Managing the Information

Data, files, sessions, & forms

What we will cover

- Form Handling
  - Security Functions (Filter Input / Escape Output)
- Sessions
- File Handling
  - Parsing Specific Formats
    - CSV
    - JSON
    - XML

Form Handling

The meat of any traditional web app

Form-related Superglobals

Data from forms comes in three superglobals:

$_GET: Contains everything submitted in a URL string after ? or in a form submitted with a get type. In practice, these are the same.

$_POST: Contains everything submitted in a form with a post type (or a properly crafted POST request).

$_REQUEST: Contains data submitted from both GET and POST sources. Avoid.

Mapping Form Elements to Superglobal Arrays

PHP treats each POST or GET element as a key-value pair. The name attribute of the form element is the key, and the value provided is the array.

So this element:

```
<input type="hidden" name="user_id" id="userId" value="123">
```

becomes this:

```
echo $_POST['user_id']; // 123
```

Multiple-Value Elements

As you might guess, putting multiple values from something like checkboxes or multiple-value select lists (multi-combobox in MS-speak) requires some special handling. Fortunately PHP's square bracket syntax for arrays comes in handy.

Multiple-Value Examples

```
<input type="checkbox" name="device[]">iOS</input><br>
<input type="checkbox" name="device[]">Android</input>
```

If both are checked, becomes

```
var_dump($_POST);
```

```
array(1) {
 ["device"]=>  array(2) {
   [0]=>   string(3) "iOS"
   [1]=>   string(7) "Android"
 }
}
```

Multiple-Value Examples

```
<select type="multiple" name="device[]">
   <option>iOS</option>
   <option>Android</option>
</select>
```

If both are selected, becomes

```
var_dump($_POST);
```

```
array(1) {
 ["device"]=> array(2) {
  [0]=>  string(3) "iOS"
  [1]=>  string(7) "Android"
 }
}
```

Form Array Tricks

```
<input type="text" name="name[first]">
<input type="text" name="name[family]">
```

…can become…

```
var_dump($_POST);
array(1) {
  ["name"]=> array(2) {
     ["first"]=>  string(4) "John"
     ["family"]=>  string(3) "Doe"
  }
}
```

PHP is a Template Language

Remember, you can do all the templating you need with just PHP, e.g.:

```
<input type="checkbox" name="model" value="A4"
<?php if($_POST['model']['A4']): ?>
   checked
<?php endif; ?>>
```

…but it can become a pain, so lots of form libraries exist, like Zend_Form and Symfony’s Form component, as well as template languages like Twig.

Security functions

Filter Input / Escape Output

Build-in Filtering Functions

- Allows you to specify data and a predefined filter.
- Provides both SANITIZE & VALIDATE style filters.
- These functions operate on any variable or array of yours:
    - **filter_var**
    - **filter_var_array**
- These operate on the superglobal arrays, such as $_GET:
    - **filter_input**
    - **filter_input_array**

Example: filter a single input

Form Data:

```
$_GET = ['page' => 1];
```

Filter:

```
$page = filter_input(
  INPUT_GET, // request type
  'page', // name of input
  FILTER_SANITIZE_NUMBER_INT, // filter
  ['options' => //options
     ['min_range' => 1, 'max_range' => 6]
  ]
);
```

Example: POST data

```
$_POST = [
    'email'     => 'adent@example.com',
    'name'      => 'Arthur P. Dent',
    'meaning'   => '42',
    'favorites' => ['towel', 'lager', 'peanuts'],
    'website'   => 'http://hoopyfrood.com/',
    'username'  => 'arthurdent',
    'password'  => '******',
];
```

Example: filter an array of input

```
$args = [
    'email'     => FILTER_VALIDATE_EMAIL,
    'name'      => FILTER_SANITIZE_STRING,
    'meaning'   => ['filter' => FILTER_VALIDATE_INT,
         'options' = ['min_range'  => 40, 'max_range' => 54] ],
    'favorites'=> ['filter' => FILTER_SANITIZE_STRING,
         'flags'   => FILTER_REQUIRE_ARRAY ],
    'website'   => FILTER_VALIDATE_URL,
    'username' => ['filter' => FILTER_VALIDATE_REGEXP,
         'options' = ['regexp' => '/^[A-Za-z0-9_]+$/'] ],
    'password'  => FILTER_UNSAFE_RAW,
];
$myinputs = filter_input_array(INPUT_POST, $args, TRUE);
```

XSS Defense

Any time you display content to HTML that isn’t meant to display as HTML (i.e., text), you can defend against XSS attacks by using htmlentities() on any output.

```
$name = htmlentities($_POST['first_name'], ENT_QUOTES, 'UTF-8',
FALSE);
```

Much More to Learn

PHP security is a big topic, and beyond the scope of this course. For an in-depth examination to improve your security skills and learn more ways to secure PHP as well as your code, see our Web Security class.

Challenge Form

```
<form action="" method="get">
  <div>
    <label>First number
      <input type="text" name="number1">
    </label>
    +
    <label>Second number
      <input type="text" name="number2">
    </label>
    <input type="submit" value="Add these">
  </div>
</form>
```

Challenge

- Using the form from the last slide: write a script to add two numbers together.
- Validate that both are present and valid numbers.
- If they are not present, display an error informing the user.
- If they are not numbers, display a different error.
- Display a success message with the sum if there are no errors.
- Bonus: Display the error next to the form element that is wrong.

Sessions

Starting a session

On each request, applications must start a session. This is done with session_start(). Doing this will generate a session ID or resume a previous session if the client passes a session ID along in a cookie.

For this reason, be sure you call session_start() before creating any output. Most frameworks handle this for you.

Storing Session Data

Like form data, session data is stored in a superglobal, $_SESSION.

Since session data is saved once an ID has been sent to the browser, set anything before sending real output to the client.

You can store and retrieve data in $_SESSION like you would in any other array. PHP handles the serialization/unserialization at the beginning of the request for you.

Example: Session

```
<?php
session_start();
$_SESSION['word'] = filter_input(
        INPUT_POST, 'word', FILTER_SANITIZE_STRING
);
echo "The word is: " . htmlspecialchars($_SESSION['word']);
```

Challenge

Fill out a multi-step form and display the results at the end

Basic File Handling

Read Directory Files into Array

```
$dir = 'example';
$files = scandir($dir, 'w') or die('Cannot read directory: '.$dir);
//implicitly creates file
foreach ($files as $file) {
//do something with $file
}
```

Open and Close Directory

```
$dir = 'example';
$dh = opendir($dir) or die('Cannot open file: '.$file);
//access directory
closedir($dh);
```

Read Directory

```
$dir = 'example';
if ($dh = opendir($dir)) {
 while (($file = readdir($dh)) !== false) {
   if (substr($file, 0, 1) !== '.') {
     //do something with $file
   }
 }
 closedir($dh);
}
```

Create a File

```
$file = 'file.txt';
$fh = fopen($file, 'w') or die('Cannot open file: '.$file);
//implicitly creates file
fclose($fh);
```

Open and Close a File

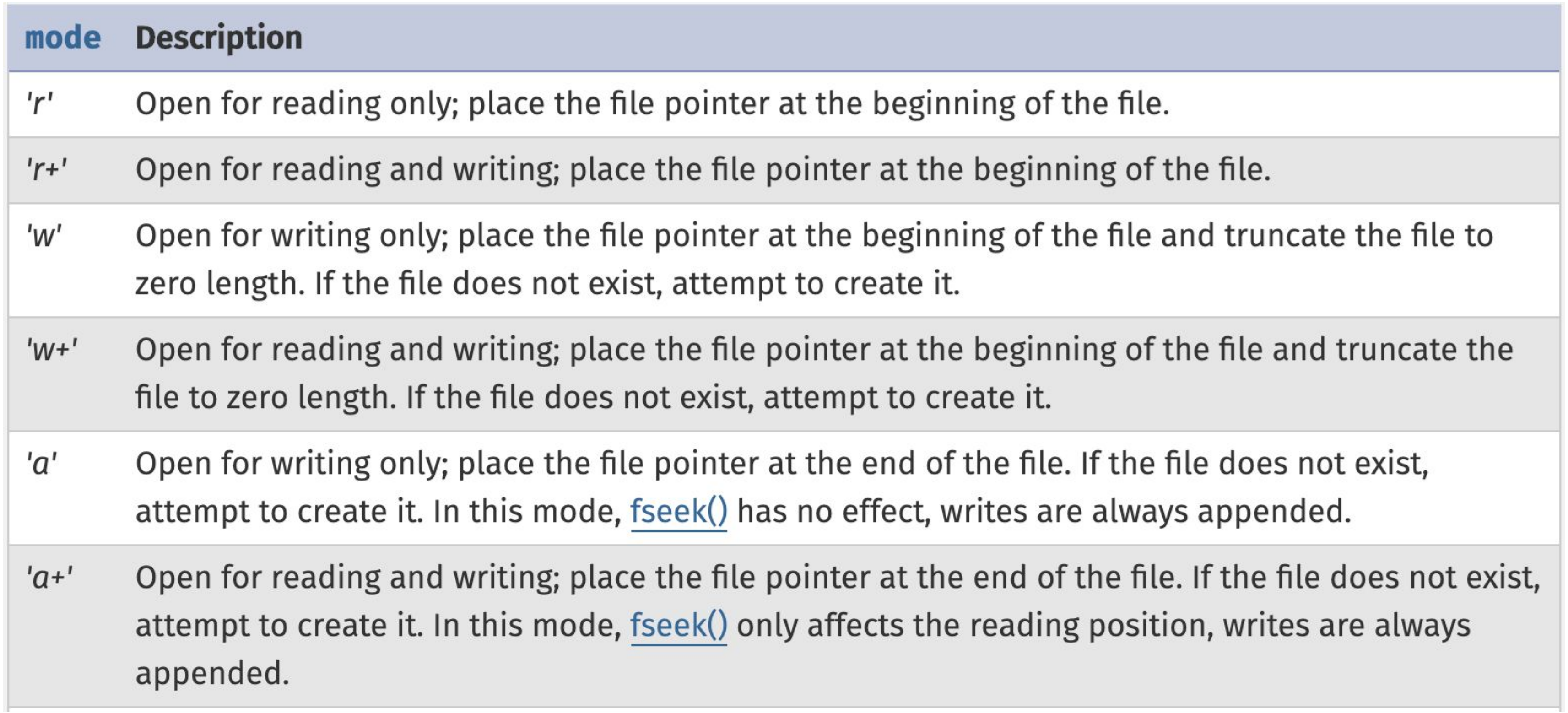
```
$file = 'file.txt';

$fh = fopen($file, 'w') or die('Cannot open file: '.$file);

//open file ('w','r','a')... see mode below

fclose($fh);
```

File Modes: most used



mode	Description
'r'	Open for reading only; place the file pointer at the beginning of the file.
'r+'	Open for reading and writing; place the file pointer at the beginning of the file.
'w'	Open for writing only; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
'w+'	Open for reading and writing; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
'a'	Open for writing only; place the file pointer at the end of the file. If the file does not exist, attempt to create it. In this mode, fseek() has no effect, writes are always appended.
'a+'	Open for reading and writing; place the file pointer at the end of the file. If the file does not exist, attempt to create it. In this mode, fseek() only affects the reading position, writes are always appended.

Read a File

```
$file = 'file.txt';
$fh = fopen($file, 'r');
$data = fread($fh,filesize($file));
fclose($fh);
```

Write to a File

```
$file = 'file.txt';
$fh = fopen($file, 'w') or die('Cannot open file: '.$file);
$data = 'This is the data';
fwrite($fh, $data);
fclose($fh);
```

Append to a File

```
$file = 'file.txt';
$fh = fopen($file, 'a') or die('Cannot open file: '.$file);
$data = 'New data line 1';
fwrite($fh, $data);
$new_data = "\n".'New data line 2';
fwrite($fh, $new_data);
fclose($fh);
```

Delete a File

```
$file = 'file.txt';
unlink($file);
```

More File Related Functions

PHP has MANY file related function. Checkout the Filesystem Functions page of the manual.

Parsing Specific Formats

CSV, JSON, XML

PHP CSV Functions

fgetcsv Similar to fgets() except that fgetcsv() parses the line it reads for fields in CSV format and returns an array containing the fields read.

fputcsv formats a line (passed as a fields array) as CSV and writes it (terminated by a newline) to the specified file handle.

Example: Reading CSV

```php
if (($fh = fopen('data/csv/people.csv', 'r')) !== false) {
  $header = fgetcsv($fh);
  extract(array_flip($header));
  while (($contact = fgetcsv($fh)) !== false) {
     echo $contact[$first];
  }
}
```

Example: Writing CSV

```
$new_person = [
  filter_input(INPUT_POST, 'first', FILTER_SANITIZE_STRING),
  filter_input(INPUT_POST, 'img', FILTER_SANITIZE_URL)
];
if (($fh = fopen('../data/csv/people.csv', 'a+' )) !== false) {
  fseek($fh, -1, SEEK_END);
  if (fgets($fh) != PHP_EOL) {
     fputs($fh, PHP_EOL);
  }
  fputcsv($fh,$new_person);
  fclose($fh);
}
```

PHP JSON Functions

json_decode Takes a JSON encoded string and converts it into a PHP variable. By default it will return an object, but the second parameter will allow you to convert the object into an associative array.

json_encode Returns a string containing the JSON representation of the supplied value. The encoding is affected by the supplied options and additionally the encoding of float values depends on the value of serialize_precision.

Example: Reading JSON as an Object

```
$books = json_decode(
      file_get_contents('data/json/top_programming_books.json')
);
if (is_object($books->collection->books[0])) {
  foreach ($books->collection->books as $book) {
     echo $book->title . '<br />';
     echo $book->author_name . '<br />';
  }
}
```

Example: Reading JSON as an Array

```
$books = json_decode(
        file_get_contents('data/json/top_programming_books.json'),
        true
);
if (is_array($books['collection']['books']) {
   foreach (is_array($books['collection']['books']as $book) {
      echo $book['title'] . '<br />';
      echo $book['author_name'] . '<br />';
   }
}
```

Example: Writing JSON

```php
$new_book = [
  'title' => filter_input(INPUT_POST, 'title', FILTER_SANITIZE_STRING),
  'author_name' => filter_input(INPUT_POST, 'author_name', FILTER_SANITIZE_STRING),
];
$file = '../data/json/top_programming_books.json';
$books = json_decode(file_get_contents($file)); // read in the json from a file
if (is_object($books->collection->books[0])) {
  $books->collection->books[] = $new_book; // add the book array to json object
}
// encode the object back to json
$json = json_encode($books, JSON_PRETTY_PRINT | JSON_UNESCAPED_SLASHES);
// write the file
file_put_contents($file, $json);
```

PHP XML Functions

simplexml_load_file Convert the well-formed XML document in the given file to an object of type SimpleXMLElement.

When you need more control of your xml, you could try a package like sabre/xml.

***TIP**: Scalable Vector Graphics (SVG) is an XML markup language for creating two-dimensional images using vectors. If you wish to work with SVGs, there are also packages specifically designed to work with SVG, such as php-svg by JangoBrick to read, edit, write, and render SVG files with PHP*

Example: Reading XML

```
$xml = simplexml_load_file($file);
echo '<h3>'$xml->channel->title .'</h3>';
echo 'Description' . $xml->channel->description .'</p>'
```

Example: Writing XML

```
if ($xml = simplexml_load_file($file)) {
  $item->addChild('title',
      filter_input(INPUT_POST, 'title', FILTER_SANITIZE_STRING));
  $item->addChild('description',
      filter_input(INPUT_POST, 'description', FILTER_SANITIZE_STRING));
  $xml->asXML($file);
}
```

Challenge

Read and Write a CSV File

Read and Write a JSON File

Read and Write a XML File

Questions and Answers

Thank you for your participation