Part 2: JavaScript, PHP, SQL, Advanced PHP

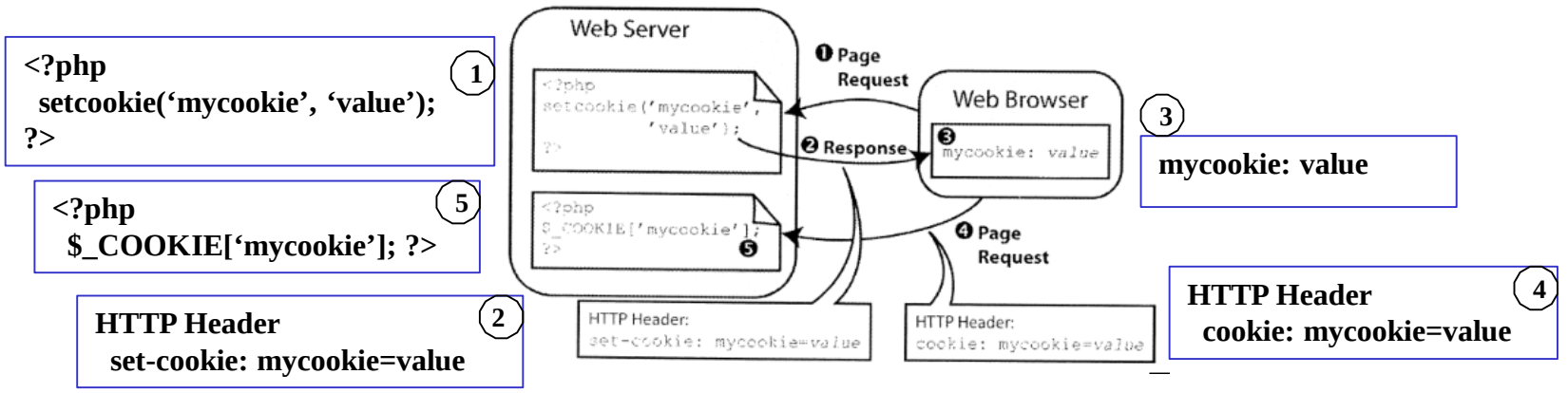
EE4717/IM4717 Web Application Design

Sessions

Lecturer :
Dr ANG Yew Hock
E-mail: iyhang@ntu.edu.sg
Tel: 67906361

Cookies: what is it?

- Why cookies
  - For preserving state across a number of transactions
- What is a cookie
  - A name/value pair associated with a given website and stored on the computer that runs the client (browser)
- Use of cookies
  - A small piece of information that PHP stores on a client-side machine
  - Browser when connected to an URL, searches cookies stored locally
  - Relevant cookie information are transmitted back to server



Cookies: use of it

- Setting cookies from PHP

```
setcookie (string name [, string value [, int expire
   [, string path [, string domain [, int secure]]]]])
```

  - **expire** sets date of expiry of cookie
  - **path** and **domain** specify the URL(s) for which cookie is relevant
    - E.g. path as '/~directory/', domain as '.domain.com'
  - **secure** requires cookie be sent over a secured HTTPS connection

- Using cookies

```
setcookie ('mycookie', 'value');
```

  - Once set by a website, all future page requests to the site will also include the cookie, until it expires
  - On reloads of current or other pages at the browser site, cookie variables will be available via…

```
$_COOKIE['mycookie']
```

  - Calling **setcookie()** again with only the name will delete the cookie

Cookies: using it

cookiecounter.php

- Set a cookie to expire in 1 year

```
setcookie('mycookie', 'somevalue', time() + 3600 * 24 * 365);
```

- Delete a cookie that has a preset expiry time

```
setcookie('mycookie', '', time() - 3600 * 24 * 365);
```

- Cookies must be set before any page content is output
  - Cookie is not actually set until the browser receives the web page
- Never assume cookies to be retained at website
  - Cookies are best used for logging in a user
    - If cookie is deleted, user simply has to reenter the user and password
- Browsers place a limit on number and size of cookies allowed per website
- Use sessions to overcome these issues

Session Control in PHP

- Why session control?
  - HTTP is stateless, no way to transactions across different pages
  - Cookies not appropriate for storing large amounts of information
- What is a session in PHP
  - Sessions let you store data on your web server
  - Is a superglobal, `$_SESSION`
- Session ID
  - Session ID generated by PHP
    - Unless configured otherwise, PHP session automatically sets in the user's browser a cookie that contains the session ID
  - Stored on client side for the lifetime of a session
    - Store a single cookie of the user's session ID
    - Session ID acts as a key to register session variables
  - Content of variables stored at the server
    - PHP keeps track registered variables in each session, and their values

Storing the Session ID

- PHP uses cookies by default with sessions in `php.ini` file

```
Session.use_cookies    = 1
```

  - If possible, a cookie will be set to store the session ID
- PHP may send session ID via URL query string variable
  - PHP automatically add the session ID to all relative links on your page via URL as in $_GET for data.
  - However, all pages must be PHP files for this to work and `session.use_trans_sid` must be enabled in `php.ini` file
  - Caution: setting `session.use_trans_sid` increases site's security risks as session ID in URL could be stored or bookmarked and become accessible by others. By default it is turned off

```
Session.use_trans_sid  = 0
```

- Alternatively, session ID can be manually embedded in links
  - <a href="link.php? **`<?php echo SID; ?>`** ">
  - `session ID` is stored in the constant SID
  - SID will evaluate to NULL if `Session.use_cookie` has been set to 1
  - use `session_id()` instead

Implementing Simple Sessions

- The basic steps

1. Starting a session
2. Registering session variables
3. Using session variables
4. De-registering variables and destroying the session

- Session functions

```
Session_start(); //start a new session
$_SESSION['pwd']='mypassword'; //$_SESSION array assignment
Session_destroy(); //end and delete all registered variables
```

Steps in using Sessions

- Starting a session

  `session_start();`

  - If not already a session, create one
  - If already exists, loads the registered session variables
- Registering session variables

  `$_SESSION['myvar'] = 5;`

  - Session variables are stored in superglobal array
  - Session variable will be tracked until session ends or manually `unset`
- Using session variables

  `If (isset($_SESSION['myvar']))…`

  - Checking if session variables have been set (or by `empty()`)
  - Note that variables can also be set by the user via GET or POST methods
- Unsetting variables and destroying the session

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

A Simple Session Example – Page 1

page1.php

- Start a session and create the variables `$_SESSION['sess_var']` and `$_SESSION['sess_var2']`
- var dump shows no variables stored in `$_SESSION` array
- Final value of variables on the page will be available on subsequent pages 2 and 3
- Session variable are frozen until they are reloaded via `session_start();`

```php
<?php //page1.php
  session_start();
  var_dump($_SESSION);
  $id=session_id();
  echo "<br>Session id in page 1= $id <br>";

  $_SESSION['sess_var'] = "Hello world!";
  $_SESSION['sess_var2'] = "Hello world2!";

 echo '<br>The content of $_SESSION[\'sess_var\'] is: '
       .$_SESSION['sess_var'].'<br />';
 echo 'The content of $_SESSION[\'sess_var2\'] is: '
       .$_SESSION['sess_var2'].'<br />';
?>
<a href="page2.php">Next page</a>
```

```
array(0) { }
Session id in page 1= kags03b0dn67ucmdoedapltn73

The content of $_SESSION['sess_var'] is: Hello world!
The content of $_SESSION['sess_var2'] is: Hello world2!
Next page
```

A Simple Session Example – Page 2

page2.php

- After session starts the variables `$_SESSION['sess_var']` and `$_SESSION['sess_var2']` are available with previously stored values
- var dump shows current variables stored in the `$_SESSION` array
- After unsetting a variable, the session still exist
- The session variables are passed along to page 3

```php
<?php //page2.php
  session_start();
  var_dump($_SESSION);
  $id=session_id();
  echo "<br>session id in page 2 = $id <br>";

  echo '<br>The content of $_SESSION[\'sess_var2\'] is '
       .$_SESSION['sess_var2'].'<br />';

  unset($_SESSION['sess_var2']);
?>
<a href="page3.php">Next page</a>
```

```
array(2) { ["sess_var"]=> string(12) "Hello world!"
["sess_var2"]=> string(13) "Hello world2!" }
session id in page 2 = kags03b0dn67ucmdoedapltn73

The content of $_SESSION['sess_var2'] is Hello world2!
Next page
```

A Simple Session Example – Page 3

page3.php

- The unset variable `$_SESSION['sess_var2']` no longer available whereas the persistent value of `$_SESSION['sess_var']` remained
- var dump shows remaining variable `$_SESSION['sess_var']` still available in the `$_SESSION` array
- Session ID no longer exist after `session_destroy()`

```php
<?php  //page3.php
session_start();
var_dump($_SESSION);
 $id=session_id();
 echo "<br>Session id in page 3 = $id <br>";

 echo '<br>The content of $_SESSION[\'sess_var\'] is: '
      .$_SESSION['sess_var'].'<br />';
 echo 'The content of $_SESSION[\'sess_var2\'] is: '
      .@$_SESSION['sess_var2'].'<br />';

 session_destroy();
  $id=session_id();
 echo "<br>Session id after destroy in page 3 = $id <br>";
?>
```

```
array(1) { ["sess_var"]=> string(12) "Hello world!" }
Session id in page 3 = kags03b0dn67ucmdoedapltn73

The content of $_SESSION['sess_var'] is: Hello world!
The content of $_SESSION['sess_var2'] is:

Session id after destroy in page 3 =
```

Implementing A Simple Shopping Cart

- The basic steps

1. Starting a session
2. Registering session variables
3. Using session variables
4. De-registering variables and destroying the session

- Pass variables to another page via URL

- Two PHP scripts:
  - A product catalogue, `catalogue.php`
  - A checkout page, `cart.php`

A Simple Shopping Cart - catalogue.php(1)

catalogue.php

- Session start
  - `session_start` either starts a new session and sets the session ID cookie
  - Or, restores variables registered in existing session, if one exists
  - Initialize `$_SESSION['cart']to an empty array()` if not already exist
  - On detecting `$_GET['buy']`, add item to `$_SESSION['cart']` array
  - Reload current page with `$_SERVER['PHP_SELF']`
    - header() function sends a HTTP header to the client
    - Redirect the page to a specific URL location; in this case the current page
    - SID contains session ID when cookies is disabled, otherwise NULL when cookies is enabled.

```php
<?php //catalogue.php
session_start();
if (!isset($_SESSION['cart'])) {
    $_SESSION['cart'] = array();
}
if (isset($_GET['buy'])) {
$_SESSION['cart'][] = $_GET['buy'];
header('location: ' . $_SERVER['PHP_SELF'] . '?' . SID);
exit();
}
?>
```

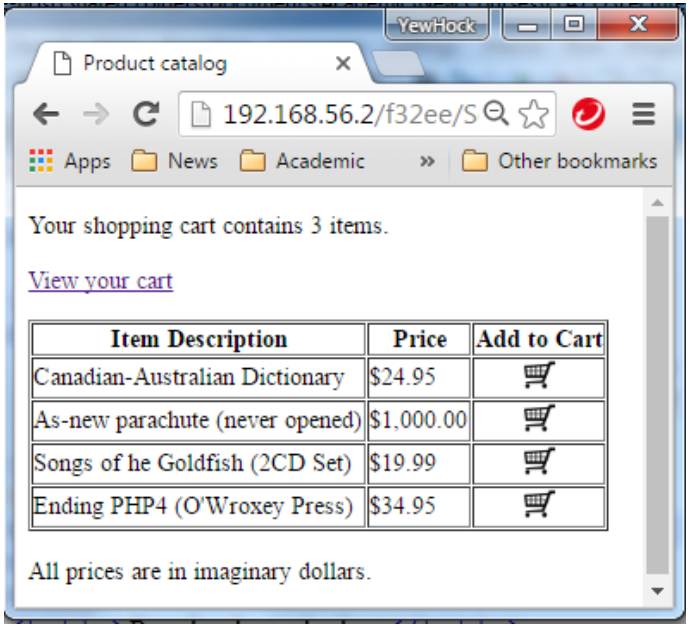
A Simple Shopping Cart - catalogue.php(2)

- Catalogue of items
  - Catalogue of items and prices in table simulate database records
- View your cart
  - Link to `cart.php`

```
<html>
<head>
<title>Product catalog</title>
</head>
<body>
<p>Your shopping cart contains <?php
    echo count($_SESSION['cart']); ?> items.</p>
    <p><a href="cart.php">View your cart</a></p>
    <?php
    $items = array(
        'Canadian-Australian Dictionary',
        'As-new parachute (never opened)',
        'Songs of the Goldfish (2CD Set)',
        'Ending PHP4 (O\'Wroxey Press)');
    $prices = array (24.95, 1000, 19.99, 34.95);
    ?>
```

A Simple Shopping Cart - catalogue.php(2)

- Catalogue table
  - Each product has a link back to the catalogue with `buy=idx` in the query string
  - `Idx` saved to `$_SESSION['cart']` **via** `$_GET['buy']`




```
<table border="1">
    <thead>
    <tr>
        <th>Item Description</th>
        <th>Price</th>
        <th>Add to Cart</th>
    </tr>
    </thead>
    <tbody>
<?php
for ($i=0; $i<count($items); $i++){
    echo "<tr>";
    echo "<td>" .$items[$i]. "</td>";
    echo "<td>$" .number_format($prices[$i], 2)."</td>";
    echo "<td align=\"center\">
        <a href=".$_SERVER['PHP_SELF']."?buy=" .$i. ">
        <img src=./images/cart.png width=20 ></a></td>";
    echo "</tr>";
}
?>
    </tbody>
</table>
<p>All prices are in imaginary dollars.</p>
</body>
</html>
```

A Simple Shopping Cart – cart.php(1)

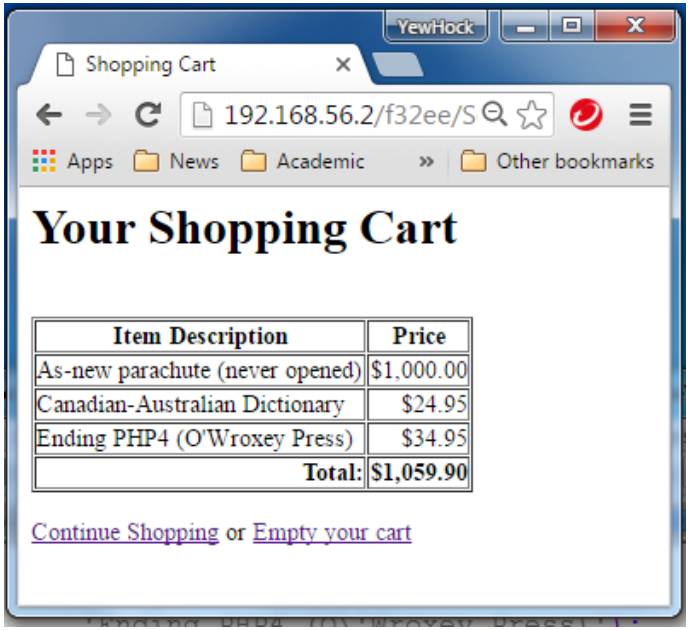
cart.php

- Session start
  - Similar to catalogue.php
- Checkout page
  - On `$_GET['empty']` unset `$_SESSION['cart']` variable; thus emptying shopping cart
  - Use the numbers stored in `$_SESSION['cart']`[] variable to print out the corresponding items from $items array

```php
<?php //cart.php
session_start();
if (!isset($_SESSION['cart'])) {
    $_SESSION['cart'] = array();
}
if (isset($_GET['empty'])) {
//Empty the $_SESSION['cart'] array
unset($_SESSION['cart']);
header('location:'.$_SERVER['PHP_SELF']. '?'. SID);
exit();
}
?>
<html>
<head>
<title>Shopping cart</title>
</head>
<body>
<h1>Your Shopping Cart</h1>
<?php
$items = array(
        'Canadian-Australian Dictionary',
        'As-new parachute (never opened)',
        'Songs of the Goldfish (2CD Set)',
        'Ending PHP4 (O\'Wroxey Press)');
    $prices = array (24.95, 1000, 19.99, 34.95);
?>
```

A Simple Shopping Cart – cart.php(2)

- Checking out
  - Compute total cost of items in shopping cart.
- Continuing shopping
  - Return to catalogue.php to continue shopping without changing the session context




```
?>
    <table border="1">
    <thead>
    <tr>
        <th>Item Description</th>
        <th>Price</th>
    </tr>
    </thead>
    <tbody>
    <?php
    $total = 0;
        for ($i = 0; $i < count($_SESSION['cart']); $i++) {
        echo '<tr>';
        echo '<td>'.$items[$_SESSION['cart'][$i]].'</td>';
        echo '<td align="right">$';
        echo number_format($prices[$_SESSION['cart'][$i]],2);
        echo '<td>';
        echo '</tr>';
        $total = $total + $prices[$_SESSION['cart'][$i]];
        }
    ?>
    </tbody>
    <tfoot>
        <tr>
            <th align="right">Total:</th><br>
            <th align="right">$<?php echo number_format($total, 2); ?>
            </th>
        </tr>
    </tfoot>
    <table>
    <p><a href="catalogue.php">Continue Shopping</a> or
    <a href="<?php echo $_SERVER['PHP_SELF']; ?>?empty=1">Empty your cart</a></p>
    </table>
</body>
</html>
```

Passing Hidden Element Variables to Other Pages

- Form variables can be passed to other pages using the $_GET() or $_POST() methods
- Variables of hidden elements are not displayed in the form, but their values may be passed through the superglobals

```
<form method=post
    action="<?php
     echo $_SERVER['PHP_SELF']; ?>">
     <input type=hidden name=elmname
          value=" <?php echo $elmvalue; ?>" >
     <input type=submit name=submit value="Input Label">
     <input type=reset name=reset value="reset">
</form>
```

Implementing Membership Registration and Login

- A common use of session control to keep track of users after they have been authenticated via a login mechanism.
- Combine authentication from MySQL database, and sessions for login functionality and page control

- Five scripts
  - Membership registration form, `registration.html`
  - Membership registration, `register.php`
  - Members authentication, `authmain.php`
  - Members-Only information page, `members_only.php`
  - Members logged out page, `logout.php`

Member Registration – registration.html

- Form action handled by `register.php`
- Global variables handled by POST method

```
<html>
<head>
    <title>Registration Page</title>
</head>
<body>
<h1><font color="blue">Member Registration</font></h1>
<form action="register.php" method=POST>
Username:<br />
<input type=text name=username><br /><br />
Password:<br />
<input type=password name=password><br /><br />
Password confirmation:<br />
<input type=password name=password2><br /><br />

<input type=submit name=submit value=Submit>
<input type=reset name=reset value="Reset">
</form>
</body>
</html>
```

Member Registration – register.php

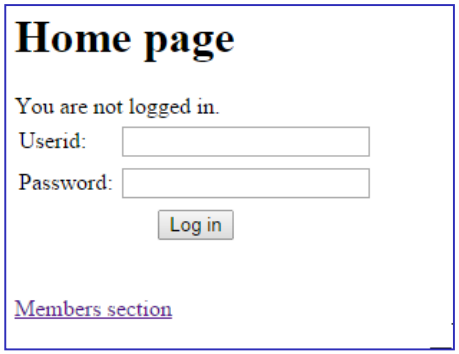
- Include `dbconnect.php`
  - Allow single point update on database connection: domain, user, password, database name.
- On 'submit', POST input form variables
  - Password is encrypted using the MD5 function for 120-bit encryption
- Username and encrypted password are stored in "users" table

```php
<?php  // dbconnect.php
@$dbcnx = new mysqli('localhost',
        'f32ee', 'f32ee', 'f32ee');
if ($dbcnx->connect_error){
    echo "Database is not online";
    exit;
    }
?>
```

```php
<?php // register.php
include "dbconnect.php";
if (isset($_POST['submit'])) {
    if (empty($_POST['username']) || empty ($_POST['password'])
        || empty ($_POST['password2']) ) {
    echo "All records to be filled in";
    exit;}
    }
$username = $_POST['username'];
$password = $_POST['password'];
$password2 = $_POST['password2'];

// echo ("$username" . "<br />". "$password2" . "<br />");
if ($password != $password2) {
    echo "Sorry passwords do not match";
    exit;
    }
$password = md5($password);
// echo $password;
$sql = "INSERT INTO users (username, password)
        VALUES ('$username', '$password')";
//  echo "<br>". $sql. "<br>";
$result = $dbcnx->query($sql);

if (!$result)
    echo "Your query failed.";
else
    echo "Welcome ". $username . ". You are now registered";

?>
```

Login Authentication with Session Control

- Three files:
  - authmain.php
  - members_only.php
  - logout.php
- First login
  - Login form in `authmain.php` posts "`userid`" and "`password`"
  - SQL query validation



- $_SESSION['valid_user'] tracks login state

```php
<?php //authmain.php
include "dbconnect.php";
session_start();

if (isset($_POST['userid']) && isset($_POST['password']))
{
  // if the user has just tried to log in
  $userid = $_POST['userid'];
  $password = $_POST['password'];

$password = md5($password);
  $query = 'select * from users '
           ."where username='$userid' "
           ." and password='$password'";

  $result = $dbcnx->query($query);
  if ($result->num_rows >0 )
  {
    // if they are in the database register the user id
    $_SESSION['valid_user'] = $userid;
  }
  $dbcnx->close();
}
?>
```

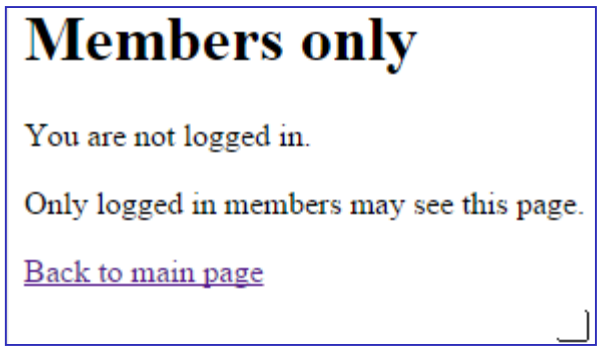
Login Authentication (2) – unsuccessful login

- If not authenticated user
  - return to members' home page and display failed to log in message
  - Allow re-login on same page


```
<html>
<body>
<h1>Home page</h1>
<?php
  if (isset($_SESSION['valid_user']))
  {
    echo 'You are logged in as: '.$_SESSION['valid_user'].' <br />';
    echo '<a href="logout.php">Log out</a><br />';
  }
  else
  {
    if (isset($userid))
    {
      // if they've tried and failed to log in
      echo 'Could not log you in.<br />';
    }
    else
    {
      // they have not tried to log in yet or have logged out
      echo 'You are not logged in.<br />';
    }
```

Login Authentication (3) – not logged in

- If not logged in and attempt to enter members_only page
  - results in “Members Only” message
  - Allow user to return to main page




```
    // provide form to log in
    echo '<form method="post" action="authmain.php">';
    echo '<table>';
    echo '<tr><td>Userid:</td>';
    echo '<td><input type="text" name="userid"></td></tr>';
    echo '<tr><td>Password:</td>';
    echo '<td><input type="password" name="password"></td></tr>';
    echo '<tr><td colspan="2" align="center">';
    echo '<input type="submit" value="Log in"></td></tr>';
    echo '</table></form>';
  }
?>
<br />
<a href="members_only.php">Members section</a>
</body>
</html>
```

Login Authentication (4) – members page

- If authenticated user
  - Go to members' home page and display username in logged in members page
- At members section,
  - allow to go back to home page
- At home page, allow return to members section or log out

```php
<?php //members_only.php
session_start();

echo '<h1>Members only</h1>';

// check session variable

if (isset($_SESSION['valid_user']))
{
  echo '<p>You are logged in as '.$_SESSION['valid_user'].'</p>';
  echo '<p>Members only content goes here</p>';
}
else
{
  echo '<p>You are not logged in.</p>';
  echo '<p>Only logged in members may see this page.</p>';
}

echo '<a href="authmain.php">Back to main page</a>';
```

Members only

You are logged in as Henry

Members only content goes here

Back to main page

Home page

You are logged in as: Henry
Log out

Members section

Login Authentication (5) – logout page

- At logout page
  - Store local variable before unset session variable and destroy session
  - Display logged out message according to login status
  - Allow return to main page


```php
<?php //logout.php
  session_start();

  // store to test if they *were* logged in
  $old_user = $_SESSION['valid_user'];
  unset($_SESSION['valid_user']);
  session_destroy();
?>
<html>
<body>
<h1>Log out</h1>
<?php
  if (!empty($old_user))
  {
    echo 'Logged out.<br />';
  }
  else
  {
    // if they weren't logged in but came to this page somehow
    echo 'You were not logged in, and so have not been logged out.<br />';
  }
?>
<a href="authmain.php">Back to main page</a>
</body>
</html>
```

Sending Emails

hellomail.php

- Setup the following:
  - $to
  - $from
  - Reply-To
  - Return mails (-f)

- Sending emails using mail() with return email address.

Mail($to, $subject, $message, $headers,'-ff32ee@localhost') ;

- Read emails on server at 192.168.56.2:2000

```
<!DOCTYPE html>
<html>
<body>
<h1>Hello Mail</h1>
<p>My first mail test.</p>

<?php
$to      = 'f32ee@localhost';
$subject = 'the subject';
$message = 'hello from php mail';
$headers = 'From: f32ee@localhost' . "\r\n" .
    'Reply-To: f32ee@localhost' . "\r\n" .
    'X-Mailer: PHP/' . phpversion();

mail($to, $subject, $message, $headers,
                                    '-ff32ee@localhost');
echo ("mail sent to : ".$to);
?>

</body>
</html>
```

Exercise

- Unzip the authentication.zip file and make necessary changes to one or more of the files, in order that the application may run successfully:
  - authmain.php
  - members_only.php
  - logout.php
  - createauthdb.sql
- Edit the multipurpose page such that the form action may be handled by $_SERVER superglobal variables rather than using explicit file name.

```
    // provide form to log in
    echo '<form method="post" action="authmain.php">';
    echo '<table>';
    echo '<tr><td>Userid:</td>';
    echo '<td><input type="text" name="userid"></td></tr>';
    echo '<tr><td>Password:</td>';
    echo '<td><input type="password" name="password"></td></tr>';
    echo '<tr><td colspan="2" align="center">';
    echo '<input type="submit" value="Log in"></td></tr>';
    echo '</table></form>';
  }
?>
<br />
<a href="members_only.php">Members section</a>
</body>
</html>
```