventing execution, or statements such as SQL Command Line (SQL*Plus) commands that will be ignored when the script is executed. The Run Script page has three controls:

   Cancel to return to the SQL Scripts page without executing the script.

   Edit Script to load the script into the Script Editor. **Edit Script** appears instead of **Run** when a script has errors.

   Run to submit the script for execution. **Run** is not available for scripts with errors.

6. Click **Run** to submit the script for execution.

   The Manage Script Results page appears listing available results for the script.

7. Click the View icon for the results you want to view. The View icon is at the right end of the scripts listed in the Manage Script Results page.

   See Also: "Viewing SQL Script Results" on page 18-9

About the Run Script Page

On the Run Script page, you can:

- **Cancel the execution.** Click **Cancel** to exit the Run Script page without executing the script. The SQL Scripts page appears.
- **Edit the script. Edit Script** appears instead of **Run** when a script has errors. Click **Edit Script** to load the script into the Script Editor to remove the lines with errors.
- **Execute the script.** Click **Run** to execute the script.

Viewing SQL Script Results

You use the Manage Script Results page to view and delete script results.

You can also select script results to view from the Icons view of the SQL Scripts page, and from the Results column of the SQL Scripts page Details view.

Topics in this section include:

- Viewing Results from the SQL Scripts Page
- About the Results Page

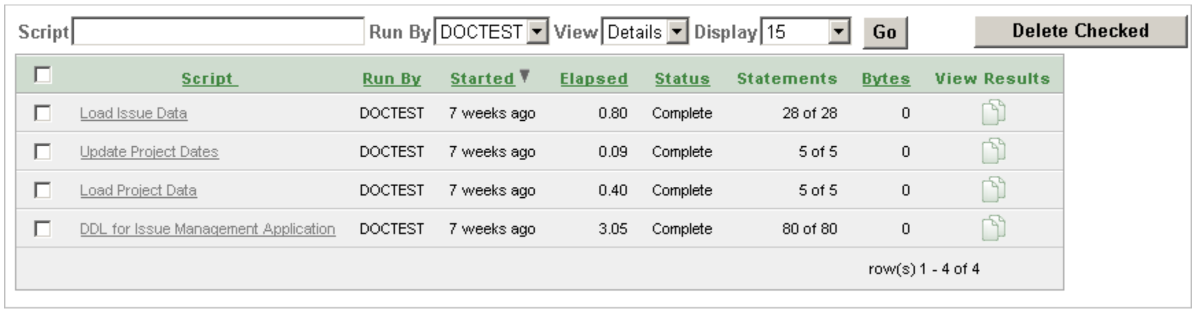
Viewing Results from the SQL Scripts Page

To view script results from the SQL Scripts page:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **SQL Scripts** icon.

   The SQL Scripts page appears.
3. You can access the Manage Script Results page in the following ways:
   - On the Tasks list, click **Manage Results**.
   - In Details view, click the Results number for the script you want to display.
   - In Icons view, click the **Show Results** check box and then the appropriate icon. Results icons only appear in the Icons view if you click the Show Results check box.

   The Manage Script Results page appears, listing available results for the script. See "About the Manage Script Results Page" on page 18-9.



	Script	Run By	Started	Elapsed	Status	Statements	Bytes	View Results
☐	Load Issue Data	DOCTEST	7 weeks ago	0.80	Complete	28 of 28	0	
☐	Update Project Dates	DOCTEST	7 weeks ago	0.09	Complete	5 of 5	0	
☐	Load Project Data	DOCTEST	7 weeks ago	0.40	Complete	5 of 5	0	
☐	DDL for Issue Management Application	DOCTEST	7 weeks ago	3.05	Complete	80 of 80	0	

row(s) 1 - 4 of 4

4. Click the **View** icon for the results you want to view. The View icons display on the far right side of page under the heading View Results.

   The Results page appears. See "About the Results Page" on page 18-10.

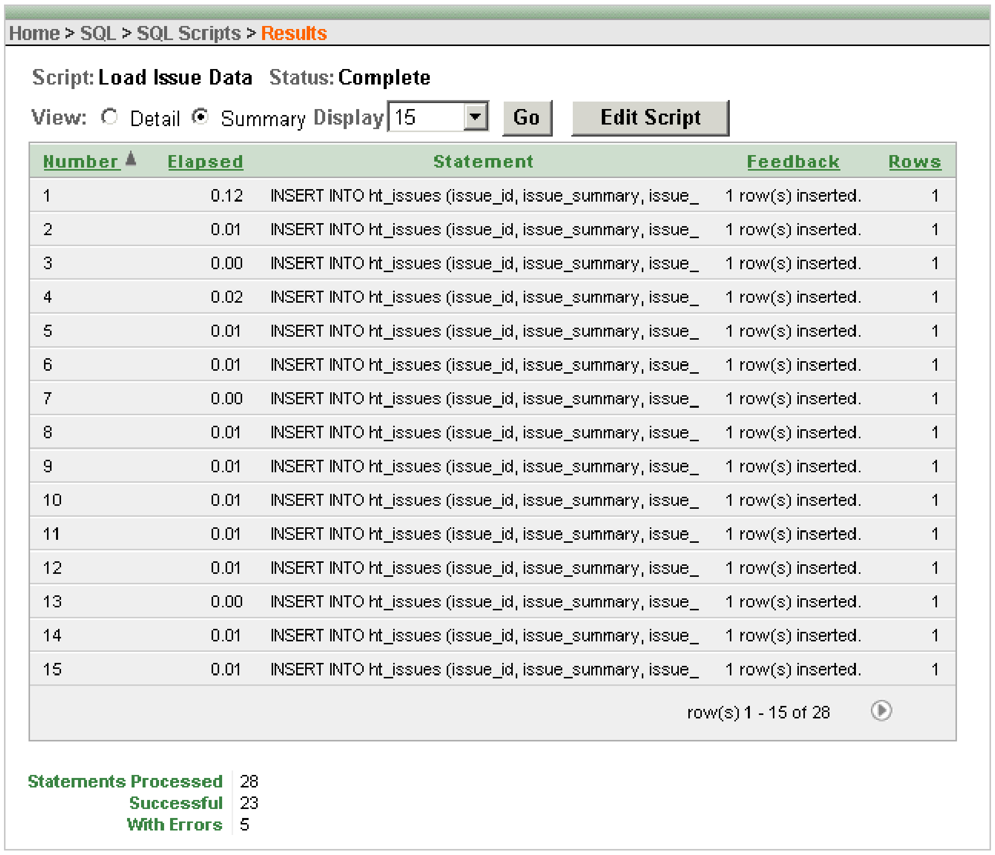
About the Manage Script Results Page

On the Manage Script Results page you can:

- **Search for a result.** Enter a result name or partial name in the Script field and click **Go**. To view all results, leave the Script field blank and click **Go**. You control how many rows display by making a selection from the Display list.
- **Change the Page View.** You can change the appearance of the page by making a selection from the View list. Available View options include:
  – **Icons** displays each result as an icon identified by the script name, and time and date stamp.
  – **Details** displays each result as a line in a report. Each line includes a check box to enable the selection of results for deletion, the associated script name which is a link enabling it to be loaded into the Script Editor, who ran the script, when the run started, how long it took to run, whether the run is complete or not, the number of statements executed, the size in bytes, and a View icon to view the results.
- **Delete a result.** In Details view, select the check box associated with each result you want to delete, and click **Delete Checked**.
- **Sort results.** In Details view, click a column heading to sort the listed results by that column.

About the Results Page

The Results page displays the script name and status (Complete, Canceled, Executing or Submitted), and lists the statements executed.


Home > SQL > SQL Scripts > Results
View: Detail Summary Display 15 Go Edit Script

Script: Load Issue Data Status: Complete

Number	Elapsed	Statement	Feedback	Rows
1	0.12	INSERT INTO ht_issues (issue_id, issue_summary, issue_	1 row(s) inserted.	1
2	0.01	INSERT INTO ht_issues (issue_id, issue_summary, issue_	1 row(s) inserted.	1
3	0.00	INSERT INTO ht_issues (issue_id, issue_summary, issue_	1 row(s) inserted.	1
4	0.02	INSERT INTO ht_issues (issue_id, issue_summary, issue_	1 row(s) inserted.	1
5	0.01	INSERT INTO ht_issues (issue_id, issue_summary, issue_	1 row(s) inserted.	1
6	0.01	INSERT INTO ht_issues (issue_id, issue_summary, issue_	1 row(s) inserted.	1
7	0.00	INSERT INTO ht_issues (issue_id, issue_summary, issue_	1 row(s) inserted.	1
8	0.01	INSERT INTO ht_issues (issue_id, issue_summary, issue_	1 row(s) inserted.	1
9	0.01	INSERT INTO ht_issues (issue_id, issue_summary, issue_	1 row(s) inserted.	1
10	0.01	INSERT INTO ht_issues (issue_id, issue_summary, issue_	1 row(s) inserted.	1
11	0.01	INSERT INTO ht_issues (issue_id, issue_summary, issue_	1 row(s) inserted.	1
12	0.01	INSERT INTO ht_issues (issue_id, issue_summary, issue_	1 row(s) inserted.	1
13	0.00	INSERT INTO ht_issues (issue_id, issue_summary, issue_	1 row(s) inserted.	1
14	0.01	INSERT INTO ht_issues (issue_id, issue_summary, issue_	1 row(s) inserted.	1
15	0.01	INSERT INTO ht_issues (issue_id, issue_summary, issue_	1 row(s) inserted.	1

row(s) 1 - 15 of 28

Statements Processed	28
Successful	23
With Errors	5

On the Results page you can:

- **Choose the view.** Click the **Detail** or **Summary** radio button and click **Go** to specify whether to display complete or summarized information.
- **Choose the number of rows to display.** In Summary view, make a selection from the Display list and click **Go** to specify the number of rows displayed.
- **Sort the statement report.** In Summary view, select a column heading to sort the listed values by that column.
- **Edit the script.** Click **Edit Script** to load the script into the Script Editor. See "Using the Script Editor" on page 18-5.

Exporting and Importing SQL Scripts

You can transfer scripts from your current Script Repository to a Script Repository in a different database user account by using the Export and Import tasks. Exported scripts are encoded in a single file on your local file system. Once exported, you then log in to other database user account and import the file. During import, the file is run to re-create the scripts in the current Script Repository.

By default, the Export SQL Scripts page lists all scripts in the Script Repository created by the current user. There are two panes on the Export SQL Scripts page, the Scripts pane and the Scripts to Export pane. You use the Scripts pane to select scripts to export. Then, you use the Scripts to Export pane to finalize the scripts to export, to choose a name for the export script, and to save the selected scripts in the export script on your local file system. You use the Import Scripts pane to select the export script containing the scripts to import.

Topics in this section include:

- Copying Scripts to an Export Script
- Importing Scripts from an Export Script

Copying Scripts to an Export Script

To copy scripts to an export script:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **SQL Scripts** icon.

   The SQL Scripts page appears.
3. On the Tasks list, click **Export**.

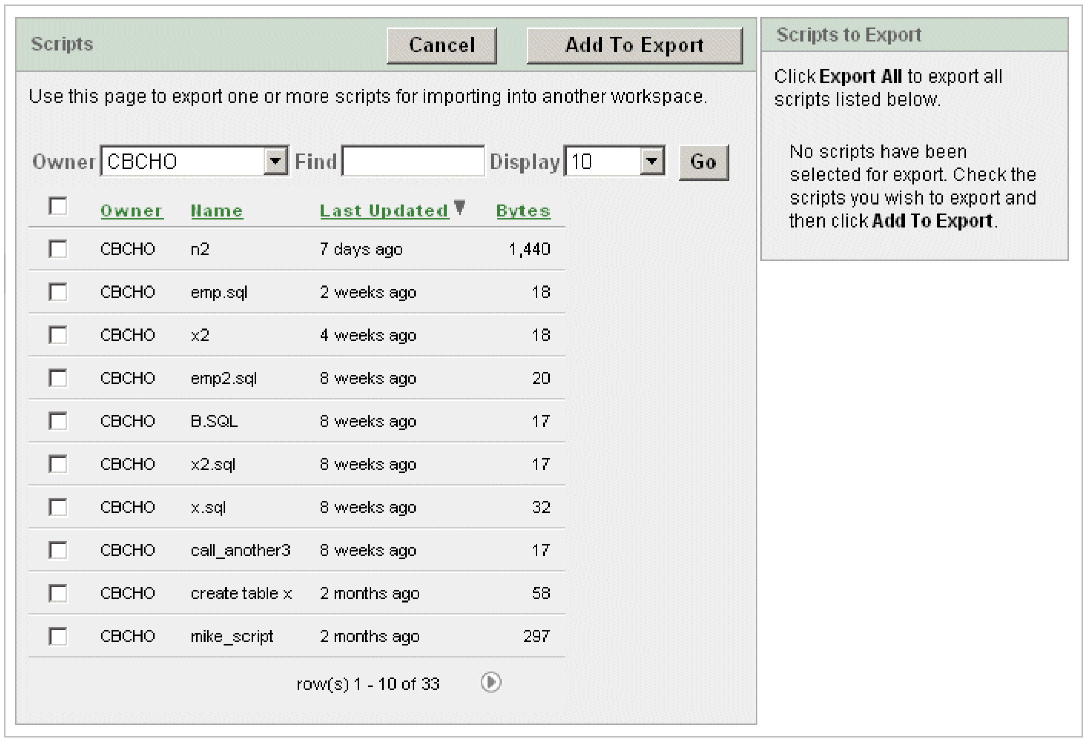
   The Export SQL Scripts page appears.
4. Click the check box for each of the scripts you want to export. The check boxes display on the left side adjacent to the script name. To select all displayed scripts for export, click the column head check box.
5. Click **Add to Export** to create a list of scripts to be added to the export script.

   The selected scripts are added to the list of scripts in the Scripts to Export pane.
6. Enter a name for the export script in the File Name field.

   The default script name is *workspace_name*_script.sql.
7. Click **Export All** to export the scripts to the export script.

   You are prompted to enter the directory where you want to save the export script.

About the Scripts Pane



Use this page to export one or more scripts for importing into another workspace.

Owner: CBCHO | Find: | Display: 10 | Go

☐	Owner	Name	Last Updated ▼	Bytes
☐	CBCHO	n2	7 days ago	1,440
☐	CBCHO	emp.sql	2 weeks ago	18
☐	CBCHO	x2	4 weeks ago	18
☐	CBCHO	emp2.sql	8 weeks ago	20
☐	CBCHO	B.SQL	8 weeks ago	17
☐	CBCHO	x2.sql	8 weeks ago	17
☐	CBCHO	x.sql	8 weeks ago	32
☐	CBCHO	call_another3	8 weeks ago	17
☐	CBCHO	create table x	2 months ago	58
☐	CBCHO	mike_script	2 months ago	297

row(s) 1 - 10 of 33

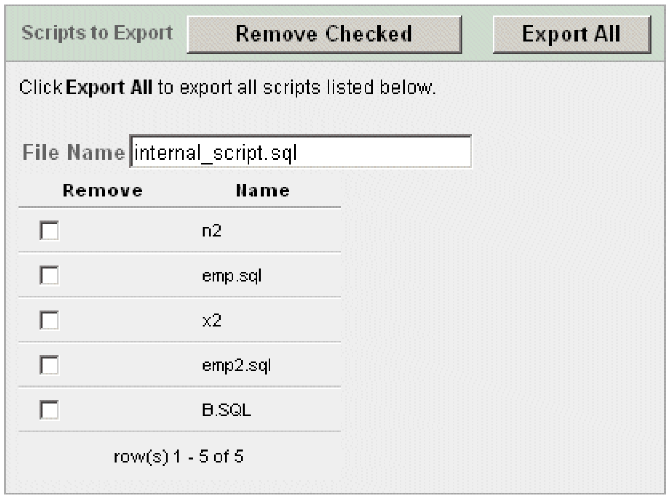
Click **Export All** to export all scripts listed below.

No scripts have been selected for export. Check the scripts you wish to export and then click **Add To Export**.

In the Scripts pane you can:

- **Search for a script.** Enter a script name or partial name in the Find field and click **Go**. To view all scripts, leave the Find field blank, select **- All Users -** from the Owner list and click **Go**. You control how many rows display by making a selection from the Display list.
- **Cancel the export.** Click **Cancel** to return to the SQL Scripts page without exporting any scripts, or to return to the SQL Scripts page after saving an export script.
- **Selecting scripts to export.** Click **Add to Export** to add scripts to the export script. Scripts added to the export script are no longer listed in the Script pane, but appear in the Scripts to Export pane.
- **Sort scripts.** Click a column heading to sort the listed scripts by that column.

About the Scripts to Export Pane



Click **Export All** to export all scripts listed below.

File Name: internal_script.sql

Remove	Name
☐	n2
☐	emp.sql
☐	x2
☐	emp2.sql
☐	B.SQL

row(s) 1 - 5 of 5

In the Scripts to Export pane you can:

- **Rename the export script.** Enter a name for the export script in the File Name field or leave the default script name.
- **Remove scripts.** Click the check box adjacent to the scripts you want to remove f and click **Remove Checked**. Scripts removed are no longer listed in the Scripts to Export pane, but appear in the Scripts pane.
- **Save the export script.** Click **Export All** to save the export script to your local file system. You are prompted to enter the directory where you want to save the export script.

Importing Scripts from an Export Script

To import scripts from an export script:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **SQL Scripts** icon.

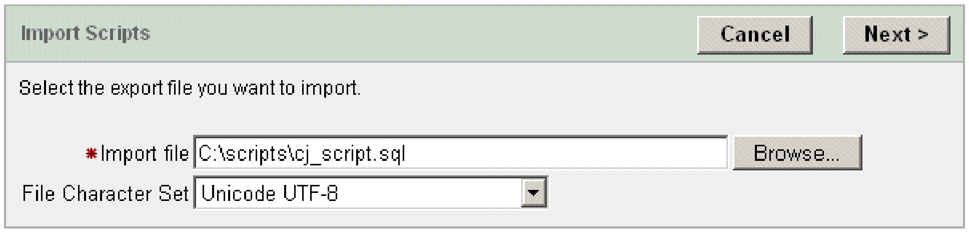
   The SQL Scripts page appears.
3. On the Tasks list, click **Import**.

   The Import Scripts pane appears. See "About the Import Scripts Pane" on page 18-14.
4. Enter the name and path to the export script you want to import to the Script Repository, or click **Browse** to locate the export script you want to import.
5. Click **Next** to list the scripts in the export script.

   The Action column indicates whether the imported script is new, or whether it will replace an existing script of the same name.
6. Click **Import Script(s)** to import the listed scripts into the current Script Repository.

   The SQL Scripts page appears listing the imported scripts.

About the Import Scripts Pane



Select the export file you want to import.

*Import file C:\scripts\cj_script.sql

File Character Set Unicode UTF-8

In the Import Scripts pane you can:

- **Enter the export script.** Enter the name and path of the script to import in the Import file field, or click **Browse** to locate the script.
- **Cancel the import.** Click **Cancel** to return to the SQL Scripts page without importing scripts.
- **Proceed with the import.** Click **Next** to import the scripts in the specified export script. You can review the listed scripts to import.
- **Choose another export file.** Click **Previous** to return to the Import Scripts file selection page to choose a different export script.
- **Import the scripts.** Click **Import Script(s)** to import the scripts contained in the export script.

Viewing Script and Result Quotas

You can view the script limits in the current database user account on the Script Quotas page.

To view the Script Quotas page:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **SQL Scripts** icon.

   The SQL Scripts page appears.
3. On the Tasks list, click **Show Quotas**.

   The Script Quotas page appears.
4. Click **OK** to return to the SQL Scripts page.

About the Script Quotas Page

The Script Quotas page displays the following limits:

- Result Quota in Bytes:
  - **Maximum Script Result Size.** The maximum size in bytes of a single script result.
  - **Quota for All Script Results.** The maximum size in bytes of all results in this database user account.
  - **Used.** The number of bytes currently used in this database user account.
  - **Free.** The number of bytes currently free in this database user account.
  - **Quota.** A usage bar illustrating the percentage of quota currently used.

- Script Quota in Bytes:
  - **Maximum Script Size.** The maximum size in bytes of a single script.

19 Using SQL Commands

This section provides information on how to use SQL Commands within Oracle Database Express Edition (Oracle Database XE) to create, edit, view, run, and delete SQL commands.

This section contains the following topics:

See Also: "Using SQL" in *Oracle Database Express Edition 2 Day Developer Guide*

What is SQL Commands?

You can use SQL Commands to create, edit, view, run, and delete SQL commands. A SQL command can contain SQL statements or PL/SQL blocks.

When using SQL Commands, remember the following:

- SQL commands created in the Query Builder can be accessed in SQL Commands.
- Saved SQL commands must have names unique within a given database user account.
- There is no interaction between SQL Commands and SQL Scripts.
- You can cut and paste a SQL command from SQL Commands to run in the SQL Script Editor.

See Also: "Using SQL Scripts" on page 18-1

Accessing SQL Commands

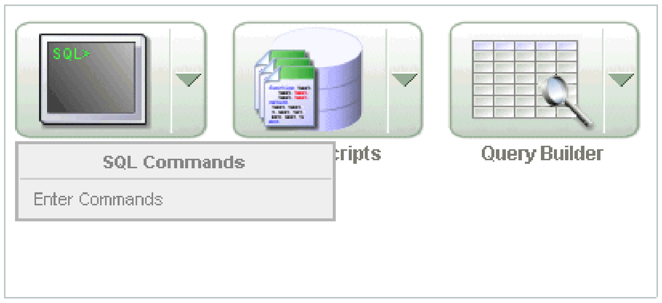
To access SQL Commands:

1. Log in to the Database Home Page.

   The Database Home Page appears.

2. To view the SQL Commands home page you can either:

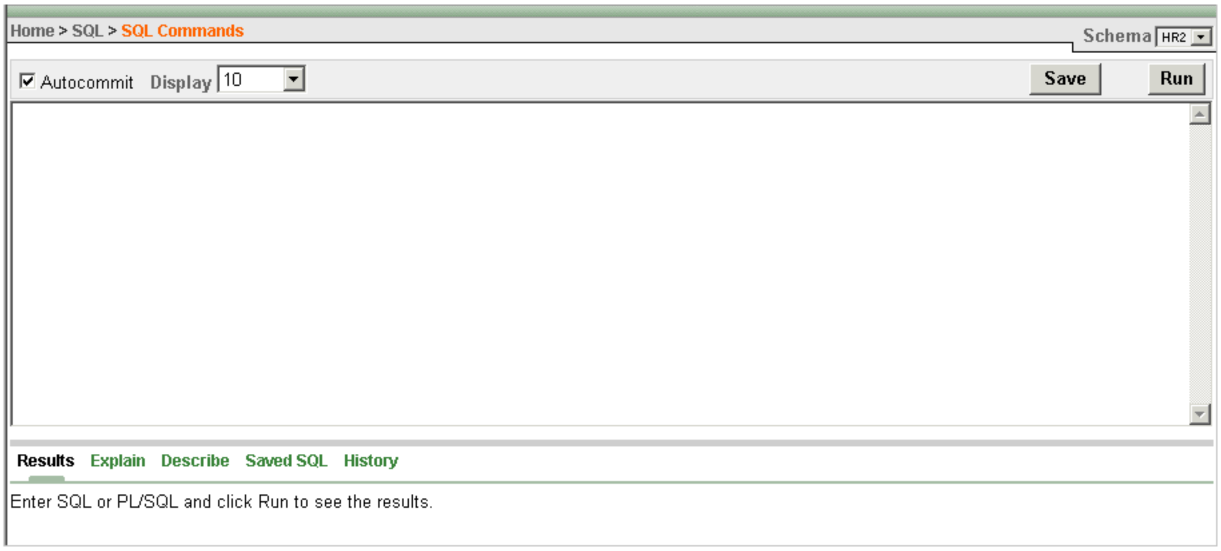
   - Click the **SQL** icon and then the **SQL Commands** icon to drill-down to the SQL Commands home page.
   - Click the down arrow on the right side of the SQL icon to view a drop down menu. Then select the **SQL Commands** menu option.



> **Note:** For the purposes of consistency, this document uses the primary navigation path (or drill-down approach) when explaining navigation.

About the SQL Commands Home Page

The SQL Commands home page contains a command editor and a display pane. You enter and edit SQL commands in the editor and view output, saved command lists, and history lists in the display pane.



Enter SQL or PL/SQL and click Run to see the results.

On the SQL Commands home page you can:

- **Disable transactional commands.** If available, click the **Autocommit** check box to enable autocommit and disable transactional commands. See "About Transactions in SQL Commands" on page 19-4.

- **Set the number of output rows.** Make a selection from the Display list to specify the number of rows of output to display at one time up to a maximum of 100,000. All rows of DBMS Output are displayed regardless of the Display list setting.
- **Save a SQL command.** Click **Save** to save the contents of the editor, or the currently highlighted content to a file. You are prompted to enter a name and an optional description. The new command appears in the Saved SQL list.
- **Execute a SQL command.** Click **Run** (or press Ctrl+Enter) to run the command in the editor, or the currently highlighted command in the editor.
- **Highlight an individual statement for execution.** Select an individual statement in the editor and click **Run** (or press **Ctrl+Enter)** to execute only the highlighted statement.

  The bottom of the SQL Commands home page features five tabs:

  – **Results.** Click the **Results** tab to see the results from the last successfully executed SQL command. Click **DBMS Output** at the bottom of the displayed results to display lines of DBMS output. This control only appears when there is DBMS output to display. Click **CSV Export** to export results to a comma separated file on your local file system. See "Viewing Results" on page 19-8.

  – **Explain.** Click the **Explain** tab to examine the execution plan used by the optimizer for statements that make changes to the database. Objects in the output are linked to the Object Browser. Click the linked object to view its properties in the Object Browser. See "Using Explain Plan" on page 19-9.

  – **Describe.** Enter Describe *object_name* and click **Run** to display column definitions for a table or view, or specifications for a function or procedure in the **Describe** tab. Select links in the Describe results to write that information into the command editor. For example, click a table name to add *owner.table*, click a column name to add the *column name*, click a procedure or function name to add the object call with parameters, or click a package name to add the package call.

  – **Saved SQL.** Click the **Saved SQL** tab to display a list of all SQL commands saved in the current database user account. Click the command title to load it into the command editor. See "Using Saved Commands" on page 19-5

  – **History.** Click the **History** tab to list your recently executed commands. Your last 200 executed commands are saved. See "Using SQL Command History" on page 19-7.

Executing a SQL Command

You use SQL Commands to execute SQL commands within Oracle Database XE.

Topics in this section include:

- Running a SQL Command
- About Transactions in SQL Commands
- About Unsupported SQL*Plus Commands
- About Command Termination
- Using Bind Variables

Running a SQL Command

To execute a SQL Command:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **SQL Commands** icon.
3. Enter the SQL command you want to run in the SQL editor pane.
4. Click **Run** (Ctrl+Enter) to execute the command.

   The results appear in the Results pane.

See Also: "Viewing Results" on page 19-8

About Transactions in SQL Commands

To disable transactional SQL commands in SQL Commands, check the Autocommit check box. Attempting to use any transactional SQL commands such as COMMIT or ROLLBACK when transactional mode is disabled returns an error message.

To enable transactional SQL commands, clear the Autocommit check box. Oracle Application Express verifies that the necessary system resources are available before entering the transactional mode. If resources are unavailable, an error message is displayed.

Transactional mode is a stateful transaction mode where you can, for example, perform an update, select data for review, and COMMIT or ROLLBACK changes. It is implemented using DBMS_JOBS.

Consider the following behavior in transactional mode:

- Actions are not committed to the database until you enter an explicit COMMIT command.
- Exiting SQL Commands terminates and rolls back the current transaction.
- A session time out terminates and rolls back the current transaction. The system preference, SQL_COMMAND_MAX_INACTIVITY, sets the time before an inactive session times out. The default timeout is 60 minutes.
- The **CSV Export** option is not available.

About Unsupported SQL*Plus Commands

SQL Commands does not support SQL Command Line (SQL*Plus) commands. If you attempt to enter a SQL Command Line command such as `SET ECHO` or `DEFINE` in SQL Commands, an error message displays.

About Command Termination

You can terminate a command in SQL Commands using a semicolon (;), a forward slash (/), or with nothing. Consider the following valid alternatives:

```
SELECT * from emp;

SELECT * from emp
/

SELECT * from emp
```

The first example demonstrates the use of a semicolon (;), the second example demonstrates the use of forward slash (/), and the final example demonstrates a command with no termination.

Using Bind Variables

Bind variables are supported. You are prompted to enter values for bind variables during command execution. Bind variables are prefixed with a colon.

For example

```
SELECT * from emp where deptno = :dept
```

Saving a SQL Command

You can save commands you enter in SQL Commands.

To save a SQL command:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **SQL Commands** icon.
3. Enter the command in the command editor.
4. Click **Save** to save the command.

   You are prompted to enter a name and description for the command.
5. Click **Save**, or click **Cancel** to return to the command editor without saving.

   The saved command is listed in the display area.

Copying a Command

To copy a SQL command:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **SQL Commands** icon.
3. Click the **Saved SQL** tab, located between the command editor and the display pane.

   The Saved SQL list of commands appears in the display pane.
4. Click the title of the command to load it into the command editor
5. Click **Save** to save the command.
6. Enter a new name for the command in the Name field and click **Save**.

   The command is copied to the new name.

Using Saved Commands

You can access the commands you save and commands saved by other users in the same database user account. You can also access SQL commands you and other users of the same database user account saved from the Query Builder.

Topics in this section include:

- Accessing Saved Commands

- About the Saved SQL Pane

Accessing Saved Commands

To access saved SQL commands:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **SQL Commands** icon.
3. Click the **Saved SQL** tab which is located between the command editor and the display pane.

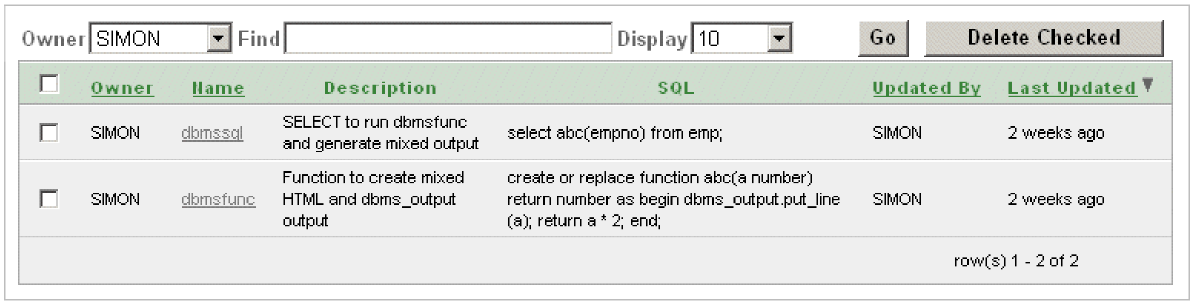
   The Saved SQL list of commands appears in the display pane.
4. Click the title of the command to load it into the command editor.

   The command appears in the editor.
5. Click **Run** to execute the command.

About the Saved SQL Pane

The Saved SQL pane displays a list of all commands saved under the current database user account. The list displays commands saved from SQL Commands and SQL commands saved from the Query Builder. Saved SQL commands must have unique names in the current database user account. The same name cannot be used in the Query Builder and SQL Commands.

Each command entry shows the owner name, the command name, the first characters of the SQL command, a description if it exists, who last updated the command and when.



Owner: SIMON Find: Display: 10

☐	Owner	Name	Description	SQL	Updated By	Last Updated
☐	SIMON	dbmssql	SELECT to run dbmsfunc and generate mixed output	select abc(empno) from emp;	SIMON	2 weeks ago
☐	SIMON	dbmsfunc	Function to create mixed HTML and dbms_output output	create or replace function abc(a number) return number as begin dbms_output.put_line (a); return a * 2; end;	SIMON	2 weeks ago

row(s) 1 - 2 of 2

On the Saved SQL pane you can:

- **Show commands by owner.** Make a selection from the Owner list to specify the user whose commands you want to display. To view all scripts select -All Users-.
- **Search for a command.** Enter a command name or partial name, or enter a code snippet in the Find field and click **Go**. To view all scripts, leave the Find field blank and click **Go**. You control how many rows display by making a selection from the Rows list.
- **Set the Number of Output Rows.** Make a selection from the Display list to specify the number of Saved SQL commands to display at one time.
- **Delete a command.** Click the check box associated with each command you want to delete, and click **Delete Checked**.
- **Sort commands.** Click a column heading to sort the listed commands by that column.

Using SQL Command History

Commands you have executed are stored in the command history regardless of whether you explicitly save them. You use SQL Command History to access commands you have executed in SQL Commands.

Topics in this section include:

- Accessing a Command from Command History
- About the History Pane

Accessing a Command from Command History

To access history commands:

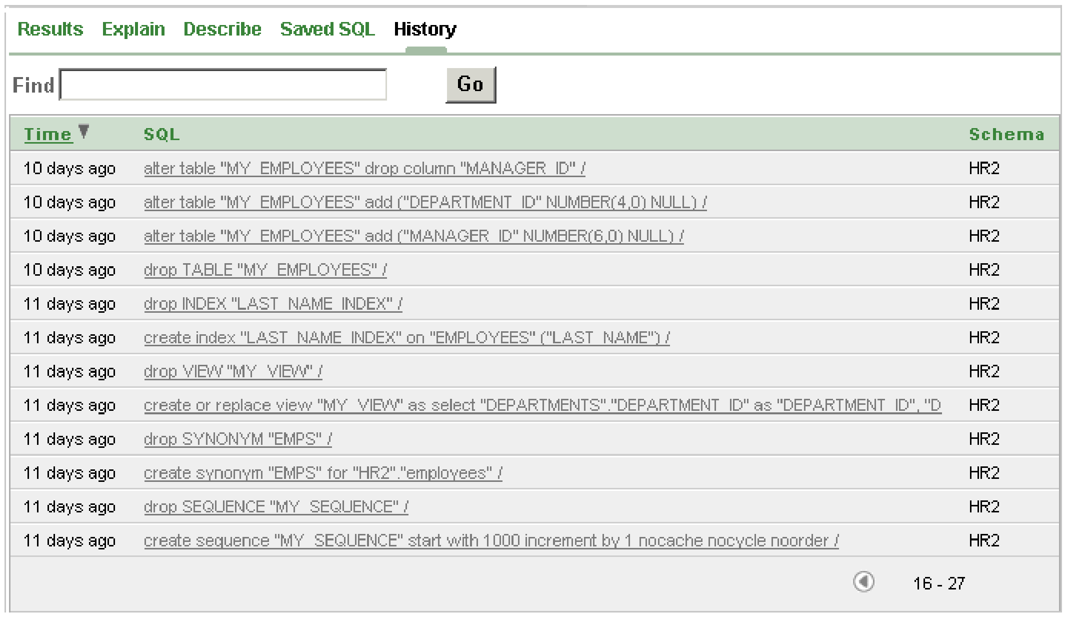
1. On the Database Home Page, click the **SQL** icon.
2. Click the **SQL Commands** icon.
3. Click the **History** tab, located between the command editor and the display pane.

   The list of commands in History appears in the display pane.
4. Click the partial command displayed in the SQL column.

   The command appears in the editor.

About the History Pane

The History pane displays a list of commands you have executed.



Time	SQL	Schema
10 days ago	alter table "MY_EMPLOYEES" drop column "MANAGER_ID" /	HR2
10 days ago	alter table "MY_EMPLOYEES" add ("DEPARTMENT_ID" NUMBER(4,0) NULL) /	HR2
10 days ago	alter table "MY_EMPLOYEES" add ("MANAGER_ID" NUMBER(6,0) NULL) /	HR2
10 days ago	drop TABLE "MY_EMPLOYEES" /	HR2
11 days ago	drop INDEX "LAST_NAME_INDEX" /	HR2
11 days ago	create index "LAST_NAME_INDEX" on "EMPLOYEES" ("LAST_NAME") /	HR2
11 days ago	drop VIEW "MY_VIEW" /	HR2
11 days ago	create or replace view "MY_VIEW" as select "DEPARTMENTS"."DEPARTMENT_ID" as "DEPARTMENT_ID", "D	HR2
11 days ago	drop SYNONYM "EMPS" /	HR2
11 days ago	create synonym "EMPS" for "HR2"."employees" /	HR2
11 days ago	drop SEQUENCE "MY_SEQUENCE" /	HR2
11 days ago	create sequence "MY_SEQUENCE" start with 1000 increment by 1 nocache nocycle noorder /	HR2

16 - 27

Each history entry shows the time the command was last executed, the first characters of the command, and the schema in which it was executed.

On the History pane you can:

- **Load a command.** Click the partial command displayed in the SQL column to load the command into the command editor. When the command loads, it also sets the schema in which it was last executed.

- **Sort by time.** Click the Time column heading to sort the command history by least recent or most recent.

Viewing Results

When you execute a SQL command, the results are displayed. The results of the last executed command are available until you execute another SQL command, or leave SQL Commands.

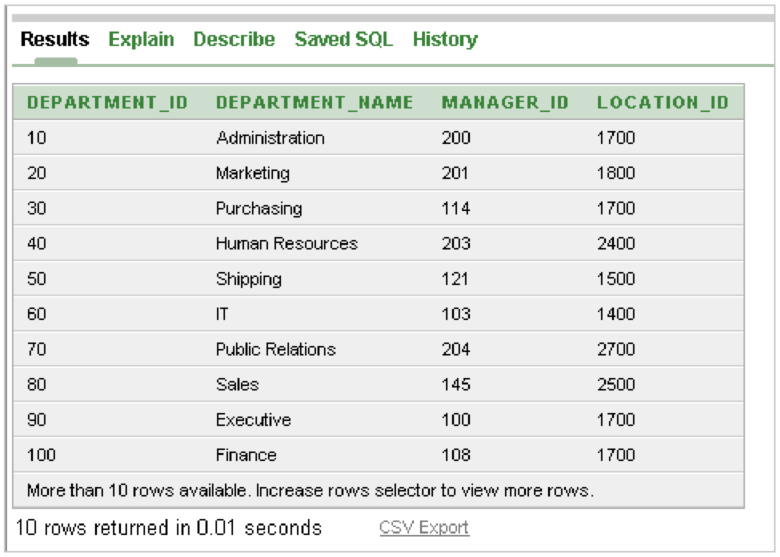
Topics in this section include:

- Accessing the Results Pane
- About the Results Pane

Accessing the Results Pane

To display SQL command results:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **SQL Commands** icon.
3. Click the **Results** tab, located between the command editor and the display pane.



DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.01 seconds CSV Export

The HTML formatted results appear in the display pane.

4. Click **DBMS Output** to display plain text DBMS output results.

   The **DBMS Output** control only appears if there is DBMS output in addition to HTML formatted results. It does not appear if there is only DBMS output, or if there is only HTML formatted output.

About the Results Pane

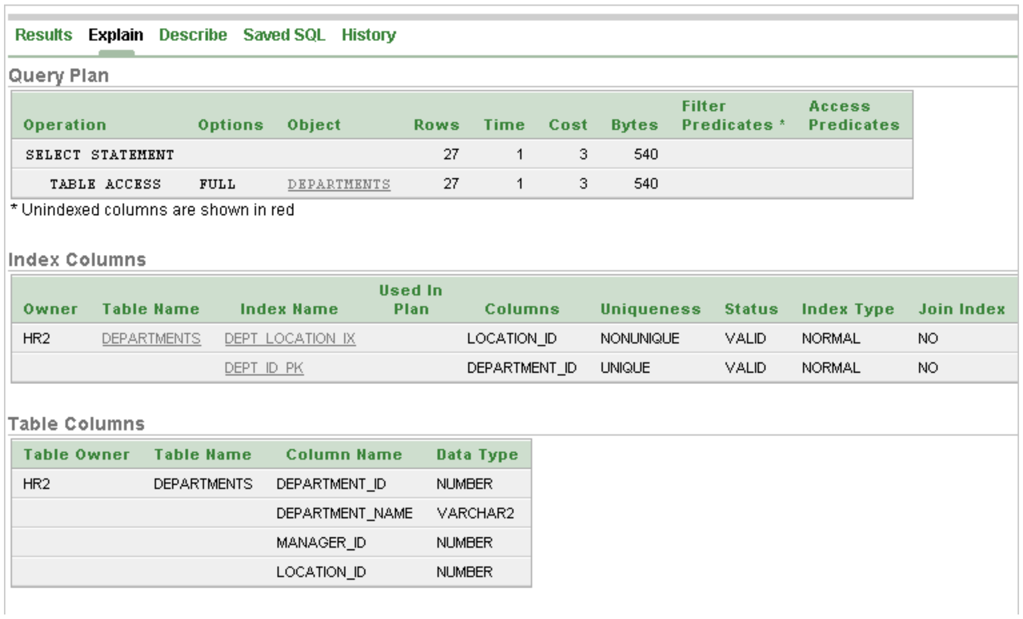
The Results pane displays SQL command results as HTML formatted table. The number of rows returned appears at the end of the output, and the time taken. DBMS output appears as plain text after the HTML formatted results.

On the Results pane you can:

- **Display DBMS output.** Click **DBMS Output** at the bottom of the displayed results to display lines of DBMS output. This control only appears when there is DBMS output to display.
- **Export results.** Click **CSV Export** to export results to a comma separated file on your local file system. You are prompted to enter a name and directory for the file.

Using Explain Plan

You can view the explain plan the Oracle Optimizer uses to run your SQL command. You do not need to execute the command to view the explain plan.



Query Plan

Operation	Options	Object	Rows	Time	Cost	Bytes	Filter Predicates *	Access Predicates
SELECT STATEMENT			27	1	3	540		
TABLE ACCESS	FULL	DEPARTMENTS	27	1	3	540		

* Unindexed columns are shown in red

Index Columns

Owner	Table Name	Index Name	Used In Plan	Columns	Uniqueness	Status	Index Type	Join Index
HR2	DEPARTMENTS	DEPT_LOCATION_IX		LOCATION_ID	NONUNIQUE	VALID	NORMAL	NO
		DEPT_ID_PK		DEPARTMENT_ID	UNIQUE	VALID	NORMAL	NO

Table Columns

Table Owner	Table Name	Column Name	Data Type
HR2	DEPARTMENTS	DEPARTMENT_ID	NUMBER
		DEPARTMENT_NAME	VARCHAR2
		MANAGER_ID	NUMBER
		LOCATION_ID	NUMBER

Topics in this section include:

- Viewing an Explain Plan
- About Explain Plan Pane

Viewing an Explain Plan

To view the Explain Plan:

1. On the Database Home Page, click the **SQL** icon.
2. Click the **SQL Commands** icon.
3. Enter or load the command whose plan you want to view.
4. Click **Explain** which is located between the command editor and the display pane.

   The explain plan used by the optimizer appears in the display pane.

About Explain Plan Pane

The Explain Plan pane shows the plan used by the Oracle Optimizer to run your SQL command. It typically displays the Query Plan, Index Columns and Table Columns used.

On the Explain Plan pane you can:

- **View object definitions.** Click the object name in Query Plan to display the object definition in the Object Browser.
- **View index definitions.** Click the index name in Table Columns to display the index definition in the Object Browser.

20 Using SQL Utilities

This section describes how to use SQL utilities to load and unload data from the Oracle Database Express Edition (Oracle Database XE) database, generate DDL, view object reports, and restore dropped database objects.

This section contains the following topics:

- Loading and Unloading Data from From the Database
- Generating DDL
- Viewing Object Reports
- Using the Recycle Bin to View and Restore Dropped Objects

Loading and Unloading Data from From the Database

You can load data into and unload data from the Oracle Database XE database using the Data Load/Unload page. Supported load formats include:

- Text file containing comma-delimited or tab-delimited data
- XML documents
- Spreadsheets

Supported unload formats include:

- Text such as comma-delimited or tab-delimited data
- XML documents

This section contains the following topics:

- Accessing the Data Load/Unload Page
- Loading Data
- Unloading Data
- Using Text Data Load Repository

See Also: "Importing, Exporting, Loading, and Unloading Data" in *Oracle Database Express Edition 2 Day DBA Guide*

Accessing the Data Load/Unload Page

To access the Data Load/Unload page:

1. On the Database Home Page, click the **Utilities** icon.
2. Click **Data Load/Unload**.

The Data Load/Unload page appears.

3. Click the appropriate icon to load data, unload data, or view the repository.

Loading Data

You can load data into the Oracle Database XE database in the following ways:

- Copy and paste data from a spreadsheet.
- Upload a spreadsheet file in a delimited format (such as comma-delimited (.csv) or tab-delimited).
- Upload a text file containing comma-delimited or tab-delimited data.

Topics in this section include:

- Loading a Text File
- Loading an XML Document
- Loading Spreadsheet Data

Loading a Text File

For files less than 30KB, you can copy and paste tab-delimited data directly into the Load Data Wizard. For files larger than 30KB, you must upload a separate file.

To load a text file:

1. On the Database Home Page, click the **Utilities** icon.
2. Click **Data Load/Unload**.
3. Click **Load**.
4. Click **Load Text Data**.

   The Load Data Wizard appears.
5. Under Load To, select either **Existing table** or **New table**.
6. Under Load from, select either **Upload file** or **Copy and paste**.
7. Follow the on-screen instructions.

Loading an XML Document

Oracle Application Express supports XML documents adhering to the Canonical XML specification.

To load an XML document:

1. On the Database Home Page, click the **Utilities** icon.
2. Click **Data Load/Unload**.
3. Click **Load**.
4. Click **Load XML Data**.

   The Load XML Data Wizard appears.
5. Follow the on-screen instructions.

Loading Spreadsheet Data

You can load spreadsheet data by either copying and pasting text, or by loading a file. To copy and paste text, the spreadsheet file must be less than 30KB. For files larger than 30KB, you can import the file in a delimited format (such as comma-delimited (.csv) or tab-delimited), upload the file, and then load the data into a new or existing table.

To load spreadsheet data:

1. On the Database Home Page, click the **Utilities** icon.
2. Click **Data Load/Unload**.
3. Click **Load**.
4. Click **Load Spreadsheet Data**.

   The Load Data Wizard appears.
5. Under Load to, select either **Existing table** or **New table**.
6. Under Load from, select either **Upload file** or **Copy and paste**.
7. Follow the on-screen instructions.

Unloading Data

You can use the Unload page to export the contents of a table to a text file or XML document.

Topics in this section include:

- Unloading a Text File
- Unloading to an XML Document

Unloading a Text File

Use the Unload to Text Wizard to export the contents of a table to a text file. For example, you could export an entire table to a comma-delimited file (.csv).

To unload a table to a text file:

1. On the Database Home Page, click the **Utilities** icon.
2. Click **Data Load/Unload.**
3. Click **Unload**.
4. Click **Unload To Text**.

   The Unload to Text Wizard appears.
5. Follow the on-screen instructions.

You select the schema and choose the table and columns to be exported. You can also specify the type of separator to be used to separate column values as well as whether column text strings are identified using single or double quotation marks.

Unloading to an XML Document

Use the Unload to XML Wizard to export the contents of a table to an XML document adhering to the Canonical XML specification.

To unload a table to an XML document:

1. On the Database Home Page, click the **Utilities** icon.
2. Click **Data Load/Unload**.
3. Click **Unload**.

4. Click **Unload to XML**.

   The Unload to XML Wizard appears.

5. Follow the on-screen instructions.

You select the schema and choose the table and columns to be exported.

Using Text Data Load Repository

Loaded text data files are stored in the Text Data Load Repository.

To access the Text Data Load Repository:

1. On the Database Home Page, click the **Utilities** icon.
2. Click **Data Load/Unload**.
3. Click **Repository**.
4. To filter the display, make a selection from the Show list and click **Go**.
5. To view information about a specific file, click the **View** icon.
6. To delete an imported file, select it and click **Delete Checked**.

Generating DDL

With Oracle Application Express, you can generate Data Definition Language (DDL) statements from the Oracle data dictionary. These scripts can be used to create or recreate database schema objects. The scripts can be generated to display inline or saved as a script file. You can generate the create scripts for all objects for a specific schema, specific object types, or specific objects.

To generate a DDL statement:

1. On the Database Home Page, click the **Utilities** icon.
2. Click **Generate DDL**.

   The Generate DDL page appears.

3. Click **Create Script**.

   The Generate DDL Wizard appears.

4. Select a database schema and click **Next**.
5. Define the object type:

   a. Output - Specify an output format. Select either **Display Inline** or **Save As Script File**.

   b. Object Type - Select the object types for which to generate DDL.

   c. To select object names for the selected object types, click **Next** and follow the on-screen instructions.

6. Click **Generate DDL**.

See Also:

- *Oracle Database SQL Reference* for information about DDL statements
- "The Data Dictionary" in *Oracle Database Concepts* for information about the data dictionary

Viewing Object Reports

SQL includes a variety to object reports to help you better manage the objects in your database.

Topics in this section include:

- Viewing All Objects Reports
- Accessing the Data Dictionary
- Viewing PL/SQL Reports
- Viewing Security Reports
- Viewing Details about the Tables in Your Database

Viewing All Objects Reports

Use the reports on the All Objects page to view all objects for the selected schema. Available reports include All Objects, Invalid Objects, Object Calendar, and Objects Counts by Type.

To access the reports available on the All Objects page:

1. On the Database Home Page, click the **Utilities** icon.
2. Click **Object Reports** and then **All Objects**.
3. Select one of the following reports:
   - **All Objects**. Sort objects by creation date as well as last DDL (data definition language). To filter the report, select a object type, specify an object name, and click Go.
   - **Invalid Objects**. View all invalid objects in the database by object type. To filter the report, enter an object name, select a object type, and click **Go**.
   - **Object Calendar**. View all objects in a calendar format based on the date each database object was created.
   - **Object Counts by Type**. View counts of database object types for the selected schema.
4. To filter the report, select a object type, specify an object name, and click **Go**.

Accessing the Data Dictionary

Each Oracle database has a data dictionary. An Oracle data dictionary is a set of tables and views that are used as a read-only reference about the database. For example, a data dictionary stores information about both the logical and physical structure of the database. A data dictionary also stores information about valid Oracle database users, integrity constraints for tables in the database, and the amount of space allocated for a schema object as well as how much of it is being used.

To browse the data dictionary:

1. On the Database Home Page, click the **Utilities** icon.
2. Click **Object Reports** and then **Data Dictionary**.

   The Data Dictionary appears, listing all the Oracle Data Dictionary views.
3. To filter the report, enter a query in the Search field and click **Go**.

   You can query for details about database objects in the Data Dictionary.

4. Click the **View** icon to display Data Dictionary Browser.

   The Data Dictionary Browser appears. Use this page to query the Oracle Data Dictionary for details about database objects.

5. On the Data Dictionary Browser page, select the appropriate views and click **Query**. To select all views, select **Check All** and then click **Query**.

   A report appears.

6. To create a new query, click **New Query** and repeat step 5.

7. To browse another view, click **Browse Another View** and repeat steps 4 and 5.

   See Also: *Oracle Database Concepts* for information about the data dictionary

Viewing PL/SQL Reports

PL/SQL reports enable you to view program unit arguments or unit line counts or search PL/SQL source code.

Topics in this section include:

- Viewing Program Unit Arguments
- Viewing Unit Line Counts
- Searching PL/SQL Source Code

Viewing Program Unit Arguments

Use the Program Unit Arguments report to view package input and output parameters.

To view the PL/SQL Arguments report:

1. On the Database Home Page, click the **Utilities** icon.
2. Click **Object Reports**, **PL/SQL**, and then **Program Unit Arguments**.
3. To filter the report, enter a query in Object Name and click **Go**.

Viewing Unit Line Counts

Use the Unit Line Counts report to view then number of lines of code for each object. Use this report to identify larger PL/SQL program units.

To view the Unit Line Counts report:

1. On the Database Home Page, click the **Utilities** icon.
2. Click **Object Reports**, **PL/SQL**, and then **Unit Line Counts**.
3. To filter the report, enter an object name and click **Go**.

Searching PL/SQL Source Code

Use the Search PL/SQL Source code page to search the text within your PL/SQL code. Use this report to find references to tables or functions you might be thinking of deleting. You can also use this page to locate code when you can only recall a code snippet.

To search for PL/SQL source code:

1. On the Database Home Page, click the **Utilities** icon.

2. Click **Object Reports**, **PL/SQL**, and then **Search PL/SQL Source**.
3. To filter the report:
    a. In Object Name, enter a query.
    b. In Text, enter the PL/SQL code you want to search for.
    c. Click **Go**.

Viewing Security Reports

Security reports enable you to see privileges granted on database objects owned by other schemas. You can also use these reports to view database roles and system privileges.

Topics in this section include:

- Viewing Role Privileges
- Viewing Object Grants
- Viewing Column Privileges

Viewing Role Privileges

Role Privileges report shows the database roles that have been granted to a selected schema. Roles are collections of various privileges.

To view Role Privileges:

1. On the Database Home Page, click the **Utilities** icon.
2. Click **Object Reports**, **Security**, and then **Role Privileges**.

    The Role Privileges report appears.

Viewing Object Grants

The Object Grants report identifies privileges granted from or to the selected database schema. Use this report to determine the privileges for an existing schema has as well understand what privileges have been granted from the selected schema to other schemas.

To view the Object Grants report:

1. On the Database Home Page, click the **Utilities** icon.
2. Click **Object Reports** and then **Security**.
3. Click **User Privileges** and then **Object Grants**.

    The Object Grants report appears.
4. From Schema, select the database schema owner.
5. To filter the report, make a selection from the Show list and click **Go**.

Viewing Column Privileges

The Column Privileges report identifies column privileges granted from or to the selected database schema. Use this report to determine the privileges for an existing schema as well as understand what privileges have been granted from the selected schema to other schemas.

To view the Column Privileges report:

1. On the Database Home Page, click the **Utilities** icon.
2. Click **Object Reports** and then **Security**.
3. Click **User Privileges** and then **Column Privileges**.

   The Column Privileges report appears.
4. From Schema, select the database schema owner.
5. To filter the report, make a selection from the Show list and click **Go**.

Viewing Details about the Tables in Your Database

You can view specific details about the tables within your database by accessing the reports available on the Tables page.

To view the reports available on the Tables page:

1. On the Database Home Page, click the **Utilities** icon.
2. Click **Object Reports** and then **Tables**.

   The Tables page appears.
3. Select a report to review. Available reports include:
   - Columns
   - Comments
   - Constraints
   - Statistics
   - Storage Sizes
   - No Primary Key
   - Unindexed
   - Unindexed Foreign Keys
4. To filter a report, enter search criteria in the fields provided and click **Go**.

Using the Recycle Bin to View and Restore Dropped Objects

You can use the Recycle Bin to view and restore dropped database objects. When you drop a table, the space associated with the table is not immediately removed. The Oracle database renames the table and places it and any associated objects in the Recycle Bin. You can recover objects in the Recycle Bin at a later time.

This section contains the following topics:

- Managing Objects in the Recycle Bin
- Emptying the Recycle Bin Without Viewing the Objects

Note: The Recycle Bin feature is only available if you are running with an Oracle 10g or later database.

See Also: "Backing Up and Recovering the Database" in *Oracle Database Express Edition 2 Day DBA Guide*

Managing Objects in the Recycle Bin

You can view objects in the Recycle Bin on the Dropped Objects page. Once you select an object and view the Object Details page, you can choose to purge the object or restore the object by clicking the appropriate button.

To view objects in the Recycle Bin:

1. On the Database Home Page, click the **Utilities** icon.
2. Click **Recycle Bin** and then **Dropped Objects**.

   The Dropped Objects page appears.
3. To filter the report, select an object type, enter the object name in the Original Name field, and click **Go**.
4. To view object details, click the object name.

   The Object Details page appears.
5. To restore the current object, click **Restore Object**.
6. To permanently delete the current object, click **Purge Object**.

Emptying the Recycle Bin Without Viewing the Objects

To empty the Recycle Bin without viewing the objects:

1. On the Database Home Page, click the **Utilities** icon.
2. Click **Recycle Bin**.
3. Click **Purge Recycle Bin**.

   The Purge Recycle Bin page appears.
4. Confirm your request by clicking **Purge Recycle Bin** again.

Part IV

Administration

Part IV describes common Oracle Database Express Edition (Oracle Database XE) administration tasks including changing your password, altering the login message, enabling other users to access the Oracle Database XE user interface, viewing database storage and memory, managing database user accounts, and monitoring the database.

Part IV contains the following chapter:

21 Administering the Database

This section describes how to administer the Oracle Database Express Edition (Oracle Database XE) database.

To access any Database Administration pages, you must have a database account that has been granted a DBA role.

This section contains the following topics:

About the Administration Page

Use the Administration page to administer Oracle Database XE. To access any Database Administration pages, you must have a database account that has been granted a DBA role.

The Administration page contains the following icons:

- **Storage**. Use the Storage page to monitor the amount of space available in your database, view tablespaces, view datafiles, compact storage, or view logging status. See "Monitoring Database Storage" on page 21-3.
- **Memory**. Use the Memory page to view or alter memory allocation for your database. See "Monitoring Memory" on page 21-8.
- **Database Users**. Use the Manage Database Users page to create, edit, or drop database users. See "Managing Database User Accounts" on page 21-10.
- **Monitor**. Click the **Monitor** icon to access reports that provide a database-wide view of the database sessions, system statistics, SQL statements, and long operations. See "Monitoring the Database" on page 21-15.

- **About Database**. View details about your Oracle Express Server database including the product version, installed options, currently used features, or National Language Support. See "Viewing Database Details" on page 21-18.

About the Tasks List

A Tasks list displays on the right side of the Administration page. This list contains the following links:

- **Change My Password**. See "Changing Your Password" on page 21-2.
- **Manage Login Message**. See "Changing the Login Message" on page 21-2.
- **Manage HTTP Access**. See "Managing HTTP Access to the User Interface" on page 21-19.

Changing Your Password

You can change your password by clicking **Change Password** on the Tasks list on the Administration page.

To change your password:

1. On the Database Home Page, click the **Administration** icon.
2. On the Tasks list, click **Change My Password**.

   The Change Password page appears.
3. In the Old Password field, enter your old password.
4. In the New Password field, enter a new password.
5. In the Confirm Password field, enter your new password again.
6. Click **Apply Changes**.

About Expired Passwords

Passwords can be set to expire when a database user account is created. When a password expires, the user is prompted to create a new one.

To change an expired password:

1. In the Old Password field, enter your old password.
2. In the New Password field, enter a new password.
3. In the Confirm Password field, enter your new password again.
4. Click **Apply Changes**.

See Also: "Creating a Database User" on page 21-12

Changing the Login Message

You can change the message that displays when users log in to Oracle Database XE by clicking **Manage Login Message** on the Tasks list on the Administration page.

To change the user interface login message:

1. On the Database Home Page, click the **Administration** icon.
2. On the Tasks list, click **Manage Login Message**.

3. If prompted, enter the appropriate administrator username and password and click **Login**.
4. From Login Message, select **Custom Message**.
5. In the Custom Message field, enter a new message.
6. Click **Apply Changes**.

Monitoring Database Storage

Because Oracle Database XE is limited to 4 GB of user data, you need to monitor the amount of space available in your database. Use the Storage page to view a summary of the amount of space available in your database, view tablespaces, view datafiles, compact the available storage, or view logging status.

> **Tip:** You can also navigate to the Storage page by clicking the **Storage** link on the Usage Monitor. See "About the Usage Monitor" on page 1-3

About the Database Size Limitation

Oracle Database XE data is limited to 4 GB of user data. However, the maximum amount of physical storage in the database is limited an overall size of 5 GB. This physical maximum size limit includes the system tablespace, but excludes rollback and temporary tablespace segments.

Topics in this section include:

- Monitoring the Space Available
- Reclaiming Space in the Database
- Viewing Tablespaces
- Viewing Datafiles
- Viewing Database Logging Status

> **See Also:** "Managing Database Storage" in *Oracle Database Express Edition 2 Day DBA Guide*

Monitoring the Space Available

You can view a summary of the amount of space allocated, potentially available, and allowable on the Storage page. If you notice your database is approaching the 4 GB limit, consider dropping unused objects to free up space, compacting the storage, or upgrading to Oracle Database Standard Edition or Enterprise Edition.

> **See Also:** "About the Database Size Limitation" on page 21-3 and "Reclaiming Space in the Database" on page 21-4

To access the Storage page, you must have a database account that has been granted a DBA role.

To access the Storage page:

1. On the Database Home Page, click the **Administration** icon and then **Storage**.
2. If prompted, enter the appropriate administrator username and password and click **Login**.

   The Storage page appears and displays the following:

- **Space Allocated**. Displays the amount of space in MB currently allocated to database user objects.
- **Available**. Displays the amount of space in MB currently available.
- **Physical Limit**. Displays the amount of space in MB available for database objects.
- **Percent Used**. Displays a percentage that indicates the space used in comparison with the maximum database size permitted.

See Also: "Managing Database Storage" in *Oracle Database Express Edition 2 Day DBA Guide* and "About the Usage Monitor" on page 1-3

Reclaiming Space in the Database

You can recover unused free space in the database on the Compact Storage page. Compacting storage is accomplished using a database job that runs in the background. This database job frees up unused spaces allocated for tables and reorganizes data for optimal space utilization.

To access the Storage page, you must have a database account that has been granted a DBA role.

To compact your database storage:

1. On the Database Home Page, click the **Administration** icon and then **Storage**.
2. If prompted, enter the appropriate administrator username and password and click **Login**.
3. On the Tasks list, click **Compact Storage**.

   The Compact Storage page appears and displays the following:

   - **Space Allocated**. Displays the amount of space in MB currently allocated to database user objects.
   - **Available**. Displays the amount of space in MB currently available.
   - **Physical Limit**. Displays the amount of space in MB available for database objects.
   - **Percent Used**. Displays a percentage that indicates the space used in comparison with the maximum database size permitted.
4. Click **Compact Storage**.

   The process may take several minutes.

Tip: You can also navigate to the Storage page by clicking the **Storage** link on the Usage Monitor. See "About the Usage Monitor" on page 1-3

See Also: "Managing Database Storage" in *Oracle Database Express Edition 2 Day DBA Guide* and "Monitoring the Space Available" on page 21-3

Viewing Tablespaces

A database consists of one or more tablespaces. A tablespace is a logical grouping of physical datafiles. Each tablespace consists of one or more physical datafiles or tempfiles.

You can view a list of the tablespaces in your database on the Tablespaces page. To access the Tablespaces page, you must have a database account that has been granted a DBA role.

Note: All user data is stored in the `USERS` tablespace. If you notice that the `USERS` tablespace is approaching the 4 GB limit, consider dropping unused objects to free up space, or upgrading to Oracle Database Standard Edition or Enterprise Edition.

To access the Tablespaces page:

1. On the Database Home Page, click the **Administration** icon and then **Storage**.
2. If prompted, enter the appropriate administrator username and password and click **Login**.

   The Storage page appears. See "Monitoring the Space Available" on page 21-3.
3. On the Tasks list, click **View Tablespaces**.

   The Tablespaces page appears and displays the following:

   - **Tablespace name**. Displays the tablespace name.
   - **Percent Used**. Displays a bar chart and percentage that indicates the space used in comparison with the maximum database size permitted.
   - **Allocated**. Displays the amount of space in MB currently allocated to database user objects.
   - **Used**. Displays the amount of space in MB currently used.
   - **Datafile**. Indicates the number of datafiles in the tablespace.
4. To view additional details, click the tablespace name.

   The Datafile page appears and displays the following information:

   - Tablespace name
   - File Name
   - Current size
   - Maximum size
   - Type of extent management (autoextensible or not)

   See Also: "Managing Database Storage" in *Oracle Database Express Edition 2 Day DBA Guide*

Viewing Datafiles

Datafiles are the operating system files that hold the data within the database. The data is written to these files in an Oracle proprietary format that cannot be read by programs other than an Oracle database.

You can view a list of datafile in your database on the Data Files page. To access the Data Files page, you must have a database account that has been granted a DBA role.

To access the Data Files page:

1. On the Database Home Page, click the **Administration** icon and then **Storage**.

2. If prompted, enter the appropriate administrator username and password and click **Login**.

   The Storage page appears. See "Monitoring the Space Available" on page 21-3.

3. On the Tasks list, click **View Data Files**.

   The Data Files page appears and displays the following:

   - Tablespace (name)
   - File Name
   - Size (MB)
   - Maximum Size (MB)
   - Auto Extensible

     Auto Extensible indicates the type of extent management (that is, autoextensible or not).

4. Use the controls at the top of the page to control the page display:

   - To search for a data file, enter a file name in the Search Data File Name field and click **Go**.
   - To view a specific tablespace, select a tablespace from the Storage list and click **Go**.
   - To control the number or rows that display, make a selection from the Display list and click **Go**.

     See Also: "Managing Database Storage" in *Oracle Database Express Edition 2 Day DBA Guide*

Viewing Database Logging Status

Every Oracle database includes two or more redo log files. You can view redo log files on the Database Logging page. A set of redo log files is collectively known as the redo log for the database. These log files function to record all changes made to data.

Archiving your redo log files is an effective way to protect your database from media failure.

See Also: "About Database Storage Structures" in *Oracle Database Express Edition 2 Day DBA Guide* for more information on redo log files

Topics in this section include:

- Accessing the Database Logging Page
- About Redo Log Files
- About the Flash Recovery Area
- About ARCHIVELOG Mode

Accessing the Database Logging Page

To access the Database Logging page:

1. On the Database Home Page, click the **Administration** icon and then **Storage**.
2. If prompted, enter the appropriate administrator username and password and click **Login**.

The Storage page appears. See "Monitoring the Space Available" on page 21-3.

3. On the Tasks list, click **View Logging Status**.

   The Database Logging page appears.

> **Tip:** You can also navigate to the Database Logging page by clicking the **Log Archiving** link on the Usage Monitor. See "About the Usage Monitor" on page 1-3

About Redo Log Files

Redo log files are the most crucial structure for database recovery. This set of files is collectively known as the redo log for the database. A redo log is made up of redo entries, which are also called redo records.

The primary function of the redo log is to record all changes made to data in the database. If an Oracle instance failure or operating system failure prevents modified data from being permanently written to the datafiles, the changes can be recovered from the redo log so that committed data updates are not lost.

> **See Also:** "About Database Storage Structures" in *Oracle Database Express Edition 2 Day DBA Guide* and *Oracle Database Administrator's Guide* for more details about redo log

About the Flash Recovery Area

Oracle Database XE stores database backups and archived redo logs in the flash recovery area. The flash recovery area is a directory, separate from the database itself, where recovery-related structures are stored.

The default configuration of Oracle Database XE stores the flash recovery area on the same disk as your database files. In this configuration, if a media failure occurs, you can lose both your database and your backups. For any database where data protection is essential, change the location of the flash recovery area so that it is stored on a different disk.

> **See Also :** "Setting Flash Recovery Area Location and Size" in *Oracle Database Express Edition 2 Day DBA Guide*

About ARCHIVELOG Mode

Oracle Database XE can be configured to be in **ARCHIVELOG** mode or in **NOARCHIVELOG** mode. In `ARCHIVELOG` mode, a background archiving process copies filled, inactive redo log files to the flash recovery area before they are reused. In `NOARCHIVELOG` mode, the Oracle Database XE is not configured to archive its redo logs.

To simplify the management of your database, log archiving is disabled by default. Oracle recommends that you enable `ARCHIVELOG` mode for optimal data protection.

> **See Also:**
>
> - "Enabling ARCHIVELOG Mode for Media Failure Protection" in *Oracle Database Express Edition 2 Day DBA Guide*
> - *Oracle Database Administrator's Guide* for more details about redo logs and log archiving

Monitoring Memory

Oracle Database XE has a 1 GB memory limit. Use the Memory page to view or alter the current memory allocation for your database. There are two types of memory that the Oracle instance allocates: System Global Area (SGA) and Program Global Area (PGA).

To access any Database Administration pages, you must have a database account that has been granted a DBA role.

> **Tip:** You can also navigate to the Memory page by clicking the **Memory** link on the Usage Monitor. See "About the Usage Monitor" on page 1-3

Topics in this section include:

- About System Global Area Memory
- About Program Global Area Memory
- Viewing Current Memory Utilization
- Configuring System Global Area Memory
- Configuring Program Global Area Memory

> **See Also:** "Managing Database Memory" in *Oracle Database Express Edition 2 Day DBA Guide* and "About the Usage Monitor" on page 1-3

About System Global Area Memory

The System Global Area (SGA) is a memory area that contains data shared between all database users such as buffer cache and a shared pool of SQL statements. SGA is allocated in memory when an Oracle Database XE instance is started. The SGA contains the following subcomponents:

- **Buffer Cache**. Functions as the buffer to store any data being queried or modified. All user processes connected to the database share access to the buffer cache. The buffer cache helps avoid repeated access from physical disk, a time consuming operation.
- **Shared Pool**. Caches operational information and code that can be shared among users. For example:
  - SQL statements are cached so that they can be reuse.
  - Information from the data dictionary, such as user account data, table and index descriptions, and privileges, is cached for quick access and reusability.
  - Stored procedures are cached for faster access.
- **Redo Log Buffer**. Improves performance by caching redo information (used for instance recovery) until it can be written at once and at a more opportune time to the physical redo log files that are stored on disk.
- **Large Pool**. Functions as an optional area for buffering large I/O requests for various server processes

> **See Also:** "Managing Database Memory" in *Oracle Database Express Edition 2 Day DBA Guide*

About Program Global Area Memory

The Program Global Area (PGA) is a memory area used by a single Oracle server process. A server process services a client's requests. Each server process has its own private PGA area, which gets created by Oracle Database XE when it starts the process. The PGA is used to process SQL statements and to hold logon and other session information.

Viewing Current Memory Utilization

To view current memory utilization:

1. On the Database Home Page, click the **Administration** icon.
2. Click **Memory**.
3. If prompted, enter the appropriate administrator username and password and click **Login**.

   The Memory page appears and displays the following:

   - **Maximum System Global Area (SGA)**. Displays Maximum SGA memory in MB. This size sets the SGA_MAX_SIZE database parameter. See "About System Global Area Memory" on page 21-8.
   - **Program Global Area (PGA)**. Displays PGA memory in MB. See "About Program Global Area Memory" on page 21-9.
   - **Current Configuration (SGA + PGA)**. Displays the sum of SGA and PGA memory in MB.
4. To configure SGA or PGA, click **Configure SGA** or **Configure PGA** on the Tasks list on the right side of the page.

Configuring System Global Area Memory

The System Global Area (SGA) is a memory area that contains data shared between all database users such as buffer cache and a shared pool of SQL statements. SGA is allocated in memory when an Oracle Database XE instance is started, and any change in the value will take effect at the next startup.

The amount of memory allocated to SGA directly affects the performance of your database. The SGA sizes are configured automatically when you install Oracle Database XE.

See Also: "Managing Database Memory" in *Oracle Database Express Edition 2 Day DBA Guide*

To alter SGA memory:

1. On the Database Home Page, click the **Administration** icon.
2. Click **Memory**.
3. If prompted, enter the appropriate administrator username and password and click **Login**.

   The Memory page appears. See "Viewing Current Memory Utilization" on page 21-9.
4. From the Task list, select **Configure SGA**.
5. In SGA Target, enter a new value in MB.

6. Click **Apply Changes**.

For SGA size changes, you must shut down and restart the database for the changes to take effect.

> **See Also:** "Starting Up and Shutting Down" in *Oracle Database Express Edition 2 Day DBA Guide*

Configuring Program Global Area Memory

The Program Global Area (PGA) is a memory area used by a single Oracle server process. The amount of memory allocated to the PGA directly affects the performance of your database. The PGA sizes are configured automatically when you install Oracle Database XE.

To alter PGA memory:

1. On the Database Home Page, click the **Administration** icon.
2. Click **Memory**.
3. If prompted, enter the appropriate administrator username and password and click **Login**.

   The Memory page appears. See "Viewing Current Memory Utilization" on page 21-9.
4. On the Tasks list, click **Configure PGA**.
5. In Aggregate PGA Target, enter a new value in MB.
6. Click **Apply Changes**.

> **See Also:** "Managing Database Memory" in *Oracle Database Express Edition 2 Day DBA Guide*

Managing Database User Accounts

You access Oracle Database XE using a database user account. Each database user account has a username and specifically defined the user's attributes, including a password for database authentication, privileges and roles, default tablespace for database objects, default tablespace for query processes, and tablespace quota.

To access any Database Administration pages, you must have a database account that has been granted a DBA role.

> **Tip:** You can also navigate to the Manage Database Users page by clicking the **Users** link on the Usage Monitor. See "About the Usage Monitor" on page 1-3

Topics in this section include:

- Viewing Database Users
- Creating a Database User
- Editing an Existing Database User
- Dropping a Database User

> **See Also:** "Managing Users and Security" in *Oracle Database Express Edition 2 Day DBA Guide*, "About the Usage Monitor" on page 1-3, and "Managing Application Express End Users" on page 8-11

Viewing Database Users

You edit an existing database user account on the Manage Database Users page.

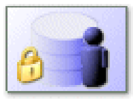
To edit an existing user account:

1. On the Database Home Page, click the **Administration** icon and then **Database Users**.
2. If prompted, enter the appropriate administrator username and password and click **Login**.

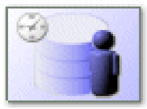
   The Manage Database Users page appears.

   By default each account displays as a large icon. The appearance of each icon indicates the current account status:

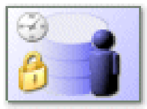
   - A small lock image indicates that the account is locked. Locking an account disables the database login.



   - A small clock image indicates that the account has an expired password. If an account has an expired password, the user will be prompted to select a new password upon the next login.



   - A small lock image and a clock image indicate that the account is both expired and locked.



   - Icons without a small lock or clock image are open and available to users.

   You can use the fields and buttons at the top of the page to search for a specific user or alter the page view.

3. To search for an existing database user:
   - To locate a specific user, type a username or partial string in the Search Username field and click **Go**.
   - To view all users, leave the Search Username field blank and click **Go**
4. To alter the types of users that display, make a selection from the Show list and click **Go**. Available options include:
   - All Users
   - Internal Users
   - Database Users
5. To change the page display, make a selection from the View list. View options include:

- **Icons** (the default) displays each database user account as a large icon identified by the username.
- **Details** displays each database user account as a line in a report. Each line includes the username, account status, lock date, expiration date, default tablespace, temporary tablespace, and how long ago the account was created.

Creating a Database User

When you create a database user, you also create a schema for that user. A schema is a logical container for the database objects (such as tables, views, triggers, and so on) that the user creates. The schema name is the same as the username and can be used to unambiguously refer to objects owned by the database user.

To create a database user account:

1. On the Database Home Page, click the **Administration** icon and then **Database Users**.
2. If prompted, enter the appropriate administrator username and password and click **Login**.

   The Manage Database Users page appears.
3. Click **Create**.

   The Create Database User page appears. Fields marked with a red asterisk (*) are required.
4. Specify user credentials:
   a. Username - Enter a username for database authentication.
   b. Password- Enter a password for database authentication.
   c. Confirm Password - Enter the password again.
5. Select **Expire Password** to prompt users to change their password the next time they log in.
6. Use Account Status to control whether a user can log into the account. Valid options include:
   - **Unlocked**. Select this option to enable a user to log in.
   - **Locked**. Select this option to prevent a user from logging in.

   Next, select a role for this user. A role is a group of related database privileges.
7. Under Roles, select the appropriate role:
   - `CONNECT` - Enables a user to connect to the database and perform basic database actions. Assign this role to any user or application that needs database access. `CONNECT` includes the system privilege `CREATE SESSION`.
   - `RESOURCE` - Enables a user to create schema objects. Assign this role only to developers and power users. RESOURCE includes the following system privileges: `CREATE CLUSTER`, `CREATE INDEXTYPE`, `CREATE OPERATOR`, `CREATE PROCEDURE`, `CREATE SEQUENCE`, `CREATE TABLE`, `CREATE TRIGGER`, and `CREATE TYPE`.
   - `DBA` - Enables a user to access the database objects of all other users and administer the database. Does not include the privileges to start up or shut down the database. DBA includes the system privilege `WITH ADMIN OPTION`.

To have more control over the privileges assigned to a user, you can create your own roles or augment the selected roles by selecting additional privilege.

8. Under Directly Grant Privileges, select additional privileges for the user. To select all privileges, click **Check All**. To deselect all privileges, click **Uncheck All**.

   Note that you can manage additional user attributes using SQL Commands. See "Using SQL Commands" on page 19-1.

9. Click **Create**.

   See Also: "Changing Your Password" on page 21-2 and "Administering User Privileges, Roles, and Profiles" in *Oracle Database Security Guide*

Editing an Existing Database User

To edit an existing user account:

1. On the Database Home Page, click the **Administration** icon and then **Database Users**.
2. If prompted, enter the appropriate administrator username and password and click **Login**.

   The Manage Database Users page appears.

   Use the fields and buttons at the top of the page to search for a specific user or alter the page view. See "Viewing Database Users" on page 21-11.
3. To edit account details, select a user.

   The User page appears.
4. To edit a user's password:
   a. Password- Enter a password for database authentication.
   b. Confirm Password - Enter the password again.
5. Select **Expire Password** to prompt users to change their password the next time they log in.
6. Use Account Status to control whether a user can log into the account. Valid options include:
   - **Unlocked**. Select this option to enable a user to log in.
   - **Locked**. Select this option to prevent a user from logging in.

   Use Roles to specify a group of related database privileges for this user.
7. Under Roles, select the appropriate role:
   - `CONNECT` - Enables a user to connect to the database and perform basic database actions. Assign this role to any user or application that needs database access. `CONNECT` includes the system privilege `CREATE SESSION`.
   - `RESOURCE` - Enables a user to create schema objects. Assign this role only to developers and power users. RESOURCE includes the following system privileges: `CREATE CLUSTER`, `CREATE INDEXTYPE`, `CREATE OPERATOR`, `CREATE PROCEDURE`, `CREATE SEQUENCE`, `CREATE TABLE`, `CREATE TRIGGER`, and `CREATE TYPE`.
   - `DBA` - Enables a user to access the database objects of all other users and administer the database. Does not include the privileges to start up or shut

down the database. DBA includes the system privilege `WITH ADMIN OPTION`.

To have more control over the privileges assigned to a user, you can create your own roles or augment the selected roles by selecting additional privilege.

8. Under Directly Grant Privileges, select additional privileges for the user. To select all privileges, click **Check All**. To deselect all privileges, click **Uncheck All**.

   Note that you can manage additional user attributes using SQL Commands. See "Using SQL Commands" on page 19-1.

9. To view current system privileges granted to the current user, click **All System Privileges Granted to** at the bottom of the page.

   A report appears.

10. Click **Alter User**.

See Also: "Changing Your Password" on page 21-2 and "Administering User Privileges, Roles, and Profiles" in *Oracle Database Security Guide*

Dropping a Database User

When you drop a database user, you also drop all objects owned by the user—that is, all objects in the user's schema.

To drip an existing user account:

1. On the Database Home Page, click the **Administration** icon and then **Database Users**.
2. If prompted, enter the appropriate administrator username and password and click **Login**.

   The Manage Database Users page appears.

3. You can locate an existing database user as follows:
   - To locate a specific user, type a username or partial string in the Username field and click **Go**.
   - To view all users, leave the Find User field blank and click **Go**
4. Select a user.

   The User page appears.

5. Click **Drop**.
6. Select **Cascade** and click **Drop User**.

   A report region appears displaying existing applications and scripts. Applications and scripts associated with the user's account will be deleted when the user account is dropped.

Note: In order to drop a database user whose schema contains database objects, you must select **Cascade**. An error will result if you do not select **Cascade** and the database user’s schema contains database objects.

Monitoring the Database

The reports available on the Database Monitor page provide a database-wide view of the database sessions, system statistics, SQL statements, and longer operations. You can use these reports to identify poorly preforming SQL and to gain a better understand the workload of the database.

To access any of the icons on the Database Monitor page, you must have a database account that has been granted a DBA role.

This section contains the following topics:

- Sessions
- About System Statistics
- About Top SQL
- About Long Operations

> **See Also:** "Monitoring the Database" in *Oracle Database Express Edition 2 Day DBA Guide*

Sessions

A session is the connection of a user to an Oracle database instance. A session lasts from the time the user connects until the time the user disconnects or exits the database application.

> **Tip:** You can also navigate to the Sessions page by clicking the **Sessions** link on the Usage Monitor. See "About the Usage Monitor" on page 1-3

You must have database administrator privileges in order to access the Sessions page.

To access reports on the Sessions page:

1. On the Database Home Page, click the **Administration** icon and then **Monitor**.
2. If prompted, enter the appropriate administrator username and password and click **Login**.
3. Click **Sessions**.

   The Sessions page appears.
4. To view a report, select one of the following tabs at the top of the page:
   - Sessions
   - Locks
   - Waits
   - I/O
   - SQL
   - Open Cursors

   The sections that follow describe each report.

> **See Also:** "About the Usage Monitor" on page 1-3

Sessions Report

The Sessions Report displays information about the current sessions in the database. Use the controls at the top of page to narrow the view:

- **Search**. Enter search criteria and click **Go**. For search details, click item help.
- **Status**. Select a status and click **Go**.
- **Show**. Select how many columns to display and click **Go**.
- **Display**. Select the number of rows to appear in the report and click **Go**.

To view session details, click the Session ID (SID). The Session Details page appears. To remove the current session, navigate to the Session Details page and click **Kill Session**.

Locks Report

The Locks report displays a report of sessions which have locks that are blocking other session(s). To control the number of rows that appear, make a selection from the Display list and click **Go**.

Waits Report

The Waits report displays the wait events for each session. Use the controls at the top of page to narrow the view:

- **Search**. Enter search criteria and click **Go**. For search details, click item help.
- **Status**. Select a status and click **Go**.
- **Show**. Select how many columns to display and click **Go**.
- **Display**. Select the number of rows to appear in the report and click **Go**.

To view session details, click the Session ID (SID). The Session Details page appears. To remove the current session, click **Kill Session**.

I/O Report

The I/0 report displays details about the I/O for each session. Use the controls at the top of page to narrow the view:

- **Search**. Enter search criteria and click **Go**. For search details, click item help.
- **Display**. Select the number of rows to appear in the report and click **Go**.

To view session details, click the Session ID (SID). The Session Details page appears. To remove the current session, click **Kill Session**.

SQL Report

The SQL report displays details about the current or last SQL statement executed for each session. Use the controls at the top of page to narrow the view:

- **Search**. Enter search criteria and click **Go**. For search details, click item help.
- **Status**. Select a status and click **Go**.
- **Show**. Select how many columns to display and click **Go**.
- **Display**. Select the number of rows to appear in the report and click **Go**.

To view session details, click the Session ID (SID). The Session Details page appears. To remove the current session, click **Kill Session**.

Open Cursors

The Open Cursors report displays details about the number of open cursors for each session. Use the controls at the top of page to narrow the view:

- **Search**. Enter search criteria and click **Go**. For search details, click item help.
- **Status**. Select a status and click **Go**.
- **Display**. Select the number of rows to appear in the report and click **Go**.

To view details about a specific open cursor count, click the numeric link under the Open Cursor Count column.

To view session details, click the Session ID (SID). The Session Details page appears. To remove the current session, click **Kill Session**.

About System Statistics

The System Statistics page displays statistics for:

- **Physical I/O**. A physical I/O is an I/O that requires disk access. This report displays disk access statistics for physical reads and writes.
- **Logical I/O**. An logical I/O is an I/O that is satisfied in memory or disk. Displays the sum of buffer reads which might be consistent gets or current mode gets. Redo is the buffer in the SGA that contains information about changes.
- **Memory Statistics**. Displays memory consumption of the database.
- **Time Statistics**. Shows various times consumed by the database.
- **SQL Cursor** Statistics. Displays statistics about the cursors in the Oracle database.
- **Transaction Statistics**. Shows the number of transactions performed.

To view the System Statistics page:

1. On the Database Home Page, click the **Administration** icon and then **Monitor**.
2. If prompted, enter the appropriate administrator username and password and click **Login**.
3. Click the **System Statistics** icon.

   The System Statistics page appears.

Additional controls on the System Statistics page include:

- **Refresh Report** - Refresh the System Statistics report.
- **Save Statistics** - Save the current report.
- **Show delta between current and saved values** - Click this check box to display actual statistic values, or display deltas between an saved value and the current value.

See Also: "Memory Configuration and Use" in *Oracle Database Performance Tuning Guide*

About Top SQL

The "top" SQL statements represent the SQL statements that are executed most often, that use more system resources than other SQL statements, or that use system resources more frequently than other SQL statements.

Use the Top SQL page to identify poorly performing SQL.

To view the Top SQL page:

1. On the Database Home Page, click the **Administration** icon and then **Monitor**.
2. If prompted, enter the appropriate administrator username and password and click **Login**.
3. Click the **Top SQL** icon.

   The Top SQL page appears.

   Use the search fields and lists and the top of the page and click **Go** to narrow the display. For details on each field or list, click item help
4. To access the SQL Plan page, click the **View** icon.

   The SQL Plan page appears.

   The SQL Plan page contains the following sections:

   - **Query Plan** - Contains a color coded explain plan. Note that unindexed columns display in red.
   - **SQL Text** - Displays the full text of the SQL statement.
   - **Indexes** - Displays all indexes on the table in the query. There is a checkmark when that index is used in the query.
   - **Table Columns** - Shows all columns on all tables or views in the query.

About Long Operations

The Long Operations page displays the status of various operations that run for longer than 6 seconds (in absolute time). These operations currently include many backup and recovery functions, statistics gathering, and query execution, and more operations are added for every Oracle release.

To view the Long Operations page:

1. On the Database Home Page, click the **Administration** icon and then **Monitor**.
2. If prompted, enter the appropriate administrator username and password and click **Login**.
3. Click the **Long Operations** icon.

   See Also: V$SESSION_LONGOPS" in *Oracle Database Reference*

Viewing Database Details

You can view details about your database on the About Database page.

To access details about your database:

1. Click the **Administration** icon.
2. Click **About Database**.
3. If prompted, enter the appropriate administrator username and password and click **Login**.

The About Database page appears. The About Database page is divided into two sections: Database and Version.

4. To view additional information about installed options, currently used features, or National Language Support, select one of the following check boxes and click **Go**:
   - Version
   - Database
   - Options
   - Feature Usage
   - National Language Support
   - CGI Environment

Managing HTTP Access to the User Interface

If you are using HTTP server, you can control access to the Oracle Database XE user interface on the Manage HTTP Access page. By default, only HTTP requests that originate from the same physical server where the database is installed are able to access the Oracle Database XE user interface. You can change this default behavior on the Manage HTTP Access page.

Note: The Manage HTTP Access page only controls access to the user interface if you are using HTTP server

To access the Manage HTTP Access page:

1. On the Database Home Page, click the **Administration** icon.
2. On the Tasks list, click **Manage HTTP Access**.
3. If prompted, enter the appropriate administrator username and password and click **Login**

   The Manage Access page appears.
4. Under Access, select one of the following:
   - **Available only from local server**. HTTP access is only allowed for HTTP requests that originate from the same physical server where the database is installed. This is the default setting.

     Selecting this option restricts access to the Oracle Database XE user interface as well as any other application using the HTTP server.
   - **Available to remote clients**. HTTP access to the user interface is enabled for all clients on your network that have a browser. Note that selecting this option may introduce additional security risks.
5. Click **Apply Changes**.

See Also: "Using Application Builder" on page 4-1 and "Deploying an Application" on page 11-1

About the Embedded PL/SQL Gateway

The embedded PL/SQL gateway provides Oracle Database XE with a Web server and also the necessary infrastructure to create dynamic applications. The embedded

PL/SQL gateway runs in the XML database HTTP server in the Oracle Database XE database. It includes the core features of mod_plsql, but does not require the Oracle HTTP server powered by Apache.

To establish connectivity from a browser via HTTP, the installation process creates database access descriptor (DAD) that is mapped to a virtual path. A DAD is a set of configuration values used to access the database and the virtual path mapping makes applications accessible under a virtual path of XML DB HTTP Server.

Topics in this section include:

- Restricting Access to Oracle Database XE

Restricting Access to Oracle Database XE

The embedded PL/SQL gateway supports a directive called `request-validation-function` which enables you to name a PL/SQL function which the embedded PL/SQL gateway will call for each HTTP request. You can use this functionality to restrict the procedures that can be invoked through the embedded PL/SQL gateway. The function returns true if the named procedure in the current request is allowed and false if it is not allowed. You can use this function to enforce access restrictions for Oracle Database XE on a per-database access descriptor (DAD) basis.

Oracle Application Express ships with a request-validation-function named `wwv_flow_epg_include_modules.authorize`. This function specifies access restrictions appropriate for the standard DAD configured for Oracle Application Express. During installation, scripts also name this function in the request-validation-function directive in the XDB configuration file.

During installation, the installer also creates a PL/SQL function in the Oracle Application Express product schema (`FLOWS_xxxxxx`). You can change and recompile this function in order to restrict access. The source code for this function is not wrapped and can be found in the database administrators product core directory in the file named `wwv_flow_epg_include_local.sql`. The source code is as follows:

```
CREATE OR REPLACE FUNCTION
wwv_flow_epg_include_mod_local(
    PROCEDURE_NAME IN VARCHAR2)
RETURN BOOLEAN
IS
BEGIN
    RETURN FALSE; -- remove this statement when
you add procedure names to the "IN" list
    IF UPPER(procedure_name) IN (
          '') THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
END wwv_flow_epg_include_mod_local;
/
```

To add names of procedures that should be allowed:

1. Remove or comment out the `RETURN FALSE` statement that immediately follows the `BEGIN` statement:

   ```
   ...
   BEGIN
       RETURN FALSE; -- remove this statement when
   ```

```
you add procedure names to the "IN" list
...
```

2. Add names to the clause representing procedure names that should be allowed to be invoked in HTTP requests. For example to allow procedures `PROC1` and `PROC2` the `IN` list you would write `IN ('PROC1', 'PROC2')`.

After changing the source code of this function, alter the Oracle Application Express product schema (`FLOWS_xxxxxx`) and compile the function in that schema.

To alter the product schema, `FLOWS_xxxxxx` :

1. Log in to SQL Command Line (SQL*Plus) as `SYS` or `SYSTEM`.
2. Alter the product schema (FLOWS_xxxxxx) by entering the following command:

   ```
   ALTER SESSION SET CURRENT_SCHEMA FLOWS_xxxxxx;
   ```

3. Compile the function `wwv_flow_epg_include_local.sql`.

> **See Also:** "Using SQL Command Line" in *Oracle Database Express Edition 2 Day Developer Guide*

The `wwv_flow_epg_include_mod_local` function is called by Oracle Application Express's request-validation-function which itself is called by the embedded PL/SQL gateway. The Oracle Application Express function first evaluates the request and based on the procedure name, approves it, rejects it, or passes it to the local function, `wwv_flow_epg_include_mod_local`, which can evaluate the request using its own rules.

When you create new DADs for use with Oracle Application Express through the embedded PL/SQL gateway, the request-validation-function directive should be specified exactly as it is for the DAD created when you install Oracle Database XE, that is, the function `wwv_flow_epg_include_modules.authorize` should be named in the directive.

If you have no additional restrictions beyond those implemented in the `wwv_flow_epg_include_modules.authorize` function, there is no need to take any action with respect to the source code for the `wwv_flow_epg_include_mod_local` function.

A Available Conditions

A condition is a small unit of logic that helps you control the display of regions, items, buttons, and tabs as well execute processes, computations and validations. When you apply a condition to a control or component, the condition is evaluated. Whether a condition passes or fails determines whether a control or component displays, or page processing executes.

You can specify conditions by selecting a condition type when you create the control (region, button, or item) or component (tab, list, or navigation bar), or by making a selection under the Condition attribute.

> **See Also:** "Understanding Conditional Rendering and Processing" on page 3-2

Table A–1 describes many Application Builder conditions. To view a complete listing of all available conditions for a given control or component, click the **View** icon to the right of the Condition Type list. Shortcuts to common selections appear directly beneath the Type list. If your condition requires an expression, type it in the appropriate field.

Table A–1 describes the conditions available in Application Builder.

Table A–1 Available Conditions

Condition	Description
Always	Always returns true. Used primarily for the read-only conditions of a page item
Current Language != Expression 1	Verifies the language setting in which the client browser is not currently running. Evaluates to true if the current language is contained within the string entered in Expression 1.
Current Language = Expression	Verifies the language setting in which the client browser is currently running. Evaluates to true if the current language matches the value entered in Expression 1.
Current Language is contained within Expression 1	Determines whether the browser current language is contained within a string. Evaluates to true if the current language matches the string entered in Expression 1. For example, to check if the current language is either en-US or en-GB, choose this condition and enter the following string in Expression 1: `en-us,en-gb`
Current Language is not contained within Expression 1	Verifies the application's current language is not contained within a specified string. Evaluates to true if the current language is not contained within the string entered in Expression 1.
Current page != Expression 1	Evaluates to true if the current page does not equal the page you specify in Expression 1.
Current Page != Page Submitted (this page was not the page posted)	Determines if the specified page was not posted. Evaluates to true if the current page does not match the value entered in Expression 1.

Table A–1 (Cont.) Available Conditions

Condition	Description
Current page = Expression 1	Evaluates to true if the current page equals the page you specify in Expression 1.
Current Page = Page Submitted (this page was posted)	Verifies the whether the specified page was posted. Evaluates to true if the current page matches the value entered in Expression 1.
Current Page is contained within Expression 1 (comma-delimited list of pages)	Verifies if the current page is part of the list of pages you specify in Expression 1. To check if the current page is in either page 1, 2, 3 or 4, select this condition type and enter the following string in Expression 1: `1,2,3,4`
Current page is in Printer Friendly mode	Only displays certain page control or components when the user has selected printer friendly mode. If the current page is in printer friendly mode, then the condition evaluates to true. Use `f?p` syntax to specify printer friendly mode.
Current page is not in Printer Friendly mode	Hides page controls or components when printer friendly mode is selected. Use `f?p` syntax to specify printer friendly mode. **See Also:** "Using f?p Syntax to Link Pages" on page 3-11 for information about `f?p` syntax
Current Page not in Expression 1 (comma-delimited list of pages)	Verifies if the current page is not part of the comma separated list of pages specified in Expression 1.
Exists (SQL query returns at least one row)	This condition is expressed as a SQL query. If the query returns at least one row then the condition evaluates as true. For example: `select 1 from emp where deptno = :P101_DEPTNO` This example references item `P101_DEPTNO` as a bind variable. You can use bind variables a within an application processes and SQL query regions to reference item session state. If one or more employees are in the department identified by the value of `P101_DEPTNO` then the condition evaluates as true. **See Also:** "About Bind Variables" on page 3-9 for information about bind variables
Never	This condition type is hard wired to always fail. It is useful in temporarily preventing controls or components (such as regions, buttons, or items) from being rendered on a page, or to prevent processes, computations and validations from running.
NOT Exists (SQL query returns no rows)	This condition is expressed as a SQL query. If the query does not return any rows, it evaluates as true.
PLSQL Expression	A PL/SQL expression is any expression in valid PL/SQL syntax that evaluates to true or false. For example: `nvl(:MY_FLOW_ITEM,'NO') = 'YES'` If the value of `MY_FLOW_ITEM` is YES then the condition evaluates as true. Otherwise it evaluates as false.

PLSQL Function Body returning a Boolean	The body of a PL/SQL function that returns true or false. For example:
	`BEGIN`
	`IF :P1_DAY = 'MONDAY' THEN`
	`    RETURN TRUE;`
	`ELSE`
	`    RETURN FALSE;`
	`END IF:`
	`END;`

Table A–1 (Cont.) Available Conditions

Condition	Description
Request != Expression 1	`REQUEST` is an internal attribute that tracks of how a page is submitted. By default, when a page is submitted, the value of the application attribute `REQUEST` is set according the name of the object that caused the page to be submitted. For a regular button, `REQUEST` is set as the name of the button (such as CANCEL or SAVE) not the label of the button. You can also set request using `f?p` syntax. For example, the event could be when a user clicks a button or selects a tab menu. Depending upon the event, you can perform different operations by referencing the value of the `REQUEST` application attribute. This condition evaluates as true if `REQUEST` does not equal the value entered in Expression 1. **See Also:** "Understanding URL Syntax" on page 3-10, "REQUEST" on page 3-21, and "Understanding the Relationship Between Button Names and REQUEST" on page 5-49
Request = Expression 1	This condition is the opposite of `Request != Expression 1`. This condition evaluates as true if `REQUEST` equals the value entered in Expression 1. From PL/SQL you can also reference the application attribute using the following syntax: `V('REQUEST')` **See Also:** "Understanding URL Syntax" on page 3-10, "REQUEST" on page 3-21, and "Understanding the Relationship Between Button Names and REQUEST" on page 5-49
Request is contained within Expression 1	`REQUEST` is an internal application attribute that tracks of how a page is submitted. By default, when a page is submitted, the value of `REQUEST` is set according to the event that caused the page to be submitted. For example, the event could be when a user clicks a button or selects a tab. Depending upon the event, you can perform different operations by referencing the value of the `REQUEST` application attribute. Use this condition to specify a list of allowed requests (such as SAVE or UPDATE) in Expression 1. The condition evaluates to true if the value of `REQUEST` is contained in the list. **See Also:** "REQUEST" on page 3-21, and "Understanding the Relationship Between Button Names and REQUEST" on page 5-49
Request is not contained within Expression 1	This condition is the opposite of `Request is contained within Expression 1`. Evaluates to true if the value of the `REQUEST` is not contained within Expression 1. **See Also:** "REQUEST" on page 3-21, and "Understanding the Relationship Between Button Names and REQUEST" on page 5-49
SQL Expression	SQL Expressions are evaluated as a WHERE clause in a SQL statement. For example suppose your expression is `:MY_ITEM = 'ABC'`. The Application Express engine processes the following: `select 1 from dual where :MY_ITEM = 'ABC'` This condition evaluates to true if a row is returned.
SQL Reports (OK to show the back button)	Use this condition for reports having pagination. It automatically determines when it is appropriate to include a button that pages back in the result set.
SQL Reports (OK to show the forward button)	Use this condition for reports having pagination. It automatically determines when it is appropriate to include a button that pages forward in the result set.
Text in Expression 1 != Expression 2 (includes &amp;ITEM substitutions)	Use this expression to compare two expressions containing strings. Either expression may contain references to session state using `&MY_ITEM` syntax. **See Also:** "Using Substitution Strings" on page 3-13 for information about `&MY_ITEM` syntax

Table A–1 (Cont.) Available Conditions

Condition	Description
Text in Expression 1 = Expression 2 (includes &amp;ITEM substitutions)	This condition is the opposite of `Text in Expression 1 != Expression 2` (includes `&amp;ITEM` substitutions). Compares two expressions containing strings. Either expression may contain references to session state using the `&ITEM.` syntax. To check if the item `F100_P2_DAY_DATE` equals "Wednesday", select this condition enter the following in Expression 1 and Expression 2: ■ Expression 1: `F100_P2_DAY_DATE` ■ Expression 2: `Wednesday` **See Also:** "Using Substitution Strings" on page 3-13 for information about `&MY_ITEM` syntax
User is authenticated (not public)	Verifies whether the current user was authenticated using one of the built-in authentication schemes or a custom authentication scheme. **See Also:** "Establishing User Identity Through Authentication" on page 12-14 for information about authentication
User is the public user (user has not authenticated)	The public user is defined as an application attribute. To set the public user for a specific application, navigate to the Application Builder home page and click the edit link corresponding to your application. A public user is a user used for multiple users. Sometimes applications have pages that are public and thus require authentication and log in. This condition returns true if the user is the public user (that is, the user is authenticated as themselves or some other user not equal to the public user identified in the application attribute Public User. **See Also:** "Authentication" on page 4-12 for information about Public User
Value of Item in Expression 1 != zero	Verifies if the value of the item in Expression 1 does not equal zero.
Value of item in Expression 1 = Expression 2	Compares the value of an item with a specific string. Comparisons using this condition are case-sensitive. For example, to verify whether the value of an item `F100_P2_WORD` is contained within the string "the quick brown fox", enter the following in the Expression 1 and Expression 2 fields: ■ Expression 1: `F100_P2_WORD` ■ Expression 2: `the quick brown fox`
Value of Item in Expression 1 = zero	Verifies if the value of the item in Expression 1 does equal zero.
Value of item in Expression 1 contains no spaces	Evaluate to true if the value of the item specified in Expression 1 contains no spaces.
Value of Item in Expression 1 is alphanumeric	Evaluates to true when the string in Expression 1 contains only alphanumeric characters.
Value of Item in Expression 1 is contained within colon-delimited list in Expression 2	Use this condition type to check whether a certain string is contained within the value of a session state item. Verifies whether the string specified in Expression 1 is contained in the value of the item specified in Expression 2.
Value of Item in Expression 1 is NOT contained within colon-delimited list in Expression 2	Evaluates to true when the value specified in Expression 1 contains a string that lists elements delimited by colons. To check if the item `P1_TODAY` is either "Monday", "Tuesday", or "Wednesday", select this condition and enter the following in Expression 1 and Expression 2: ■ Expression 1: `P1_TODAY` ■ Expression 2: `Monday: Tuesday:Wednesday`
Value of Item in Expression 1 is NOT NULL	In Expression 1, enter the name (uppercase) of the application or page item. Evaluates as true, if the current cache value of the item is not null and has a value. If not, the condition evaluates as false.
Value of Item in Expression 1 is NULL	Evaluates as true if the item in Expression 1 has no value.
Value of Item in Expression 1 is NULL or zero	Evaluates as true if the item in Expression is either NULL or zero.
Value of item in Expression 1 is numeric	Evaluates to true if the value of the Item in Expression 1 is numeric.
Value of user preference in Expression 1 != Expression 2	This condition is the opposite of `Value of user preference in Expression 1 = Expression 2`. Evaluates to true if the name of the user preference specified in Expression 1 is not equal to the string in Expression 2.

Table A–1 (Cont.) Available Conditions

Condition	Description
Value of user preference in Expression 1 = Expression 2	Verifies the value of a user preferences. Evaluates to true if the name of the user preference specified in Expression 1 is equal to the string in Expression 2.
When any item in comma-delimited list of items has changed	Evaluates to true when the value of any nonnull session state item in the list of items specified in Expression 1 has changed.
When any item in comma-delimited list of pages has changed	Evaluates to true when the value of any nonnull session state item in the list of pages specified in Expression 1 has changed.
When any item in current application has changed	This condition passes when the value of any nonnull session state item in the current application has changed.
When any item in current page has changed	Evaluate to true when the value of any nonnull session state item in the current page has changed.
When any item in current session has changed	Evaluates to true when the value of any nonnull session state item in the current session has changed.
When cgi_env DAD_NAME != Expression 1	This condition is the opposite of `When cgi_env DAD_NAME = Expression 1`. Checks for the data access descriptor (DAD) that is being used in the URL to call the current page in the application and compares it to Expression 1. Evaluate to true, when the DAD is not the same as Expression 1.
When cgi_env DAD_NAME = Expression 1	Checks for the data access descriptor (DAD) that is being used in the URL to call the current page in the application and compares it to Expression 1. Evaluate to true, when the DAD is the same as Expression 1.
When cgi_env HTTP_HOST != Expression 1	This condition is the opposite of `When cgi_env HTTP_HOST = Expression 1`. Checks for the value of the common gateway interface (CGI) environment variable `HTTP_HOST` that is the value returned by `owa_util.get_cgi_env ('HTTP_HOST')`. Evaluate to true, when this value is not equal to the string in Expression 1.
When cgi_env HTTP_HOST = Expression 1	Checks for the value of the common gateway interface (CGI) environment variable `HTTP_HOST` that is the value returned by `owa_util.get_cgi_env ('HTTP_HOST')`. Evaluate to true, when this value is equal to the string in Expression 1.
When cgi_env SERVER_NAME != Expression 1	This condition is the opposite of `When cgi_env SERVER_NAME = Expression 1`. This condition checks for the value of the common gateway interface (CGI) environment variable `SERVER_NAME`, that is the value returned by `owa_util.get_cgi_env ('SERVER_NAME')`. Evaluate to true, when this value is not equal to the string in Expression 1.
When cgi_env SERVER_NAME = Expression 1	This condition checks for the value of the common gateway interface (CGI) environment variable `SERVER_NAME`, that is the value returned by `owa_util.get_cgi_env ('SERVER_NAME')`. Evaluate to true, when this value is equal to the string in Expression 1.

Index