COMP519 Web Programming

Lecture 23: PHP (Part 5)

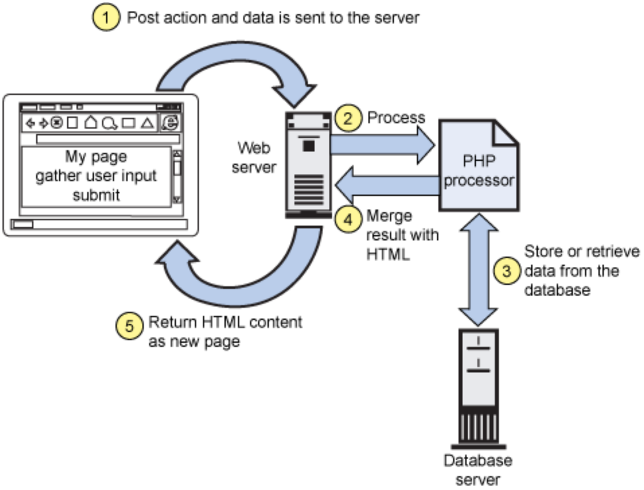
Handouts

Ullrich Hustadt

Department of Computer Science
School of Electrical Engineering, Electronics, and Computer Science
University of Liverpool

Contents

Web Applications using PHP


1 Post action and data is sent to the server
My page
gather user input
submit
Web server
2 Process
PHP processor
4 Merge result with HTML
3 Store or retrieve data from the database
Database server
5 Return HTML content as new page

IBM: Build Ajax-based Web sites with PHP, 2 Sep 2008.
`https://www.ibm.com/developerworks/library/wa-aj-php/` [accessed 6 Mar 2013]

HTML Forms

When considering Python CGI programming we have used HTML forms that generated a client request that was handled by a Python CGI program:

```
<form action=
 "http://student.csc.liv.ac.uk/cgi-bin/cgiwrap/uh/demo"
 method="post">
...
</form>
```

Now we will use a PHP script instead:

```
<form action="http://student.csc.liv.ac.uk/~uh/demo.php"
 method="post">
...
</form>
```

- The PHP script file must be stored in a directory accessible by the web server, for example `$HOME/public_html`, and be readable by the web server
- The PHP script file name must have the extension `.php`, e.g. `demo.php`

Information Available to PHP Scripts

- Information on the PHP environment
- Information on the web server and client request
- Form data
- Cookie/Session data
- Miscellaneous
  - `string date(`*format*`)`
    returns the current date/time presented according to *format*
    for example, `date('H:i␣l,␣j␣F␣Y')`
    results in `12:20 Thursday, 8 March 2012`
    (See `http://www.php.net/manual/en/function.date.php`)
  - `int time()`
    returns the current time measured in the number of seconds
    since January 1 1970 00:00:00 GMT

PHP Environment

- `phpinfo()` displays information about the PHP installation and EGPCS data (Environment, GET, POST, Cookie, and Server data) for the current client request
- `phpinfo(`*part*`)` displays selected information

```
<html lang="en-GB"><head></head><body>
<?php
  phpinfo();                    // Show all information
  phpinfo(INFO_VARIABLES);      // Show only info on EGPCS data
?>
</body></html>
```

`http://cgi.csc.liv.ac.uk/~ullrich/COMP284/examples/phpinfo.php`

`INFO_GENERAL`	The configuration, php.ini location, build date, web server
`INFO_CONFIGURATION`	Local and master values for PHP directives
`INFO_MODULES`	Loaded modules
`INFO_VARIABLES`	All EGPCS data

Manipulating the PHP Configuration

The following functions can be used to access and change the configuation of PHP from within a PHP script:

- `array ini_get_all()`
  - returns all the registered configuration options
- `string ini_get(option)`
  - returns the value of the configuration option on success
- `string ini_set(option, value)`
  - sets the value of the given configuration option to a new value
  - the configuration option will keep this new value during the script's execution and will be restored afterwards
- `void ini_restore(option)`
  - restores a given configuration option to its original value

Server Variables

The superglobal $_SERVER array stores information about the web server and the client request

⤳ Similar to os.environ for Python CGI programs

```
<html lang="en-GB"><head></head><body>
<?php
echo 'Server software: ',$_SERVER['SERVER_SOFTWARE'],'<br>';
echo 'Remote address:  ',$_SERVER['REMOTE_ADDR'],     '<br>';
echo 'Client browser:  ',$_SERVER['HTTP_USER_AGENT'],'<br>';
echo 'Request method:  ',$_SERVER['REQUEST_METHOD'];
?></body></html>
```

http://cgi.csc.liv.ac.uk/~ullrich/COMP284/examples/server.php

```
Server software: Apache/2.2.22 (Fedora)
Remote address: 10.128.0.215
Client browser: Mozilla/5.0 ... Chrome/41.0.2272.53 ...
Request method:
```

See http://php.net/manual/en/reserved.variables.server.php for a list of keys

Form Data

- Form data is passed to a PHP script via the three superglobal arrays:

`$_POST`	Data from POST client requests
`$_GET`	Data from GET client requests
`$_REQUEST`	Combined data from POST and GET client requests (derived from `$_POST` and `$_GET`)

⇝ Accessing `$_REQUEST` is the equivalent in PHP to accessing the 'dictionary' of a `cgi.FieldStorage` instance in Python

```
<form action="process.php" method="post">
<label>Enter your user name:
      <input type="text" name="username"></label><br>
<label>Enter your full name:
      <input type="text" name="fullname"></label><br>
<input type="submit" value="Click␣for␣response"></form>
```

`$_REQUEST['username']`	Value entered into field with name 'username'
`$_REQUEST['fullname']`	Value entered into field with name 'fullname'

Forms in PHP: Example (1)

- Create a web-based system that asks the user to enter the URL of a file containing bibliographic information
- Bibliographic informatiom will have the following form:

```
@entry{
 name={Jonas Lehner},
 name={Andreas Schoknecht},
 title={<strong>You only live twice</strong>},
}
@entry{
 name={Andreas Schoknecht},
 name={Eva Eggeling},
 title={No End in Sight?},
}
```

- The system should extract the names, count them, and create a table of names and their frequency, ordered from most frequent to least frequent

Forms in PHP: Example (1)

Useful PHP functions:

- `string file_get_contents(filename)`
  returns the contents of the file/URL *filename*, or `FALSE` on failure
- `array array_count_values(arr)`
  returns an array using the values of the array *arr* as keys and their frequency in *arr* as values

```
$array = array('a','c','b','b','c','c');
$count = array_count_values($array);
# $count = ['a' => 1,  'c' => 3,  'b' => 2]
```

- `bool arsort(arr)`
  sorts *arr* according to associated values maintaining their correlation with keys, returns `TRUE` on success and `FALSE` on failure

```
arsort($count)
# $count = ['c' => 3, 'b' => 2, 'a' => 1]
```

Forms in PHP: Example (1)

extract_names.php

```
<!DOCTYPE html>
<html><head><title>Name Extraction</title></head><body>
<?php
 require_once 'extraction.php';
 if (isset($_SERVER['REQUEST_METHOD']) &&
     $_SERVER['REQUEST_METHOD'] == 'POST' &&
     isset($_REQUEST['url'])) {
   $extracted_names = extract_names($_REQUEST['url']);
   echo "<div>The names occurring in <br>",htmlspecialchars($_REQUEST['url']),
        "<br>are</div>$extracted_names\n";
 } else {
   echo <<<FORM
   <form method="post">
     <label>Enter a URL:
       <input type="text" name="url" size="100"
        value="http://cgi.csc.liv.ac.uk/~ullrich/COMP284/tests/a1test1.txt">
     </label><br><br>
     <input type="submit" value="Extract Names">
   </form>
FORM;
 }
?>
</body></html>
```

http://cgi.csc.liv.ac.uk/~ullrich/COMP519/examples/extract_names.php

Forms in PHP: Example (1)

extraction.php

```
<?php
function extract_names($url) {
 $text = file_get_contents($url);
 if ($text === false)
   return "ERROR: INVALID URL!";
 else {
   $correct = preg_match_all("/name={([^\}]+)}/",
              $text,$matches,PREG_PATTERN_ORDER);
   if ($correct == 0) return "ERROR: NO NAMES FOUND";
   $count = array_count_values($matches[1]);
   arsort($count);
   foreach ($count as $name => $number) {
     $table .= "<tr><td>$name</td><td>$number</td></tr>";
   }
   $table = "<table><thead><tr><th>Name</th><th>No of occur".
   "rences</th></tr></thead><tbody>".$table."</tbody></table>";
   return $table;
} }
?>
```

http://cgi.csc.liv.ac.uk/~ullrich/COMP519/examples/extraction.php

Revision and Further Reading

Read

- Chapter 11: Form Handling

of R. Nixon: Learning PHP, MySQL & JavaScript: with jQuery, CSS & HTML5. O'Reilly, 2018.