# SOEN 287
# WEB PROGRAMMING

PHP – 3

Form handling

File I/O

# Form Handling

http://www.w3schools.com/tags/ref_httpmethods.asp

- The Hypertext Transfer Protocol (HTTP) is designed to enable communications between clients and servers.

- One way is through forms.

# Form Handling - Steps

http://www.tutorialspoint.com/php/php_get_post.htm

- User enters information in a form and presses the `Submit` button
- Before the browser sends the information the information is encoded
- Encoded information transmitted to server
- Server uses a program to decode contents
- Performs what ever computation/checks required
- Produces output in the form of markup document
- Result returned to client

# Form Handling

- Two ways the browser client can send information to the web server.
  - The `GET` Method
  - The `POST` Method
- What is difference????

# GET Method

http://www.w3schools.com/tags/ref_httpmethods.asp

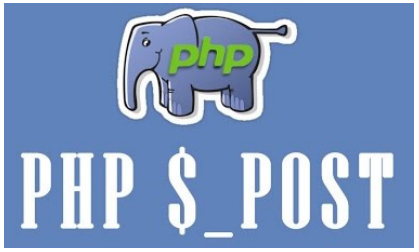

**Query strings (name/value pairs) is sent in the URL of a GET request**

- GET requests
  - can be cached
  - remain in the browser history
  - can be bookmarked
  - are visible to everyone
  - should never be used when dealing with sensitive data
  - have length restrictions (~ 2000 characters)
  - result in an implicit array $_GET

➔ Example: php1_get.php

# POST Method

http://www.w3schools.com/tags/ref_httpmethods.asp




**Query strings (name/value pairs) is sent in the HTTP message body of a POST request (so don't see it)**

- POST requests
  - are never cached
  - do not remain in the browser history
  - cannot be bookmarked
  - have no restrictions on data length
  - result in an implicit array $_POST

➔ Example: php1_post.php

# PHP commands

- **`empty()`**

  `empty(variable)` returns false if the variable exists and has a non-empty, non-zero value otherwise returns true.

  Following are considered *empty*:
  `""` (empty string), 0, 0.0, `"0"`, NULL, False, a variable declared but without a value.

- **`isset()`**

  `isset(variable1)` OR `isset(variable1, variable2…)`
  Returns true if set to any value except NULL, false otherwise.

  ```
  $var = 0;
  ```
  `isset($var)` will return true (unlike `empty()` )

# Processing a simple form


```
<body>
  <h1>Processing a Simple Form</h1>
  <form method="post" action="welcome.php">
    <label>Full Name: <input name="client_name"
        size="25" /></label>
    <br /><br />
    <label for="e">Email:</label>
    <input id="e" type="email" name="client_email“
         size="25" />
    <br /><br />
    <input type="reset" value="Clear Form" />
    <input type="submit" value="Send" />
  </form>
</body>
```

FormAction.html

# welcome.php (1/3)

```
<?php
    $title="A Warm Welcome";
    if (  empty($_POST['client_name']) ||
          empty($_POST['client_email']) )
    { $error=TRUE;
      $title="Please Go Back";
    }
?>


<!DOCTYPE html>
<html lang="en">
  <head><meta charset="utf-8"/>
  <title><?php echo $title; ?></title></head>


  ........
```

# welcome.php (2/3)

```
  ...
<body>
  <h1><?php echo $title; ?></h1>
  <?php if ( isset($error) ) {?>
     <p>Sorry, the form is incomplete.</p>
     <p>Please go back and fill out all the
       required entries.  Thank you.</p>
```

# welcome.php (3/3)

```
…
  <?php } else { ?>
     <p>Hello
     <?php echo $_POST['client_name']; ?>
     , it is our great pleasure to welcome you
to our site.</p>
     <p>We have your email address,
     <?php echo $_POST['client_email']; ?>,
         and we will contact you shortly.</p>
  <?php } ?>
</body></html>
```

# Form Handling

- A more complex example (from your textbook)
- → SHOW `popcorn3.html` & `popcorn3.php`

# File I/O in PHP

PHP provides a complete set of file and directory functions enabling you to easily access and manipulate files and folders on the local file system.

- **File tests**: `file_exists, is_dir, is_file, is_readable, is_writable, is_executable, filesize (in bytes)`

Syntax:

```
function_name($filename)
```

Demo: `file_io.php`

# File I/O in PHP ...

Syntax:

```
function_name($filename)
```

- **File status:**
- `fileatime (last access time of file),`
- `filectime (inode change time of file),`
- `filegroup (file group),`
- `filemtime (file modification time),`
- `fileowner (file owner),`
- `fileperms (file permissions),`
- `filesize (file size),`
- `filetype (file type)`

Demo: `file_io2.php`

# File I/O in PHP ...

- **File manipulation**:
  - `copy` (copies a file),
  - `unlink` (deletes a file),
  - `rename` (renames a file),

Demo: `file_io3.php`

# File I/O in PHP ...

- **File I/O**:
  - **fopen** (opens file for I/O, returns handle)

```
$file = fopen("test.txt","r");
$file = fopen("/SOEN287/slides/test.txt","r");
```

  - **fclose** (closes handle) returns TRUE or FALSE
  - **feof** (tests eof) returns TRUE or FALSE
  - **fflush** (writes all buffered output to open file) returns TRUE or FALSE

Complete: `file_io4.php`

# File I/O in PHP ...

- **File I/O**:
  - **`fwrite(file,string)`** (writes to handle)
    `fwrite($file, "Hello World!")`
    returns # of bytes written or FALSE if could not write

  - **`fread(file, length)`** (reads from handle)
    `fread($file, "50")` → read 50 bytes from $file
    `fread($file, filesize($file))` → read entire file
    returns read string of FALSE

  - **`readfile(file)`** (sends file to output buffer)
    returns # of bytes read or FALSE if error

Complete : `file_io4.php`

# File I/O in PHP ...

- `file(`*`name`*`)`: returns the lines of the given file into an array

- `file_get_contents(`*`name`*`)`: returns the file contents as a string.

- PHP file I/O work not only with file names, but also URLs to access remote files.

Complete: `file_io4.php`

# File I/O in PHP ...

- **Directory handling:**
  - `mkdir` (creates dir)
  - `rmdir` (removes dir)
  - `chdir` (changes dir)

  - `opendir` (opens dir, returns handle)
  - `closedir` (closes dir handle)
  - `...`

# More File-Related Functions

## To include other PHP files in a PHP script.

- **`include(`*`file`*`)/require(`*`file`*`)`**
- Difference in how errors handled. If an error occurs:
  - `include()` generates a warning, but the script will continue execution
  - `require()` generates a fatal error, and the script will stop.

- **`include_once(`*`file`*`)/require_once(`*`file`*`)`**
  - will only include file once – will check before including
  - important for files that have class or function definitions as can't define these twice.

# Example of 2 include()

- **test1.php**

```
<?php $var=20;
....
```

- **test2.php**

```
<?php
include('test1.php');
echo “$var </br/>;     // "prints" 20
$var = 30;
include('test1.php'); //if this line is removed second echo prints 30
echo “$var <br/>”;    // "prints" 20 again.
……
```