PHP: Dynamic Web Gallery

Webworks – A Workshop Series in Web Design (Session #3)

Table of Contents:

1. Introduction

For those unfamiliar with PHP (Hypertext PreProcessor), it is important to emphasize that it is a web based language that is usually intermixed with HTML. Pages with PHP code are processed first by the server and then sent to the browser of a viewer. So called "dynamic" sites are created because the page is processed beforehand according to certain variables set or information present to dictate how the page or content is to be served.

When using PHP to design layouts such as web galleries, the basic principal is:

1. You set a variable
2. This variable is processed to relate how a page is displayed
3. The corresponding HTML code or any other code is then written into the page and sent for viewing.

Now it's just up to you how you achieve certain effects.

2. Advanced PHP Control Structures

During this workshop, we will be using several control structures that may differ from other languages.

Custom Functions

A custom function is written in the format:

```
function NameOfFunction(Argument){
  CodeForFunction;
}
```

Example:

```
function show_image_id($image){
   echo 'This is image'.$image;
}
```

As your pages become more complicated, it is good programming habit to keep any specialized feature that is repeated in a function.

Callback Functions
Callback functions are functions which evaluate a given variable and return a corresponding response. Typical uses deal with arrays containing data that needs filtering or altering. A callback function is used to do the filtering or altering. For today's workshop, we will be using callback functions with the built in `array_filter()` function.

Array_filter()
The `array_filter()` function basically filters a given array according to rules governed in a callback function. The format is:

NewArray = array_filter(ArrayToBeFiltered,"callbackfunction");

Example:

```
//Callback function
function positiveNumber($var){
return $var>0;
}

//your array of numbers
$array_to_be_filtered = array(1, -2, 4, 7, -10);

//using array_filter()
$newarray =
array_filter($array_to_be_filtered,"positiveNumber");

//your new array would look like this:
$newarray = (1, 4, 7);
```

Array_search()
The `array_search()` function accepts a string and searches an array for that element. When it finds it, it returns the corresponding key. If no element is found, FALSE is returned. The format is:

NewArray = array_search("string",ArrayOrString);

Example:

```
//Your Array
$target = array("one", "two", "three");

//Search
$search = array_search("two",$target);

$search would be number: 1 (Note: Remember array keys begin with
0)
```

To access the searched element, you would then just use:

```
$target[$search];
```

Opendir()
The `opendir()` function accepts a string that is a directory path and opens the directory into a handle to which you can then call other directory related functions, such as `readdir()` which we will get to. Format is:

Handle = opendir('directoryPath');

Example:

```
$dir = opendir('images/icons/');
```
(Note: make sure you include a '/' after your last folder)

Readdir()
The `readdir()` function accepts a string that is a handle for a directory and return the filename of the next file in the folder. If there is no next file, it will return a FALSE and stop the loop.
Format is:

$file = readdir(directoryHandle);

Example:
```
//Open a directory
$dir = opendir('images/icons/');

//Now read a file from it
$file = readdir($dir);
```

Note: The first two values of readdir() will return the values '.' , '..' which are directory navigation entities. For our project here and for usual projects, you will not need them, so you can use the unset() function to delete them.

Unset()
The `unset()` function accepts an element of an array to be deleted from the array. The format however must be in the array-key format:

Unset($array[key]);

Example:
```
//To strip out the '.' And '..' entrees mentioned above
unset($file[0]);
unset($file[1]);
```

Since '.' is the element in key 0, and '..' is the element in key 1, we delete them from the array using unset().

Sort()
The `sort()` function simply takes an array and sorts the elements within in chronological or alphabetical order depending on what type of data is in the array. You want to be careful

though because this function will reassign keys to the rearranged values, deleting any keys you may have given it.

Example:

```
$array = (1, 5, 13, 2, 3);
sort($array);

//Would look like this:
$array = array(1, 2, 3, 5, 13);

Unset($array[key]);
```

3. Variables

Variables in PHP are prefixed with the dollar sign '$'. They can be string, integer or Boolean. For this workshop however, we will be using variables that are embedded in the page dynamically. This is done through the $_GET variable.

$_GET

In PHP, variables can be appended to the end of URLs and links to make it easier to send variables to and from pages. These variables can then be taken out of the link and used in the page.

You append variables with the following format:
http://www.site.com/page.php?variable=value&variable2=value2

If you have more than 1 variable, separate them with a '&' sign. Make sure there are no spaces in your variable or value as that would break your link. If you do have spaces, make sure they are url encoded. (More info: http://www.albionresearch.com/misc/urlencode.php)

To get variables out of the link, you use `$_GET['variable']`. If my link looks like this:

http://www.simte.com/page.php?color=blue

Then, `$_GET['color']` = blue.

4. Looking at the Code

We begin by looking at the typical HTML of a picture gallery page on the net. It is very basic and is just a table with a bunch of random pictures on it. For sake of ease, thumbnails for images in this workshop are created by hand in an image editor and named the same name, except prefixed with 'th_'.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>My Dog Gallery</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
<style type="text/css">
```

```
<!–
//This CSS gives any image in the page a 5 pixel border of space
img {
   margin: 5px;
}
-->
</style>
</head>

<body>
<table width="100%"  border="0" cellspacing="0" cellpadding="0">
  <tr>
    <td> </td>
    <td align="center"><font size="5">My Dog Gallery</font></td>
    <td> </td>
  </tr>
  <tr>
    <td> </td>
    <td width="80%" align="center"><a href="#">Prev</a> | <a
href="#">Next</a></td>
    <td> </td>
  </tr>
  <tr>
    <td> </td>
    <td width="80%" align="center">
   <img src="imgs/th_edward2.jpg" width="200" height="175">
   <img src="imgs/th_lazydog.jpg" width="200" height="150">
   <img src="imgs/th_sammysmiles.jpg" width="200" height="133">
   <img src="imgs/th_stuffeddog.jpg" width="200" height="183">
   </td>
    <td> </td>
  </tr>
</table>

</body>
</html>
```

Code for PHP enriched HTML web gallery:

```
<?php
function scandir($dir){
   $dh  = opendir($dir);
      while (false !== ($filename = readdir($dh))) {
         $files[] = $filename;
      }
   sort($files);
   unset($files[0]);
   unset($files[1]);
   return $files;
}

function isthumb($var){
```

```
return (ereg('th_',$var));
}

function isntthumb($var){
return (!ereg('th_',$var));
}

function display($f){
if($f == 'gallery'){
  $array = scandir('imgs/');
  $array = array_filter($array,"isthumb");
      foreach ($array as $gh){
          echo '<a
href="?mode=image&id='.str_replace('th_',"",$gh).'"><img
src="imgs/'.$gh.'"></a>';
      }
}

if($f == 'image'){
   echo '<img src="imgs/'.$_GET['id'].'">';
}
}

function go($direction){
   $files = scandir('imgs/');
   $files = array_filter($files,"isntthumb");
   $current_loc = array_search($_GET['id'],$files);

   if($direction == 'next'){
      $go_key = $current_loc + 1;
   }

   if($direction == 'prev'){
      $go_key = $current_loc - 1;
   }

   if(empty($files[$go_key])){
   return '#';
   }
   else {
   return '?mode=image&id='.$files[$go_key];
   }
}

?>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>My Dog Gallery</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
```

```
<style type="text/css">
<!--
img {
   margin: 5px;
}
-->
</style>
</head>

<body>
<table width="100%"  border="0" cellspacing="0" cellpadding="0">
  <tr>
    <td> </td>
    <td align="center"><font size="5">My Dog Gallery</font></td>
    <td> </td>
  </tr>
  <tr>
    <td> </td>
    <td width="80%" align="center">

<?php
if(!empty($_GET['mode'])){
   echo '<a href="'.go(prev).'">Prev</a> | <a
href="'.go(next).'">Next</a>';
}
?>

</td>
    <td> </td>
  </tr>
  <tr>
    <td> </td>
    <td width="80%" align="center">
   <?php
        if(empty($_GET['mode'])){
           $type = 'gallery';
        }
        else {
           $type = 'image';
        }
        display($type);

        ?>
   </td>
    <td> </td>
  </tr>
</table>

</body>
</html>
```

Code Explanation: Functions

First off, this function here is a directory scan function that scans a directory's filenames into an array. The function is then sorted in alphabetical order and the '.' and '..' entries are deleted. The whole array containing all the files are then returned to the calling script.

```
function scandir($dir){
   $dh  = opendir($dir);
      while (false !== ($filename = readdir($dh))) {
         $files[] = $filename;
      }
   sort($files);
   unset($files[0]);
   unset($files[1]);
   return $files;
}
```

These two functions are simple callback functions that serve as filters to pick out files from arrays that are thumbnails and files that are not. The `ereg()` function here is a matching function that returns any value with the portion indicated. In this case, since our thumbnails are prefixed with 'th_', the function searches for that in the name and either returns a if it has that portion, or does not return a file if it has that portion.

```
function isthumb($var){
return (ereg('th_',$var));
}

function isntthumb($var){
return (!ereg('th_',$var));
}
```

This display function dictates what kind of format the page will be. Either thumbnails will be displayed if the page is in thumbnail mode, or large images will be displayed, as when a user clicks on a thumbnail. The argument `$f` can either be `'gallery'` or `'image'` and depending on which, a different process is evaluated. If it's a gallery, then you scan the image folder for all the image names and filter out all the large images using the callback function "isthumb". Then a foreach structure is used which goes through each value of the array and does an action. In this case, for every value of the `$array` , we are going to name it `$gh` and echo:

```
'<a href="?mode=image&id='.str_replace('th_',"",$gh).'"><img
src="imgs/'.$gh.'"></a>'
```

As you can see, the link is being appended with the variable 'mode' with value 'image' and another variable 'id' with value `$gh` which is the name of the image file currently in the array. The variable mode is actually used by this same script later on in the page code, you call the display function based on what the 'mode' variable in the link is. If the mode is 'image' then a simple <img> tag is echoed along with the filename which now was to be accessed via `$_GET` since the only time the page would be in 'image' mode would also mean that the 'id' variable is also there and it contains the filename of the image.

```
function display($f){
if($f == 'gallery'){
  $array = scandir('imgs/');
  $array = array_filter($array,"isthumb");
      foreach ($array as $gh){
          echo '<a
href="?mode=image&id='.str_replace('th_',"",$gh).'"><img
src="imgs/'.$gh.'"></a>';
      }
}

if($f == 'image'){
   echo '<img src="imgs/'.$_GET['id'].'">';
}
}
```

The last function we use in this web gallery is a navigation function that works with the “Prev” and “Next” links that popup when your in `‘image’` mode. First, the function scans the directory for all the files, but this time filters out all the thumbnails. This is because your viewing the large images, and the navigation should direct you to the next or previous large image. Now we want to find out where in this array of image names our current image is located. So we use the `array_search()` function to find the key of our current image in this array. We get the name of our image by using the $_GET method. Since our `go()` function is called with either the argument ‘prev’ or ‘next’, we use a simple `if` structure to decide how we wanna advance the array. If we want to go forward, then we want to add 1 to our current image key to get the key for the next image, and subtract 1 to get the key for the previous image. Then its just a matter of returing the string:

```
'?mode=image&id='.$files[$go_key]
```

We are again still in ‘image’ mode so there is the appended variable. This time the variable ‘id’ will be accessed by the array we scanned and the key we calculated. This will be echoed later on in the HTML where the normal HREF link goes. The empty condition checks to see if there is any image before or after. If there aren’t, then the calculation would be invalid and so you can just echo a ‘#’ which nullifies the link.

```
function go($direction){
   $files = scandir('imgs/');
   $files = array_filter($files,"isntthumb");
   $current_loc = array_search($_GET['id'],$files);

   if($direction == 'next'){
      $go_key = $current_loc + 1;
   }

   if($direction == 'prev'){
      $go_key = $current_loc - 1;
   }
```

```
   if(empty($files[$go_key])){
   return '#';
   }
   else {
   return '?mode=image&id='.$files[$go_key];
   }
}
```

Code Explanation: Body Structure

In the body of our HTML page, you will notice several areas that have been PHP enriched.

The first area deals with the navigation bar. In PHP, where you call a function is where the data will be sent. In this case, we want the nav bar to be in this section of the body in this TD. This snippet checks to see what mode the page is in according to the link. If the `'mode'` variable is not empty, then you are in image mode since that is the only mode we gave mode a value. Then you can echo the navigation bar. As you can see, the URLs for those links are generated dynamically using the `go()` function we wrote.

```
<?php
if(!empty($_GET['mode'])){
   echo '<a href="'.go(prev).'">Prev</a> | <a
href="'.go(next).'">Next</a>';
}
?>
```

The second area deals with the image area. This is where our images or thumbnails will be located. First we check what mode we are in according to the link. If the variable `'mode'` is empty, then we are in gallery mode and the `$type` will be gallery. If it is not, then we are in image mode and the $type will be image. Then we simply call our `display()` function with the corresponding argument.

```
<?php
if(empty($_GET['mode'])){
   $type = 'gallery';
}
else {
   $type = 'image';
}
display($type);

?>
```

5. Further Resources

PHP is an open source language which makes it easy for 3rd party developers to create their own functions and submit it to the source. As such, each new release of PHP includes new built in functions that make life easier for you. For example, the scandir() function we wrote by hand is now included in PHP versions 5.0 and up. Here are some links to help you with your PHP.

http://www.php.net
http://www.w3schools.com/php/default.asp
http://www.hotscripts.com/PHP/
http://www.onlyphp.com/