Server Side Languages

Web Design & Development

Day 2

Lecture Overview

- 2.2 Sessions
- 2.3 File uploads
- 2.4 Redirects
- 2.5 Password hashing, salting, security
- 2.6 Manipulating Graphics
- 2.7 Filesystem Functions
- 2.8 User input validation, Regex

State Persistence

Traditional HTML requests by nature result in short-term, stateless interactions.

1. Browser open connection to server (IP address, port 80) requests page from web server (HTTP GET Request)
2. Server retrieves or generates HTML response and sends back to browser through IP connection made by initial request (HTTP Response)
3. Server and Browser close IP connection when data transfer is complete.

For EACH HTTP request sent to/received by server these steps take place. This includes EACH .js .css .jpeg .png etc. file that is included on a single web page.

Because each transaction is a standalone request, there is no way for the server to know which client made each request.

- Could we track by IP address and client port?
- Could we store data on the browser client that could be read back later?

Sessions

Remember that there is no persistence between HTML request/ response pairs from the client/web server. Sessions allow some persistence to emulate a connection-oriented environment.
PHP sets default session length in php.ini file (usually 30 minutes, but can vary).

```
session_start();
if (!isset($_SESSION['count'])) {
$_SESSION['count'] = 0;
} else {
$_SESSION['count']++;
}

session_start();
unset($_SESSION['count']); //remove the session variable count
session_destroy(); // Finally, destroy the session.
```

Exercise 2.2: PHP Sessions

1. Create two PHP scripts:
   setsession.php - Start a session and set session variables based on GET parameters you pass through the querystring. Set at least 3 variables.
   getsession.php - Retrieve an existing session and echo out the session variables you set in setsession.
   Wait 60 seconds and try to retrieve getsession again - does it still retrieve variables?
2. Add the following line to both scripts.
   ini_set('session.gc_maxlifetime', 60); //set session timeout to 60sec

Stop and restart Apache, then run setsession/getsession again.
Wait for over 60 seconds and run getsession again.

File Uploads

The $_FILES[] is a superglobal that contains an array of files and their attributes as uploaded by the browser.

When files are uploaded through a POST method, PHP will store the uploaded files in a temporary location and with a temporary filename.

```
$_FILES['userfile']['name']
The original name of the file on the client machine.
$_FILES['userfile']['type']
The mime type of the file, if the browser provided this information. This
mime type is however not checked on the PHP side.
$_FILES['userfile']['size']
The size, in bytes, of the uploaded file.
$_FILES['userfile']['tmp_name']
The temporary filename of the file in which the uploaded file was stored on
the server.
$_FILES['userfile']['error']
The error code associated with this file upload.
```

File Uploads

Form type must include multipart/form-data as shown below.

```
<form enctype="multipart/form-data" action="upload.php" method="POST">
<!-- Name of input element determines name in $_FILES array -->
Send this file: <input name="userfile" type="file" />
<input type="submit" value="Send File" />
</form>
```

```
$uploaddir = './uploads/'; //physical path on Apache
$uploadfile = $uploaddir . basename($_FILES['userfile']['name']);
if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
echo "File is valid, and was successfully uploaded.\n";
} else {
echo "Possible file upload attack!\n";
}
print_r($_FILES);
```

Exercise 2.3: File Uploads

Create a test script to upload a JPEG file to your web server. When upload is completed, echo back out the file attributes and display the JPEG image.

```
<form enctype="multipart/form-data" action="upload.php" method="POST">
<!-- Name of input element determines name in $_FILES array -->
Send this file: <input name="userfile" type="file" />
<input type="submit" value="Send File" />
</form>

$uploaddir = './uploads/'; //physical path on Apache
$uploadfile = $uploaddir . basename($_FILES['userfile']['name']);
if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
echo "File is valid, and was successfully uploaded.\n";
}
```

Redirects

Sends a response to browser to redirect it to another page. Also sends HTTP Status Code about whether it is a permanent or temporary redirection.

http://www.w3.org/QA/Tips/reback

```
<?php
http_redirect("relpath", array("name" => "value"), true, HTTP_REDIRECT_PERM);
?>
```

```
The above example will output:
HTTP/1.1 301 Moved Permanently
X-Powered-By: PHP/5.2.2
Content-Type: text/html
Location: http://www.example.com/curdir/relpath?name=value&PHPSESSID=abc

Redirecting to <a href="http://www.example.com/curdir/relpath?
name=value&PHPSESSID=abc">http://www.example.com/curdir/relpath?
name=value&PHPSESSID=abc</a>.
```

Hashing

A hash is a one-way cryptographic algorithm to mathematically compute an input to a fixed-length result. Hashes are quickly computed with minimal processing resources (making them fast), and are often used as a checksum to compare two larger values.

MD5 - 32 character hexadecimal number
http://us1.php.net/manual/en/function.md5.php
http://www.faqs.org/rfcs/rfc1321.html

SHA1 - 40 character hexadecimal number
http://us1.php.net/manual/en/function.sha1.php
http://www.faqs.org/rfcs/rfc3174.html

Hashing

Therefore, a string of “apple” could be hashed using MD5 to generate a checksum of

“1f3870be274f6c49b3e31a0c6728957f”

Notice the result is a value with hexadecimal numbers (0-9, A-F)

```
$str = 'apple';
if (md5($str) === '1f3870be274f6c49b3e31a0c6728957f') {
  echo "Would you like a green or red apple?";
}
if (sha1($str) === 'd0be2dc421be4fcd0172e5afceea3970e2f3d940') {
  echo "Would you like a green or red apple?";
}
```

How Secure is Hashing?

While hashing is a one-way encryption (meaning it cannot be reversed or “unencrypted” by just reversing the algorithm), it is possible to guess combinations until you find a match.

The easiest way is to use a pre-built dictionary, hash the words and store in a database, then search by hashed value for a match.
(These are often called Rainbow Tables.)

How long does it take to find a hashed value?
https://crackstation.net/

```
md5 – '1f3870be274f6c49b3e31a0c6728957f'
sha1 – 'd0be2dc421be4fcd0172e5afceea3970e2f3d940'
```

Salting

Because hash values can be guessed (imagine looping through all possible combinations until you find one that works), they are generally considered insecure along for use as storing passwords.

Salting adds an additional value that is unknown to an attempted hacker to make hashes more secure. Concatenate the salt to the string you need to hash.

```
$str = 'apple';
$salt = 'mysectretpassword';
//without salting
if (md5($str) === '1f3870be274f6c49b3e31a0c6728957f') { echo "valid user";}
//with salting
if (md5($salt.$str)==='asda3870be2345kf431a0c6728957f') {echo "valid user";}
```

Password Hash: PHP5.5.x

password_hash() creates a new password hash using a strong one-way hashing algorithm.

```
/**
 * We just want to hash our password using the current DEFAULT algorithm.
 * This is presently BCRYPT, and will produce a 60 character result.
 *
 * Beware that DEFAULT may change over time, so you would want to prepare
 * By allowing your storage to expand past 60 characters (255 would be good)
 */
echo password_hash("rasmuslerdorf", PASSWORD_DEFAULT)."\n";
```

The above example will output something similar to:

```
$2y$10$.vGA1O9wmRjrwAVXD98HNOgsNpDczlqm3Jq7KnEd1rVAGv3Fykk1a
```

Images

GD is a graphics and image manipulation library that is accessible through PHP.

Image Resources (references to objects) are passed to/from functions.

```
header("Content-type: image/png"); //must set header type to PNG
$string = $_GET['text']; //querystring ?text=somevalue
$im = imagecreatefrompng("images/button1.png"); //background image
$orange = imagecolorallocate($im, 220, 210, 60); //RGB Values
$px = (imagesx($im) - 7.5 * strlen($string)) / 2; //find center
imagestring($im, 3, $px, 9, $string, $orange); //merge text onto image
imagepng($im); //output image to browser as PNG
imagedestroy($im); //release resources
```

Image Functions

Some commonly-used image functions. See PHP reference.
http://us3.php.net/manual/en/book.image.php

getimagesize($image_path); //retrieve image dimensions
imagecreatefromjpeg($filename); //read jpg input file
imagejpeg(); imagepng(); imagegif(); //output/save as jpg, png, or gif
imagedestroy(); //release resources
imagecreatetruecolor(); //create blank canvas
imagecopyresampled(); //copy image resource to another resource
imagerectangle(); //create rectangle inside resource

Image Copy

Sample function to copy image.

```
function imagecopy($inputfile, $newfile) {
  $im = imagecreatefromjpeg($inputfile); //read original image into resource
  imagejpeg($im, $newfile); //save resource to jpg
  imagedestroy($im); //release resource
}
```

$n is a resource (or pointer) that contains the binary bitmap contents of the imported image.

CAPTCHA

http://www.captcha.net/

CAPTCHA is a method to verify that a human is interacting with a site, to avoid automated “bot” submissions from scripts or attempts to SPAM.

The term CAPTCHA (for Completely Automated Public Turing Test To Tell Computers and Humans Apart) was coined in 2000 by Luis von Ahn, Manuel Blum, Nicholas Hopper and John Langford of Carnegie Mellon University.

Exercise 2.4: Image Creation

1. Create a CAPTCHA-type image based on a word using the code from the previous page.

2.Once you have successfully created the image, embed it using an <img> tag on a sample webpage and have your script create the image based on a GET variable.

Ex:

```php
public function msg($msg){

    $container = imagecreate(300,200);
    $black = imagecolorallocate($container,0,0,0);
    $white = imagecolorallocate($container,255,255,255);
    $font = 'fonts/texb.ttf';
    imagefilledrectangle($container, 0, 0, 250,150,$black);
    imagerectangle($container,0,0,50,60,$white);
    imagefttext($container,12,0,0,12,$white,$font,$msg);
    header('Content-Type: image/png');
    imagepng($container,null);
    imagedestroy($container);

}
```

Filesystem Functions

Filesystem components allow interaction with local files on the webserver, or retrieving remote files through URLs.

```
$file = file_get_contents('./people.txt', true);
$homepage = file_get_contents('http://www.example.com/');
echo $homepage;
```

Writing Files

```
$file = 'people.txt';
// The new person to add to the file
$person = "John Smith\n";
// Write the contents to the file,
// using the FILE_APPEND flag to append the content to the end of the file
// and the LOCK_EX flag to prevent anyone else writing to the file at the sa
me time
file_put_contents($file, $person, FILE_APPEND | LOCK_EX);
```

Validation

filter_var — Filters a variable with a specified filter

http://www.php.net/manual/en/function.filter-var.php

```
$email_a = 'joe@example.com';
$email_b = 'bogus';
if (filter_var($email_a, FILTER_VALIDATE_EMAIL)) {
echo "This ($email_a) email address is considered valid.";
}
if (filter_var($email_b, FILTER_VALIDATE_EMAIL)) {
echo "This ($email_b) email address is considered valid.";
}
```

preg_match - Searches subject for a match to the regular expression given in pattern.

http://us1.php.net/manual/en/function.preg-match.php

```
$subject = "abcdef";
$pattern = '/^def/';
preg_match($pattern, $subject, $matches, PREG_OFFSET_CAPTURE, 3);
```

Exercise 2.5: Server-Side Validation

Create an HTML form with email input.

(Do not use JavaScript or HTML5 client-side validation.)

Create a PHP script to capture the form POST results, and test whether the input is a valid email address. Echo out “Valid” or “Invalid”

filter_var($email_a, FILTER_VALIDATE_EMAIL)

Questions?

Lab 2: Sessions, Images, Filesystem

Refer to FSO Lab 2 and Screencast 2 for this assignment.