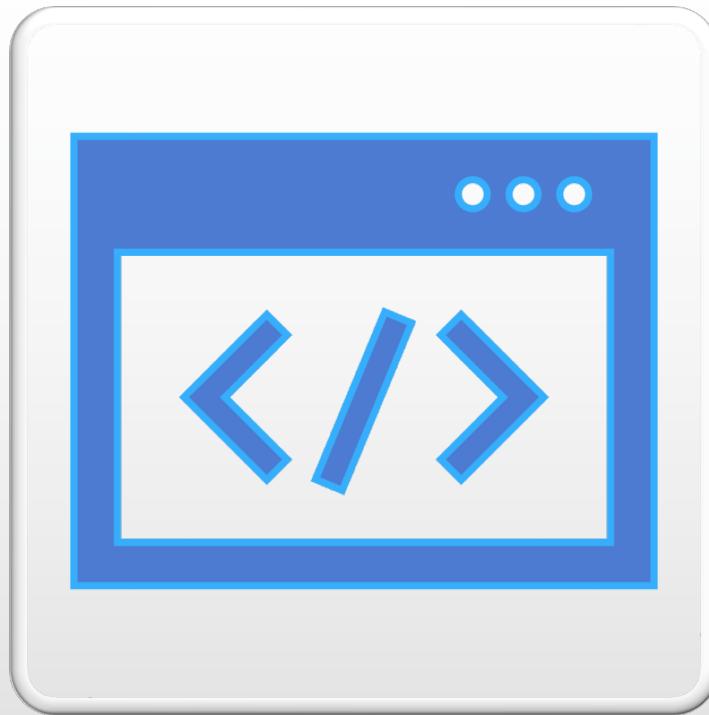


# **LECTURE07: PHP SESSIONS, COOKIES, AND FILES**

CS418/518: WEB PROGRAMMING

BY DR. JIAN WU

COURTESY: DR. JUSTIN BRUNELLE



# PHP SESSIONS

CS418/518

# THE IMPORTANCE OF SESSIONS

- A web service often involves multiple web pages
  - Online shopping: Log in, product browsing, shopping cart, payment, etc.
- How to remember “global” information involved in the service so users do not provide it at every operation?
  - User information, product information, etc.
- HTML pages forget variables
- PHP Session control allows users to keep information for later use
  - Information is lost after session is terminated

# COOKIES VS. SESSIONS

- Cookies

- Store information as files on local clients
- Require database procedures on the server side
- Have longer life span: months or even years

- Sessions

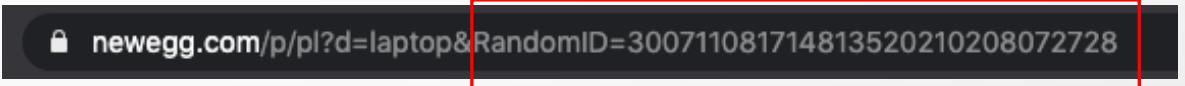
- Store information on the server side
- Require database procedures on the server side
- Have shorter life span: minutes to hours

# PHP SESSIONS

- Like a cookie, but variables/data not stored on the client
- Ends when browser/window is closed
- Persists across multiple pages

```
<?php  
session_start();  
  
print_r($_SESSION);  
?>
```

# HOW DO PHP SESSIONS WORK?

- Creating a unique identification (UID) for each user session
- Keeping the variables based on this UID
- Example from Newegg.com 

The screenshot shows a browser address bar with the URL `newegg.com/p/pl?d=laptop&RandomID=300711081714813520210208072728`. A red rectangular box highlights the long string of digits at the end of the URL, which represents the session ID.
- Warning:
  - Sessions in PHP have some security concerns because other people may obtain your session data
  - Be careful when using session control

# EXAMPLE: CREATE A SESSION VARIABLE

create\_session.php

```
<?php  
    session_start();  
  
    $_SESSION['userID'] = 'your ODU ID';  
  
    echo "Create a session variable<br>";  
    echo $_SESSION['userID'];  
  
?>
```

access\_session.php

```
<?php  
    session_start();  
  
    echo "Access the session variable <br>";  
    echo $_SESSION['userID'];  
?>
```

# SESSION CONTROL

- Use `session_start()` at the beginning of **every** php file that needs access to session variables
  - This function **registers** this session with the web server and gets a UID for the session
  - This function **initializes** the `$_SESSION` array to store data
- Session variables are stored in an array named as `$_SESSION`
  - You can specify keys and values like  
`$_SESSION['variable_name'] = value`

# ENDING SESSIONS

```
<?php  
//remove all session variables  
session_unset();  
  
//destroy the session  
session_destroy();  
?>
```

# SESSION CONTROL EXAMPLE

login.php

```
// Authenticate the user
if (authenticateUser($connection, $loginUserId, $loginPassword))
{
    //the user id and password match,
    // set the session
    $_SESSION['db_is_logged_in'] = true;
    $_SESSION['userID'] = $loginUserId;

    // after login we move to the main page
    header('Location: main.php');
    exit;
} else {
    $errorMessage = 'Sorry, wrong username / password';
}
```

# SESSION CONTROL EXAMPLE

main.php

```
//is the one accessing this page logged in or not?  
if (!isset($_SESSION['db_is_logged_in'])  
    || $_SESSION['db_is_logged_in'] != true) {  
    // not logged in, move to login page  
    header('Location: login.php');  
    exit;  
} else {  
    //logged in, display appropriate information  
    echo "Hello ", $_SESSION['userID'], "!";  
}
```



# PHP COOKIES

CS418/518

# PHP COOKIES

- File delivered to the browser from the server
- PHP can access the contents of the cookie
- To make (or set) a cookie:

`setcookie(name, value, expire, path, domain, secure, httponly)`

## `setcookie(name, value, expire, path, domain, secure, httponly)`

| Parameter            | Description  |
|----------------------|--|
| <code>name</code>    | The name of the cookie.  |
| <code>value</code>   | The value of the cookie. Do not store sensitive information since this value is stored on the user's computer.   |
| <code>expires</code> | The expiry date in UNIX timestamp format. After this time cookie will become inaccessible. The default value is 0.                                       |
| <code>path</code>    | Specify the path on the server for which the cookie will be available. If set to <code>/</code> , the cookie will be available within the entire domain. |
| <code>domain</code>  | Specify the domain for which the cookie is available to e.g <code>www.example.com</code> .   |
| <code>secure</code>  | This field, if present, indicates that the cookie should be sent only if a secure HTTPS connection exists.   |

If the expiration time of the cookie is set to 0, or omitted, the cookie will expire at the end of the session i.e. when the browser closes.

# RETRIEVE A COOKIE USING `$_COOKIE`

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
// 86400 = 1 day

if (!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named " . $cookie_name . " is not set!";
} else {
    echo "Cookie " . $cookie_name . " is set! <br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>
```

<https://stackoverflow.com/questions/6970754/why-are-my-cookies-not-setting>

**Play**

check-cookie-is-set-or-not.php  
get-cookie-value.php  
cookieexample/cookieexample.php

delete-a-cookie.php  
set-cookie.php

# MODIFY A COOKIE VALUE

- To modify a cookie, just set (again) the cookie using the `setcookie()` function:

this is modified

```
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), '/');
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

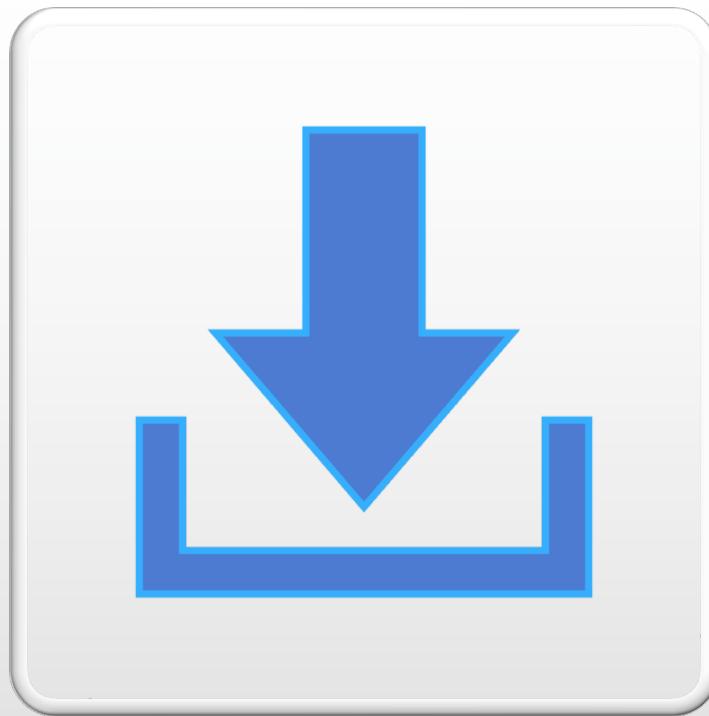
</body>
</html>
```

# REMOVING A COOKIE BY SETTING THE EXPIRATION DATE

```
<?php  
// set the expiration date to one hour ago  
setcookie("user", "", time() - 3600);  
?>
```

There is no special dedicated function provided in PHP to delete a cookie. All we have to do is to update the **expire-time** value of the cookie by setting it to a past time using the **setcookie()** function. A very simple way of doing this is to deduct a few seconds from the current time.

```
<?php  
if(count($_COOKIE)>0) {  
    echo "Cookies are enabled.";  
} else {  
    echo "Cookies are disabled.";  
}  
?>
```



FILE I/O

# INCLUDES & REQUIRES FILES

- Neither **include** or **require** are functions, they are **constructs**
- **include:** include the content of another PHP script and execute it.
  - When the file being included cannot be found, **include** will emit a warning (**E\_WARNING**) and the script will continue
  - Example: <?php include 'commonFile.php'; ?>
- **require:**
  - Copy the code in the file;
  - If not found, **require** will emit a fatal error (**E\_COMPILE\_ERROR**) and halt the script.
  - Example: <?php require 'commonFile.php'; '?>
  - Example: <?php require 'fileThatDoesntExist.php'; ?>
- Example file: [includeexamples/includeExample.php](#)

# FUNCTIONS

- Open files:
  - fopen(): open a file
- Reading files:
  - file(): Reads entire file into an array
  - fread(): reads up to length bytes from the file pointer referenced by stream. better for parsing data.
  - file\_get\_contents(): has no limit on the input to read the contents of a file into a string.
- Writing files:
  - fwrite(): write files with a limit (see <https://stackoverflow.com/questions/33500998/php-fwrite-for-writing-a-large-string-to-file>) If the file is too large (say >10MB), you need to split it.
  - file\_put\_contents(): identical to calling fopen(), fwrite() and fclose() successively to write data to a file.

# FILE I/O: FILE()

- `file()`: Reads entire file into an array
  - Example: `readfile.php`

```
<?php

// Get a file into an array.  In this example we'll go through HTTP to get
// the HTML source of a URL.
$lines = file('http://www.example.com/');

// Loop through our array, show HTML source as HTML source; and line numbers too.
foreach ($lines as $line_num => $line) {
    echo "Line #<b>{$line_num}</b> : " . htmlspecialchars($line) . "<br />\n";
}

// Another example, let's get a web page into a string.  See also file_get_contents
$html = implode('', file('http://www.example.com/'));

// Using the optional flags parameter since PHP 5
$trimmed = file('somefile.txt', FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);
?>
```

`implode` – Join array elements with a delimiter (an empty string in this case)

`FILE_IGNORE_NEW_LINES`,  
`FILE_SKIP_EMPTY_LINES` – options can trim off empty lines.

# FILE I/O: FOPEN() AND FREAD()

Open a file using **fopen()** function.

Get the file's length using **filesize()** function.

Read the file's content using **fread()** function.

Close the file with **fclose()** function.

fopenfile.php

```
<html>
  <head>
    <title>Reading a file using PHP</title>
  </head>

  <body>
    <?php
      $filename = "tmp.txt";
      $file = fopen( $filename, "r" );

      if( $file == false ) {
        echo ( "Error in opening file" );
        exit();
      }

      $filesize = filesize( $filename );
      $filetext = fread( $file, $filesize );
      fclose( $file );

      echo ( "File size : $filesize bytes" );
      echo ( "<pre>$filetext</pre>" );
    ?>

  </body>
</html>
```

# FILE I/O – FILE\_GET\_CONTENTS()

- `file_get_contents()`: read the entire file into a string
- `$text = file_get_contents($filename)`

```
<?php
$homepage = file_get_contents('http://www.example.com/');
echo $homepage;
?>
```

```
<?php
// Read 14 characters starting from the 21st character
$section = file_get_contents('./people.txt', FALSE, NULL, 20, 14);
var_dump($section);
?>
```

`FALSE`: `include_path`: optional, setting this parameter to search for the file in `include_path` directory

`NULL`: `context`: optional, specifying the context of the file handle. Can be skipped by using `NULL`.

More information: [https://www.w3schools.com/php/func\\_filesystem\\_file\\_get\\_contents.asp](https://www.w3schools.com/php/func_filesystem_file_get_contents.asp)

# FILE I/O: FWRITE()

`fwritefile.php (Part 1)`

```
<?php
    $filename = "/home/user/guest/newfile.txt";
    $file = fopen( $filename, "w" );

    if( $file == false ) {
        echo ( "Error in opening new file" );
        exit();
    }

    fwrite( $file, "This is a simple test\n" );
    fclose( $file );
?>
```

Note: if you cannot create newfile.txt, it is probably because apache does not have permissions to write into that directory. You may (1) create a directory called 'files' and change its ownership to apache. See Slide #26 for how.

```
<html>
    <head>
        <title>Writing a file using PHP</title>
    </head>

    <body>

        <?php
            $filename = "newfile.txt";
            $file = fopen( $filename, "r" );

            if( $file == false ) {
                echo ( "Error in opening file" );
                exit();
            }

            $filesize = filesize( $filename );
            $filetext = fread( $file, $filesize );

            fclose( $file );

            echo ( "File size : $filesize bytes" );
            echo ( "$filetext" );
            echo("file name: $filename");
        ?>

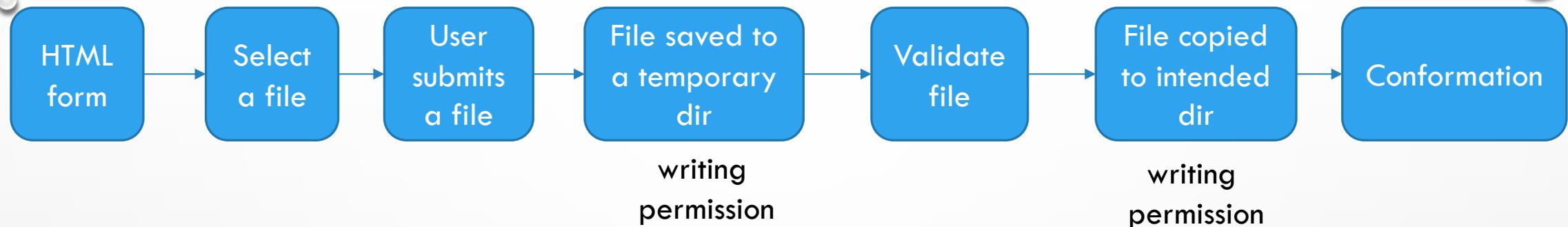
    </body>
</html>
```

# FILE I/O – WRITE

- `file_put_contents()`: write data to a file
- `file_put_contents($filename, $text);`
- Example: `file-put-content.php`

```
<?php
$file = 'people.txt';
// Open the file to get existing content
$current = file_get_contents($file);
// Append a new person to the file
$current .= "John Smith\n";
// Write the contents back to the file
file_put_contents($file, $current);
?>
```

# FILE UPLOADING



phpinfo.php

If sys\_temp\_dir is empty, you should be able to use /tmp as the temporary dir

|                           |          |          |
|---------------------------|----------|----------|
| sys_temp_dir              | no value | no value |
| track_errors              | Off      | Off      |
| unserialize_callback_func | no value | no value |
| upload_max_filesize       | 2M       | 2M       |

What permissions should we set? Two choices:

1. Quick and dirty: writable for everyone 777 (bad idea)
2. Owner of the Apache process.
  1. Use \$ ps aux | grep apache to find out the Apache **userid**
  2. Use \$ id [userid] to find out the **group**
  3. Use \$ chown -R **userid:group** to change the ownership of that folder.

```
<?php
if(isset($_FILES['image'])){
    $errors= array();
    $file_name = $_FILES['image']['name'];
    $file_size =$_FILES['image']['size'];
    $file_tmp =$_FILES['image']['tmp_name'];
    $file_type=$_FILES['image']['type'];
    $file_ext=strtolower(end(explode('.',$_FILES['image']['name'])));
}

$extensions= array("jpeg", "jpg", "png");

if(in_array($file_ext,$extensions)== false){
    $errors[]="extension not allowed, please choose a JPEG or PNG file.";
}

if($file_size > 2097152){
    $errors[]='File size must be exactly 2 MB';
}

if(empty($errors)==true){
    move_uploaded_file($file_tmp,"images/".$file_name);
    echo "Success";
} else{
    print_r($errors);
}
?>
```

uploadfile.php (Part 1)

**explode()** – Split a string by another string.  
**end** - returns the value at the **end** of the array

# UPLOAD FILES

visual effect (before uploading)

|             |                |        |
|-------------|----------------|--------|
| Choose File | No file chosen | Submit |
|-------------|----------------|--------|

visual effect (after uploading)

|         |             |                |        |
|---------|-------------|----------------|--------|
| Success | Choose File | No file chosen | Submit |
|---------|-------------|----------------|--------|

uploadfile.php (Part 2)

```
<html>
    <body>

        <form action="" method="POST" enctype="multipart/form-data">
            <input type="file" name="image" />
            <input type="submit"/>
        </form>

    </body>
</html>
```

# **`$_FILES`**

- Global variable
- Associate double dimension
- If the value assigned to the input's name attribute in uploading form was **file**, then PHP would create following five variables
- **`$_FILES['file']['tmp_name']`** – the uploaded file in the temporary directory on the web server.
- **`$_FILES['file']['name']`** – the actual name of the uploaded file.
- **`$_FILES['file']['size']`** – the size in **bytes** of the uploaded file.
- **`$_FILES['file']['type']`** – the MIME type of the uploaded file.
- **`$_FILES['file']['error']`** – the error code associated with this file upload.

# EXAMPLE: UPLOAD AN IMAGE AND SHOW FILE INFORMATION

uploadimage.php (Part 1)

```
<?php
if(isset($_FILES['image'])){
    $errors= array();
    $file_name = $_FILES['image']['name'];
    $file_size = $_FILES['image']['size'];
    $file_tmp = $_FILES['image']['tmp_name'];
    $file_type = $_FILES['image']['type'];
    $file_ext=strtolower(end(explode('.',$_FILES['image']['name'])));

    $extensions= array("jpeg", "jpg", "png");

    if(in_array($file_ext,$extensions)== false){
        $errors[]="extension not allowed, please choose a JPEG or PNG file.";
    }

    if($file_size > 2097152) {
        $errors[]='File size must be excately 2 MB';
    }

    if(empty($errors)==true) {
        move_uploaded_file($file_tmp,"images/".$file_name);
        echo "Success";
    }else{
        print_r($errors);
    }
}
?>
```

LIVE DEMO

uploadimage.php (Part 2)

```
<html>
<body>

<form action = "" method = "POST" enctype = "multipart/form-data">
    <input type = "file" name = "image" />
    <input type = "submit"/>

    <ul>
        <li>Sent file: <?php echo $_FILES['image']['name']; ?>
        <li>File size: <?php echo $_FILES['image']['size']; ?>
        <li>File type: <?php echo $_FILES['image']['type'] ?>
    </ul>

</form>

</body>
</html>
```

# DEBUGGING

- Echo statements
- Apache log files: usually located at `/var/log/apache2` (MacOS, Linux)



# **BACKUP SLIDES BEYOND THIS POINT**



CS418/518