CITS1231 Web Technologies
PHP Emails, Cookies and Session Control
Sending email with PHP

• We have looked at storing user information using files.

• Email messages can also be thought of as data streams, providing another means of retaining user information.

• Syntax: \[ \text{mail(to, subject, message, headers)} \]

<table>
<thead>
<tr>
<th>argument</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>to</td>
<td>Destination email account</td>
</tr>
<tr>
<td>subject</td>
<td>Text that appears in the subject line</td>
</tr>
<tr>
<td>message</td>
<td>Text that appears in the body of the email</td>
</tr>
<tr>
<td>headers</td>
<td>Multiple headers that are separated by a carriage return and a new-line character (\r\n).</td>
</tr>
</tbody>
</table>
Simple Example

```html
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<title>PHP Script</title>
</head>
<body>
<?php
$strEmail =  "wei@csse.uwa.edu.au";
$strSubject = "Hello";
$strMessage = "Hello Wei, how are you?";
$strHeaders = "From: wei@csse.uwa.edu.au\n";
$strHeaders .= "Reply-To: wei@csse.uwa.edu.au";
mail($strEmail, $strSubject, $strMessage, $strHeaders);
echo "<p>Message sent</p>";
?>
</body>
</html>
```

Sending (X)HTML email messages

- PHP allows to create more complex emails, instead of simply including raw text, we can insert (X)HTML code into the message part of the email.

- This allows for greater control over the look and feel of the email.

- However, not all email clients support (X)HTML emails and many individuals choose to turn off this facility.
HTTP is Stateless

- HTTP is a stateless protocol – cannot tell the difference between request from different users
- Cookies and Sessions can be used to track a single user during their visit to a website
- An old way is to use the hidden form control to relay data from one page to another.

<input type="hidden" name="user" value="wei">
Cookies

• Cookies are a mechanism for a web page to store data on the client computer.

• Because the cookie is available the next time the web page is visited, cookies can be used to track or identify returning users to a web page.

• User can control whether cookies can be written and thus you cannot guarantee that they are allowed.
Cookies are Stored on Client Computer

Client browser

Cookie

Data Communication

PHP Server
How to Create a Cookie

• **setcookie()** function:

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

• Only the name argument is mandatory. You can replace any argument with a empty string (""") to skip that argument.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the cookie file</td>
</tr>
<tr>
<td>value</td>
<td>Data to be stored in the cookie file</td>
</tr>
<tr>
<td>expire</td>
<td>Date string that defines the valid life time of the cookie</td>
</tr>
<tr>
<td>path</td>
<td>Subset of URLs in the domain for which the cookie is valid</td>
</tr>
<tr>
<td>domain</td>
<td>The domain for the which the cookie is available</td>
</tr>
<tr>
<td>secure</td>
<td>If set to “1” the cookie is only transmitted if the communication channel with the host is secure (https)</td>
</tr>
</tbody>
</table>
Setting a Cookie

• Cookie creation is part of the HTTP header.

• The setcookie() function must be called before any output is sent to the browser.

```php
<?php
    setcookie("TestCookie", "Hello, this is a cookie");
?>
```

```html
/html
</html>
```

setcookie() called before <html>
Reading a cookie

- The cookie variables could be accessed using the predefined \$_COOKIE array.
- Cookies do not become visible until the next loading of a page.
- Example:

```php
<?php
    $strCookieData = \$_COOKIE["TestCookie"];
    echo "<p>$strCookieData</p>";
?>

Name of cookie file previously created
Deleting a cookie

- Cookies can be deleted by simply using the `setcookie()` function with only the name of the cookie.

- Example:

  ```php
  setcookie("TestCookie");
  ```
Aside: Variables Inside Double Quotes

- If variables are inside double quotes, PHP will replace the variables with their string values.
- This is not true if you use single quotes.
- Example:

```php
<?php
    $a = "Hello";
    $b = "World";
    $strInfo = "\$a to my \$b!";
    echo $strInfo;
    $strInfo2 = '$a to my $b!';
    echo $strInfo2;
?>
```

Hello to my World!

$a to my $b!
Storing/Retrieving Multiple Items in a Cookie

```php
<?php
// $_SERVER and $_ENV are predefined
$strAddress = $_SERVER['REMOTE_ADDR'];
$strBrowser = $_SERVER['HTTP_USER_AGENT'];
$strOs = $_ENV['OS'];
$strInfo = "$strAddress::$strBrowser::$strOs";
setcookie("AnotherCookie", $strInfo);
?>
<html><head><title>PHP Script</title></head>
<body>
<?php
    echo "<p>Cookie Set</p>";
?>
</body></html>

<?php
$strReadCookie = $_COOKIE['AnotherCookie'];
$arrList = explode("::", $strReadCookie);
echo "<p>IP Address: $arrList[0] </p>
?>
```

- Concatenate multiple strings delimited by ‘::’
- Save this in a cookie
- explode($delimiter, $str) splits $str into substrings using $delimiter and returns array of these substrings
Sessions

• Sessions are similar to cookies in that they serve the same purpose – to preserve some data between pages on a website.

• However, sessions are stored on the server, therefore, more secure as there is no data being passed back and force between the client and the server.

• Sessions work even if a user has disabled cookies on the client browser.

• Sessions allow variables and their values to be stored for each and every user.
Sessions Are Stored On The Server

Client browser

Data Communication

PHP Server

Session file
How Do Sessions Work?

• Sessions work by assigning a visitor a unique id, known as a session id.

• The session id can be stored in a cookie or embedded as part of a URL.

• A session id looks something like this:
  
  `sess_f231be97d46fb1ca96c1323e88f4523f`

• On server, a session file is created with same name.

• The session file is used to store values of variables assigned to that session.
Starting a session

• The `session_start()` function checks whether a session has been created for this user and, if not, creates one. This function always return `true`.

• A session is only active as long as the current client’s browser is open.

• If a session exists, then all variables and their values are retrieved and available for use.

• Session variables are registered using the `$SESSION` associative array, for example:

```php
$_SESSION["intCount"] = 0;
```
Example: Visit Counter

```php
<?php
session_start();

if (!$_SESSION["intCount"])
    $_SESSION["intCount"] = 1;
else
    $_SESSION["intCount"]++;

?>
<html><head><title>PHP Script</title></head>
<body>

<?php
    echo "<p>You have accessed this page " . $_SESSION["intCount"] . " times.</p>";

?>
</body>
</html>
```

- Must be used to access session variables
- First visit to this page
- Subsequent visits
- Print out number of visits
Using a Session To Set Style

```php
<?php
session_start();

if (!$_SESSION['strBgCol'])
    $_SESSION['strBgCol'] = "blue";

if (!$_SESSION['strTextCol'])
    $_SESSION['strTextCol'] = "yellow";

$strBgCol = $_SESSION['strBgCol'];
$strTextCol = $_SESSION['strTextCol'];

…PHP Code Part 2 (see later) …

?>
```

Create/resume a session

Initialize session variables if first time

Initialise style variables based on session variables

We will return to this code later
Example Continued

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" 
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<title>PHP Script</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
<style type="text/css">
  body {background-color: <?php echo $strBgCol ?>}
  p {color: <?php echo $strTextCol ?>}
  h2 {color: <?php echo $strTextCol ?>}
</style>
</head>
Use previously defined style variables
Example Continued

<body>
<h2>What Colours would you like?</h2>
<form action='<?php echo $_SERVER["PHP_SELF"]?>' method='post'>
  <p>
    <label for="strNewBgCol">Background Colour: </label>
    <select name='strNewBgCol' id='strNewBgCol'>
      <option>red</option>
      <option>green</option>
      <option>blue</option>
      <option>cyan</option>
      <option>yellow</option>
    </select>
  </p>
</form>
</body>

Form action points to this page itself
Allow user to change background colour
Example Continued

```html
<p>
<label for="strNewTextCol">Text Colour: </label>
<select name='strNewTextCol' id='strNewTextCol'>
  <option>red</option>
  <option>green</option>
  <option>blue</option>
  <option>cyan</option>
  <option>yellow</option>
</select>
<input type='submit' name='submit'/>
</p>
</form>
</body>
</html>
```

Allow user to select text colour
Submit button
Now need to update the previously defined PHP style variables based on user selection!
Example Continued

```
if (isset($_POST['submit'])) {
    $strBgCol = $_POST['strNewBgCol'];
    $strTextCol = $_POST['strNewTextCol'];
    $_SESSION['strBgCol'] = $strBgCol;
    $_SESSION['strTextCol'] = $strTextCol;
}
```

This is PHP Code Part 2 (top of file)
Unsetting and Destroying Session Variables

• You can get the session id using:

   `session_id();`

• You can unset or destroy a session variable using:

   `unset ($_SESSION['myvar'])`

• To unset all session variables:

   `session_unset()`

• To completely destroy the session:

   `session_destroy()`
<?php
    session_start();
    $_SESSION['Greeting'] = 'Hello world!';
    echo "<p>Session1.php creates a session variable</p>";
    echo "<hr/>";
    echo "Session variable=".$_SESSION['Greeting']."</p>";
?>
<hr/>
<a href="Session2.php">Next page</a>
Session2.php

```php
<?php session_start();

echo "<p>Session2.php shows that the session variable still exists</p>";
echo "<hr/>";
echo "<p>Session variable=".$_SESSION['Greeting']."</p>";
echo "<hr/>";
echo "<p>Session variable is now unset</p>";
unset($_SESSION['Greeting']);

?>
<hr/>
<a href="Session3.php">Next page</a>
```

Resume with current session
Retrieve session variable
Destroy session variable
Link to next PHP page
Session3.php

```php
<?php
    session_start();
    echo "<p>Session3.php shows that the session variable no longer exists</p>";
    echo "<hr/>";
    echo "<p>Session variable=\$_SESSION[\'Greeting\'].</p>";
    echo "<hr/>";
    echo "<p>Session will now be destroyed</p>";
    session_unset();
    session_destroy();
?>
```

Session variable no longer exists

Destroy all variables

Destroyed session