Introduction to multi-page PHP applications; Functions

Page 1

Introduction to Multi-Page PHP applications; Functions

- GET requests and query strings
- Hidden form fields
- GET vs POST requests
- Functions

Page 2

Query Strings

- We've already looked at sending information to a PHP script using an HTML **form**
- You may have noticed that the form data is added onto the end of the URL when the form is submitted
- e.g.

```
http://server/searchresults.php?artist=Oasis
```

- This information is called a **query string**
- In this example, we are sending a piece of information called **artist** to the PHP script **searchresults.php**, which has the value **Oasis**
- You could actually run the script directly (without going through the form) by manually adding the query string onto the URL yourself
- Query strings can be described as **name-value pairs**: the piece of information has a **name**, artist, and a **value**, Oasis
    - The general form of a URL containing a query string is:

```
http://server/script.php?name=value
```

Page 3

Query Strings without Forms

- Query strings can be useful in web applications even when forms are not involved
- Imagine a typical search results page, in which the user is searching for films
- Each search result has a "Book" link, which links to a "book.php" script to allow the user to book that film
- The "book" script would book the film using an UPDATE statement to reduce the number of tickets
- It needs to know which film we're buying, so we need to pass the ID of the film through
- This is done by adding the ID of the film to the URL of "book.php" as a query string
- **To do this, we need to write out the query string dynamically from PHP, so that it contains the ID of the film from the database**

Page 4

Maintaining State

- The previous scenario is an example of the more general concept of **maintaining state**
- Maintaining state is the concept of preserving information from page to page
- In the previous example, the ID of the film was passed from the search results script to the book script

Page 5

Reading information from a query string

- Use **$_GET** with the query string variable name to read the query string value in
- For example, for this query string:

```
<a href="lecture.php?week=11">
```

you would read in the query string variable **week** with **$_GET["week"]**, e.g.:

```
$w = $_GET["week"];
```

Page 6

Generating a query string from PHP

- In many cases, the query string is itself dynamically generated from PHP
- A common example of this is where the query string contains information from the database, such as the ID of a record
- This frequently happens on e-commerce sites where we want to give the user the opportunity to buy a product after searching for it
- The approach here would be to write a "Buy" link for each search result, linking to a buy script and passing across the ID through a query string
- **Incomplete** example using the student database from last week :

```
<!DOCTYPE html>
<html>
<body>
<?php

$a = $_GET["course"];
$conn = new PDO("mysql:host=localhost;dbname=student_records;", "username","password");


$results = $conn->query("select * from students where course='$a'");
while($row=$results->fetch())
{
    echo "<p>";
    echo "Student ID " . $row["ID"] ." <br/> ";
    echo "Name " . $row["name"]  . "<br/>";
    echo "Phone ".  $row["phone"] . "<br/>";
    echo "<a href='showmarks.php?QUERYSTRINGNAME=" . QUERYSTRINGVALUE . "'>Show Marks</a>";
    echo "</p>";
}


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

- Note how we **write out the link to the showmarks.php script** (which will show the marks for that student) and **the link contains the query string**
- To avoid making the lab exercise too easy, I haven't shown you the actual query string you need to write!
  - As in the previous example, the query string needs to have a **name** and a **value**
  - In the complete version, you would replace **QUERYSTRINGNAME** and **QUERYSTRINGVALUE** with something else!
- In this way, we pass the ID of the student over to the **showmarks.php** script

Page 7

Passing more than one piece of information in a query string

- We can pass **more than one** piece of information in a query string
- We do this by separating each item with a **& sign**
- Each piece of information has a name and a value

- e.g:

```
http://edward/lecture.php?week=11&unit=dfti
```

- We are passing to lecture.php **two pieces of information** with the names **week** and **unit** and the values **11** and **dfti**
- These would be read from PHP with **$_GET["week"]** and **$_GET["unit"]** respectively
- The general form of a query string with more than one piece of information is:

```
http://server/script.php?name1=value1&name2=value2&...nameN=valueN
```

Page 8

Hidden form fields

- Hidden form fields are another way in which we can pass information from one page to the next
- We add an extra field to a form which passes information through that the user cannot change
- e.g.

```
<input type="hidden" name="name" value="Tim Smith" />
```

- Typically, we would echo out a form from within a PHP script, and embed values from the previous page in the form as hidden fields

Page 9

Hidden form fields: example

- Imagine you have a two-step event booking application in which the user enters the event and number of tickets on the first page, and their name and phone number on the second
- The information in this form is then sent to the final script, which actually does the booking
- Since this final script requires the event name and number of tickets (read from the confirmation page) as well as the customer name and phone number, we need to pass this information across too
- Hidden fields would be an ideal solution here
- To do this we can echo out the whole form from within PHP
- See this example

Page 10

Hidden form fields in PHP: example

```
<?php

// Read event and tickets from the previous page
$a = $_POST['event'];
$b = $_POST['tickets'];

// Echo the form, with hidden fields containing the event and number of tickets

echo "<form method='post' action='bookevent.php' >";
echo "Your name: <input name='name' />";
echo "Your phone number: <input name='phone' />";
echo "<input type='hidden' name='event' value='$a' /> ";
echo "<input type='hidden' name='tickets' value='$b' /> ";
echo "<input type='submit' value='Go!' />";
?>
```

See here: the page with the hidden fields is the third page

Page 11

Using GET and POST correctly in multi-page applications

- Technically, a script which uses a query string to send an ID across to a "buy" or "book" script is violating the GET/POST usage rules
  - Query strings use GET requests, and we are communicating with a script which **changes** something, i.e. reduces the quantity in stock of the song
  - Really we should use POST
  - However we're doing it this way initially to gently introduce query strings
- A more correct and realistic approach would work as follows:
  - The search results page links (using a query string) to a **confirmation page**, passing the ID of the item over via a query string;
  - The confirmation page displays information about the item being bought and asks the user to confirm;
  - This confirmation page contains a form with a method of **"post"** which sends a request to the actual "buy" or "book" script, using a hidden field to pass on the ID and a regular field for the quantity of the item desired by the user

---

Page 12

Functions

- In addition to query strings and hidden fields, this week provides a convenient opportunity to introduce **functions**
- Functions are reusable modules of code, which you can write once and then re-use again and again

---

Page 13

Example: writing out a sidebar

We can use a PHP function to write a sidebar, then call the function on all of our pages

```
<?php
function write_sidebar()
{
    ?>
    <div id='sidebar'>
    Sidebar content
    </div>
    <?php
}
?>
```

- **write_sidebar()** is the function name
- Notice also how in this code we come out of PHP, write some straight HTML, then go back into PHP again

---

Page 14

Where do we write the functions?

- Functions go in a separate PHP file called the **include file**
- We write them once in this file, then we can import the file into any other PHP script, allowing us to reuse the functions over and over again

---

Page 15

Calling the functions

- Having written our functions, we can then **call** them from the main block of PHP code
- e.g.

```
<html>
<body>
<?php

include('functions.php');

```

```
write_sidebar();
echo "<div id='main'>";
echo "Here is the main part of the page";
echo "</div>";
?>
</body>
</html>
```

- The **include('functions.php')** line includes the functions.php file, which is the file containing our functions above
- The code then calls the function itself: **write_sidebar()** to write out the sidebar

Page 16

Example 2: function to connect to a database

```
<?php
function connect()
{
    $connection=new PDO("mysql:host=localhost;dbname=mydb","username","password");
    return $connection;
}
?>
```

- Note how here, we **return** the connection variable $connection to the outside world, so that it can be used outside the function
- We would call the function in this way:

```
$conn=connect();
```

Notice how we are assigning the **returned** value from the function ($connection) to the variable $conn

Page 17

Functions Example 3: Parameters

- Sometimes a function needs information from outside, in order to do its job
- For example, imagine a function which displays a message a certain number of times - it needs to know **how many times** to display the message
- We use a **parameter** to tell the function this information
- A **parameter** is a piece of information sent to a function from outside
- e.g.:

```
function write_message($ntimes)
{
    $i = 1;
    while ($i <= $ntimes)
    {
        echo "Hello! <br/> ";
        $i++;
    }
}
```

- Example of calling the function (to display the message 3 times):

```
write_message(3);
```

- The parameter **$ntimes** takes on the value passed in, i.e. 3 here

Page 18

Multiple parameters

- We can pass more than one parameter to a function, by separating them with a comma, e.g:

```
function write_message($message,$ntimes)
{
    $i = 1;
    while ($i <= $ntimes)
    {
        echo "$message <br/> ";
        $i++;
    }
}
```

- In this example, both the message and the number of times to display it are passed to the function
- Example of calling the function:

```
write_message("Hello!", 5);
```
