Class Note 5
PHP Oracle Development
(Updated 6/3/2015)

Functions and HTML Form

The "class note" is the typical material I would prepare for my face-to-face class. I am sharing the notes with everyone assuming you are in the class.

For the best result of your study, please make sure you do the following:

1. Read the chapter(s) in your textbook, corresponding to the week.

2. Read this class note. Please understand that I will only highlight the important points from your chapters in the textbook. I do not intend to repeat it.

3. Study the "Review Questions" for each week. I intend to combine the chapter content and class notes in the "Review Questions" section. Many of the concepts can be better understood if you fully understand the "Review Questions."

Computers are meant to provide us with speed and efficiency. In order for us to gain the best efficiency, most likely, you do not want to repeatedly write the same code for 200 times in your system. In case you will have to make any changes in the future, you do not want to make the same changes 200 times. Chances are you will miss some of those needed changes. If you think you want to repeat the same laborious work, you may stop at this point and go back to your "hard work." The class is meant for those who want to be efficient and do interesting work instead of repetitive tasks.

To avoid rewriting the same code again and again, you generally have three weapons, as follows:

1. Use "include" function to include your PHP program you have already developed in a different program. The content of "include" function was discussed in your textbook, on pages 79-81.

2. Use the functions that come with PHP language. You have seen and used some of the those functions such as "echo," "print" or "printf" and so on. We will not cover those functions directly in this chapter.

3. Define and use your own (user-defined) functions. This will be the focal point of this chapter.

I. File Inclusion:

We will start the "include."

Suppose you want to define some variables for your software such as database connection we will deal with later in this course, and you expect to use the same variables in many files in your software system, you may have one file that defines those variables. One simple example will be something in the following code (named as variables.php).

```
<?php
//Define variables that will be used by many files.
$product='Oracle';
$field='database';
?>
```

Then, our challenge is to use the parameters whenever we need them. In other words, when we write another PHP program, we have to be able to include those parameters from the file (variables.php). Now, you know this already, we may accomplish this mission by using "include" function. Below is a simple code (named as file_inclusion.php):

```
<?php
$product='';
$field='';



//Before "include"

echo "The output before \"include\" function is<br>";

echo "An $product  $field <br><br>";



include 'variables.php';

echo "The output after \"include\" function is<br>";

echo "An $product  $field <br>";

?>
```

Here, we are demonstrating the changes made with the help of "include" function. Before the "include," the two variables ($product, $field) were defined as blanks. Then, after the "include," the two variables were defined as intended ("Oracle" and "database" respectively). Please test the code, and you will see what I meant.

Please note:

1. The “include” function can include variables such as those in the above example, it can also include actions or output. For example, you may put “echo” statement in the “variables.php” file. When the “variables.php” is included, “echo” command will be executed.

2. The “include” function can be placed anywhere (beginning, middle or at the end) of the PHP program. In many practical applications, it is placed at the beginning.

3. The “include” function is tremendously useful when we connect our applications (PHP programs) to a database since we have to define parameters for the database connection. Please keep this in mind. I will test you to see if you understand what I am saying.

II. User-Defined Function.

I will start with simple code, as following:

```
<?php

//Define the variables for PHP program
$username='Brenda';
$password='bikeforfun';

//call the function "ping_function"
ping_function($username, $password);

//The following defines the function "ping-function".
//The simple function was created to show the basic concepts of defining and calling a
//function.



function ping_function ($user,$pwd)
{
print "[User Name]:  &nbsp&nbsp ".$user."<br/>";
print "[Password]:  &nbsp&nbsp ".$pwd."<br/>";
}
?>
```

I will start with the last block of code, in which we defined the function called “ping_function()”. The two parameters were defined as “$user” and “$pwd,” which are to be used within the “ping_function()”. You can see now what the function does is to print the user name and password.

We will then go back to the main program. As the first step, we defined the two variables ($username and $password). Then, we simply call the user-defined function "ping_function()." When we call the function, we use the two variables we previously defined. Then, the function did what is supposed, using the two variables passed to the function.

I would strongly recommend that you test the above code, perhaps add some actions in the function, change the names for variables, just to experiment it. You will learn the idea quickly if your mind and hands are involved.

For example, what if you add the following statement after the first call of "ping_function()"?

```
ping_function('Mike','baseball');
```

In other words, you may see different way to define variables.

1. The above function is very simple. You may argue "I do not need the function to perform the task." In this simple case, yes, you are right. You can easily do those prints within your main program. But, imagine if you are dealing with a sophisticated face recognition task, in which you have to capture the image, convert it into face template (a complicated mathematical operation), and compare it with the existing template. The point is that you would be lost easily or overwhelmed if you do not use functions.
2. In theory, the function definition block can be placed anywhere in the PHP program. In practice, for all professional-grade programs, the functions are always placed at the end of the program. In this way, you can easily see the logical flow in your main program.

III. HTML Form:

In the above PHP code, we have defined the variables ($username and $password) within the PHP program. For web applications, those values can come from a HTML form. Typically, a HTML form can be used to interact with the user, and then pass the values as the user input into your PHP program for process.

Since detailed HTML coding is beyond this course, I will use a simple HTML form to illustrate the point. That is all we require for this course anyway. Let us take a look at the following bare-bone HTML code:

```
<html>
<form name="myInputForm" action="php_action.php" method="post">
User Name: <input type="text" name="username" /><br />
Password: <input type="password" name="pwd" /><br />
<input type="submit" value="Submit" /><br />
</form>
</html>
```

I am going to focus on the bare minimum of HTML form. It does not look very pretty as you tested the code. You may add <table> tags to make the fields look better. But, the above code will do the technical tricks.

A HTML form starts with the tag <form> and ends with a </form> tag.

Within the <form name="myInputForm" action="php_action.php" method="post"> tag:

The name of the form is called myInputForm. After the “Submit” button is clicked by the user, the form will invoke the action, which will execute php_action.php. The way of data passing from the HTML to the server is by “post” method. You may research to find more details on the “post” method.

The first input field (User Name) is defined by <input type="text" name="username" /> tag. Here the input type is “text” and the name of the input variable is called “username.” With the text type, you will be able to see what you type in the computer.

The second input field (Password) is defined by <input type="password" name="pwd" /> tag. Here the input type is “password,” with which you will see only ****** when you type in your password.

The submit button is defined by <input type="submit" value="Submit" /> tag.

What will happen is: After the user click the “Submit” button, the form will take the user to php_action.php file.

Now, I am going to modify the first PHP program we have seen on this class as php_action.php file as follows.

```
<?php
//Get the values of the variables from the HTML form.
$username=$_POST["username"];
$password=$_POST["pwd"];

//call the function "ping_function"
ping_function($username, $password);

//The following defines the function "ping-function".
//The simple function was created to show the basic concepts of defining and calling a function.
```

```
function ping_function ($user,$pwd)
{
print "[User Name]:  &nbsp&nbsp ".$user."<br/>";
print "[Password]:  &nbsp&nbsp ".$pwd."<br/>";
}
?>
```

What I did was to simply use the PHP function of “$_POST” to get the values of the two variables passed by the form.html, after the user clicks the “Submit” button.

The typical syntax is:

$username=$_POST["username"];

where the username within the quotations has to be the same as the name in the HTML form.

The above linkage between “form.html” (user input through HTML form) and php_action.php (invoked action by the HTML form) illustrated the basic interaction between web forms and PHP program. The PHP program can be linked with the Oracle database.

During our class meeting on June 13, 2015, a team consisting of Rishank Chandra Puram, Raveendranath Thota and Divya Kumar Revindra came up with a code that combine the HTML form with the PHP action file. The PHP file was named as if_team4.php, as shown in the following block:

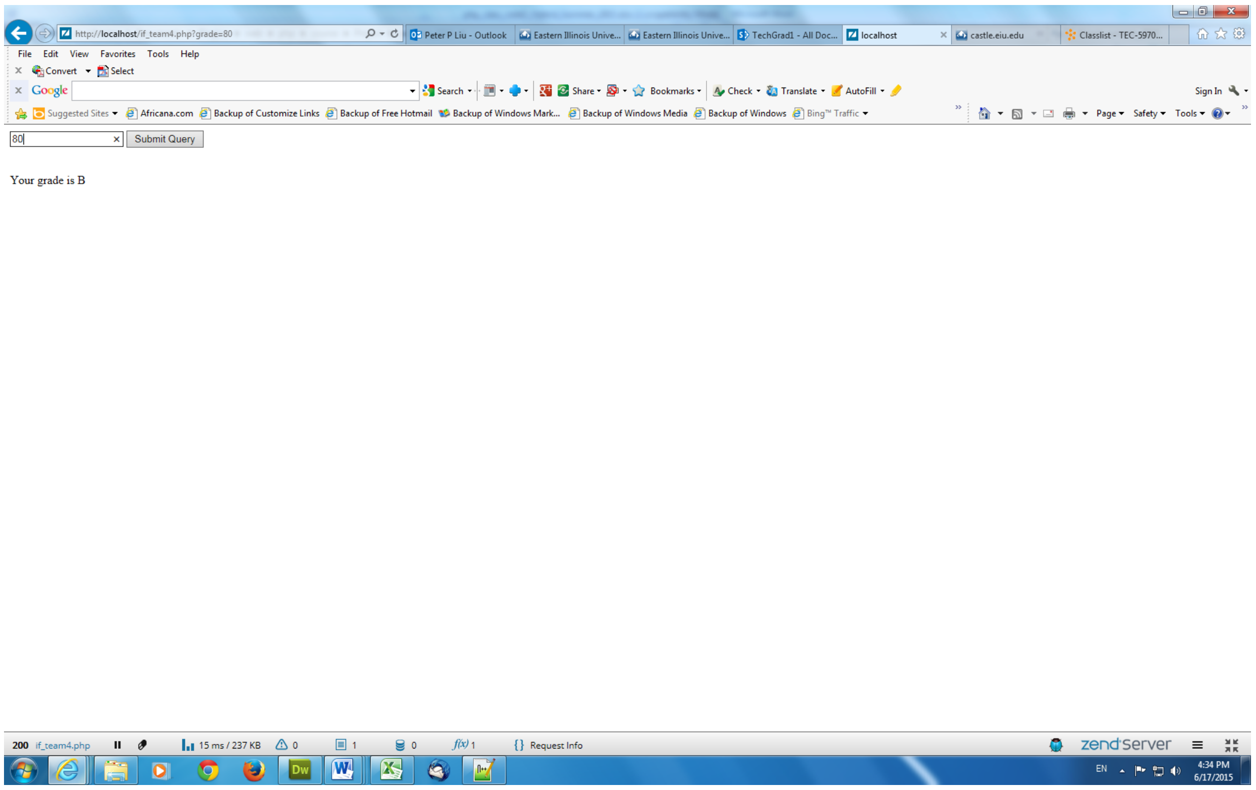
```
<html>
<form action="if_team4.php" method="get">
<input type="number" name="grade" id="grade" value="<?php echo
$grade; ?>"/>
 <input type="submit"/>
</form>

<?php
echo "</br>";

$t ="";
if(isset($_GET['grade']))
{
$t= $_GET['grade'];
if ($t < 60) {
echo "Your grade is F";
} elseif ($t<70) {
echo "Your grade is D";
}
elseif ($t<80) {
echo "Your grade is C";
}
elseif ($t<90) {
echo "Your grade is B";

}
elseif ($t<0 || $t>100) {
echo "Input should be between 0 - 100 only";
}
else {
echo "Your grade is A";
}
}
?>
```

I personally liked the way the team approached their work with an extra mile. You will see the screen similar to the following:



```
Your grade is B
```

Disclaimer: Since we are learning the basic concepts, we did not put a lot of details in the programs. Thus, those code are not professional grade yet. For example, the HTML form did not have any data validation check in it. You can input anything and the "php_action.php" will take them as they are, even with blank spaces.

For the time being, those are sufficient. By the way, the above HTML form will be used when you connect your PHP program with Oracle database.

PS: When I was revising the above file, I heard the good news from the radio. US Women's Soccer team toped Germany 2 to 0 to enter the final on June 30, 2015 in Women's World Cup. Good luck to US team.

14
14