PHP: Functions, Patterns, Forms, Files

CS174

Chris Pollett

Sep 22, 2008.

Outline

- More on PHP Arrays
- Functions
- Variable Scope
- Pattern Matching
- Form Handling
- File Handling

More on PHP Arrays

- Last Wednesday, we saw the basics of creating/accessing an array in PHP: $arr=array(1,2,3); echo $arr[2]; /*would print 3 */
- Recall also $carr = array(); //create an empty array
- Arrays in PHP are similar to Perl hashes.
- The above way to create $arr can also be written in PHP as:
  $arr = array(0=> 1, 1=>2, 2=>3);
- Like hashes we can do things like $arr = array(“joe”=> 5, “mary” =>6);
- To get the keys and values we can use the functions: $keys = array_keys($arr) and $values = array_values($arr);
- Arrays can also be created by an assignment: $barr[1] = 5; // creates array $barr if doesn’t exist.
- If did the assignment $barr[] = 6; Then since the argument to [] wasn’t specified PHP will assign $barr[2] = 6;

Yet More on PHP Arrays

- You can call the unset function on the element in an array: $list= array(2,4,6,8); unset($list[2]);
- Some useful array functions: count -- returns the number of elements in an array, is_array, in_array, implode, explode, sort.
- To see how implode/explode work consider:

  $str=“this is a string”;

  $words = explode(“ ”, $str); /*acts like split except here the first argument is a string rather than a regular expression. So words is an array(“this”, “is”, “a”, “string”). PHP has a split function but not as fast, since arg might be a regular expression. */

  $str2 = implode(“ ”, $words); //undoes the explode.

Iterating Through Arrays

- The function current can be used to return a pointer to the current element in an array. The next function can be used to advance this pointer and get its value:

```
$cities = array("San Jose", "San Diego");
echo current($cities); // prints San Jose
$another = next($cities); // $another is now San Diego;
```

- There are also the functions each, prev, end, and reset to facilitate moving through array.
- The function each is similar to next except after advancing the current pointer, it returns the old pointer as a two element array consisting of a key/value pair.
- We saw last day that one can iterate through arrays using foreach($arr as $val){…}
- PHP also supports code like

```
$lows = array("Mon" => 23, "Tue" => 18);
foreach($lows as $day =>$temp )
{echo "$day lows were $temp";}
```

Functions

- The general format of a PHP functions is:

  function *name*([*parameter*]){…}

  For example,

  function inc($i){return ++$i;}
- A return value can be sent back using a return call as in many programming languages.
- You can modularize your code by putting several function definitions into a file and then use the include function to include them into any document that needs those functions.
- Parameters are passed by value. So the function call:

$b = inc($a); // leaves the value of $a unchanged

- You can call by reference by using an ampersand:

$b =inc(&$a); //here the value of $a is changed (one is added to it).

- You can also create functions with pass by reference parameters: function inc(&$i){…}

Variable Scope

- The default scope of a variable in PHP is only within the function that it is used. That is local scope:

```
$bob = 5;
function test()
{ $bob=6; echo $bob; //echo’s 6}
test();
echo $bob; //echo’s 5
```

- In order to access global variables within a local function one would need to use the global declaration:

```
$bob =5;
function test()
{
   global $bob; # if did not do bob would be NULL
   $bob-6;
   echo $bob;
}
test();
```

- PHP also supports static local variables. These preserve states between function calls: function addone () {static $count =0; echo $count++;}

Pattern Matching

- PHP supports Perl style regular expression and POSIX regular expressions.
- For example,

$fruits = preg_split(“/:/”, “apples:oranges”);

//would act like Perl’s split

- preg_match acts like acts like Javascript match.

Types of web forms

- Recall a basic web form looks like:

```
<form method="get" action="script.php">
   <input type="text" name="my_textfield"
       size="10" />
   <input type="hidden" name="secret_data"
      value="do not peak" />
   <input type="submit" name="sendform"
      value="Send this Form" />
</form>
```

- The method can be get or post.
- The get method sends the fields as urlencoded name=value pairs appended to the URL:

```
script.php?my_textfield=hello&secret_data=do%20not%20peak&sendform=
     Send%20this%20Form
```

- Post variables are sent as part of the content of an HTTP POST command (you won't see this in URL bar)
- File upload forms must use post and in addition must set and encoding type: <form method="post" action="script.php" enctype="multipart/form-data"><input type="file" name="my_file" />…</form>

Built-in Globals

- PHP makes available several important global variables which are useful for server side scripts.
- The phpinfo() function can be used to find out all of these globals.
- Here are some of the main ones:
  - $_SERVER -- an array of information about the server like $_SERVER[“SERVER_NAME”], $_SERVER [“DOCUMENT_ROOT”], $_SERVER [“QUERY_STRING”], etc
  - $_ENV -- an array of info about the runtime environment: $_ENV[“PATH”], $_ENV[“PWD”], etc.
  - $_REQUEST -- an array of the variables that have been get’d or post’s from forms. So if $_SERVER[“QUERY_STRING”] was hi=there&hi2=there2 would have $_REQUEST[“hi”] == “there” and $_REQUEST[“hi2”]=“there2”;
  - $_GET -- like $_REQUEST but only for get’d variables.
  - $_POST -- like $_REQUEST but for post’d variables.
- PHP can be configured with register_globals = On, in which case the variable $hi would be a global with value “there”. This is a bit risky security-wise.

File Reading

- Since PHP is a server side technology it is allowed to create, read, and write file on the server's filesystem.
- To open a file for reading one can do:

```
$fileHandle = fopen("my.dat", "r");
$file_string = fread($fileHandle, filesize("my.dat"));
fclose($fileHandle);
```

- Here fread reads in its second parameter many bytes.
- To read in a single line from a file one can use:

```
$line = fgets($fileHandle, $max_num_bytes_line);
```

- Alternatively, one can read the whole file in as a string using a single command like:

```
$string = file_gets_content("my.dat");
```

- Similarly, sometimes it is useful to read in the whole file as an array of lines:

```
$lines = file("my.dat");
```

File Writing

- File writing can be done in a similar fashion to file reading in PHP.

  $fileHandle = fopen(“my.dat”, “w”); // use “a” for append

  fwrite($fileHandle, $out_data);

  fclose($fileHandle);

- One can also write out a whole file based on a string using:

  file_put_contents(“out.dat”, $str);

File Locking

- Unless you as a coder do something, it is completely possible for two scripts to try to access the same file at the same time.
- To prevent this you should call the flock function to get a lock before you try to do something with a file:

```
$fp = fopen("/tmp/lock.txt", "w");
 if (flock($fp, LOCK_EX)) {
     // do an exclusive/write lock. use LOCK_SH (for  shared/read lock)
     fwrite($fp, "Write something here\n”);
     flock($fp, LOCK_UN); // release the lock
  }
 else {echo "Couldn't lock the file !";}   fclose($fp);
```

- Locks are released when fclose() is called.