SWINBURNE UNIVERSITY OF TECHNOLOGY

Advanced Web Development: Managing State Information and Security

Week 9

Outline

- Understanding state information
- Saving stating information
  - Using hidden form fields to save state information
  - Using query strings to save state information
  - Using cookies to save state information
  - Using sessions to save state information
- Understanding PHP Security Issues

Reading: Textbook Chapter 9 & Appendix D

PHP: Cookies, Sessions, and Authentication

http://php.net/manual/en/features.php

http://php.net/manual/en/book.session.php

UNDERSTANDING STATE INFORMATION

Understanding State Information

- Information about individual visits to a Web site is called **state information**
- HTTP was originally designed to be **stateless** – Web browsers store no persistent data about a visit to a Web site
- **Maintaining state** means to store persistent information about Web site visits, that can be passed backwards and forwards between the client and the server.

Understanding State Information (continued)

Some reasons why a web application may need to **maintain state** information:

- Temporarily store information for a user as a browser navigates within a multipart form
- Allow a user to create bookmarks for returning to specific locations within a Web site
- Customize individual Web pages based on user preferences
- Provide shopping carts that store order information

Understanding State Information (continued)

- Store user IDs and passwords
- Use counters to keep track of how many times a user has visited a site

The four tools for **maintaining state** information with PHP are:

- ☐ Hidden form fields
- ☐ Query strings
- ☐ Cookies
- ☐ Sessions

SAVING STATING INFORMATION

Using Hidden Form Fields

to Save State Information

- Hidden form fields temporarily store data that needs to be sent to a server that a user does not need to see
- Examples include the result of a calculation
- Create hidden form fields with the `<input />` element
- The syntax for creating hidden form fields is:

```
<input type="hidden" ... />
```

Using Hidden Form Fields to Save State Information (continued)

- Hidden form field attributes have **name** and **value**
- When submitting a form to a PHP script, access the values submitted from the form with the `$_GET[]` and `$_POST[]` autoglobals
- To pass form values from one PHP script to another PHP script, store the values in hidden form fields

Using Hidden Form Fields to Save State Information (continued)

```
<form action="courseListings.php" method="get">
<p>
<input type="submit" value="Register for Classes" />
<input type="hidden" name="diverID"
           value="<?php echo $diverID ?>" />
</p>
</form>
```

Using Query Strings to Save State Information

- A **query string** is a set of name=value pairs appended to a target URL
- A **query string** consists of a single text string containing one or more pieces of information
- Any forms that are submitted with the `GET` method automatically add a question mark (?) and append the **query string** to the URL of the server-side script

Using Query Strings to Save State Information (continued)

- To pass information from one Web page to another using a query string,
  - add a question mark (?) immediately after the URL
  - followed by the query string containing the information in name=value pairs, and
  - separate the name=value pairs within the query string by ampersands (&)

```
<a href="page2.php?firstName=Don&lastName=Gosselin
&occupation=writer">Link Text</a>
```

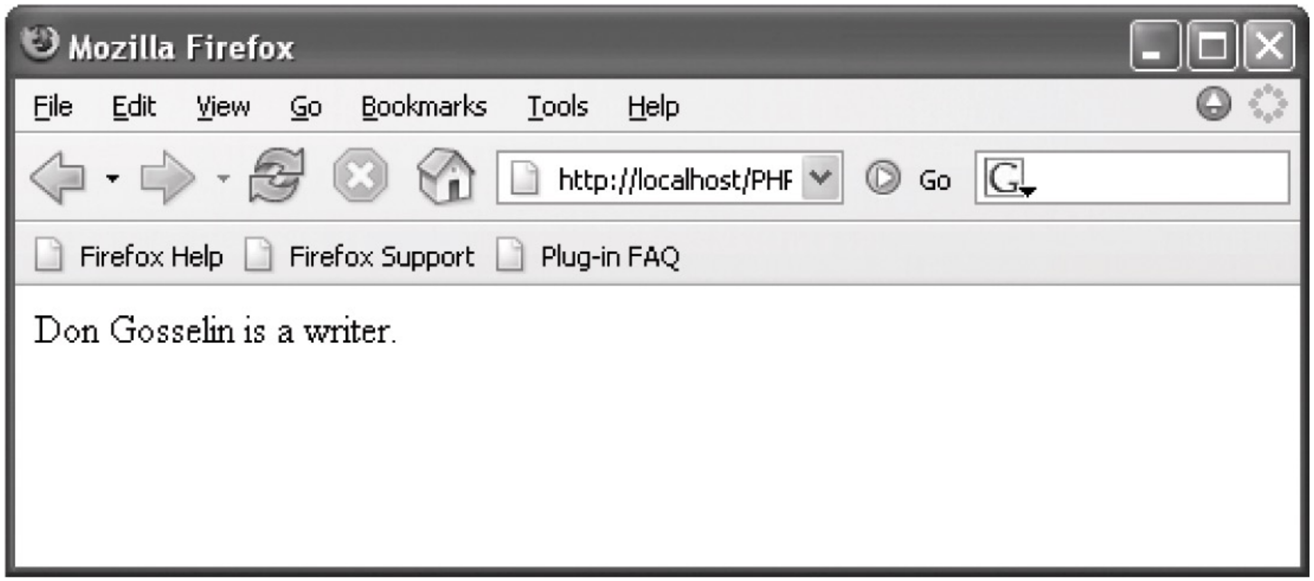
Using Query Strings to Save State Information (continued)

- To pass query string information from one PHP script to another PHP script, echo the values in the script

```
<a href="page2.php?firstName="<?php echo $fname ?>
"&lastName="<?php echo $lname ?>
"&occupation="<?php echo $occ ?>">Link Text</a>
```

Using Query Strings to Save State Information (continued)

```
echo "{$_GET['firstName']} {$_GET['lastName']}
 is a {$_GET['occupation']}. ";
```


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/PHP
Go
Firefox Help Firefox Support Plug-in FAQ

Don Gosselin is a writer.

Output of the contents of a query string

Using Cookies
to Save State Information

- Query strings do not permanently maintain state information
- After a Web page that reads a query string closes, the query string is lost
- **Cookies** are small pieces of information about a user that are stored by a Web server in text files on the user's computer

Using Cookies
to Save State Information (continued)

- **Temporary cookies** remain available only for the current browser session
- **Persistent cookies** remain available beyond the current browser session and are stored in a text file on a client computer
- Each individual server or domain can store only 20 cookies on a user's computer
- Total cookies per browser cannot exceed 300
- The largest cookie size is 4 kilobytes

Using Cookies: Creating Cookies

- The syntax for the `setcookie()` function is:

```
setcookie(name [,value ,expires, path, domain, secure])
```

- You must pass each of the arguments in the order specified in the syntax
- To skip the `value`, `path`, and `domain` arguments, specify an empty string as the argument value
- To skip the `expires` and `secure` arguments, specify 0 as the argument value

Using Cookies:Creating Cookies (continued)

- Call the `setcookie()` function before sending the Web browser any output, including white space, HTML elements, or output from the `echo()` or `print()` statements
- Users can choose whether to accept cookies that a script attempts to write to their system
- A value of true is returned even if a user rejects the cookie

Using Cookies Creating Cookies (continued)

- Cookies cannot include semicolons or other special characters, such as commas or spaces, that are transmitted between Web browsers and Web servers using HTTP
- Cookies ***can*** include special characters when created with PHP since encoding converts special characters in a text string to their corresponding hexadecimal ASCII value

Using Cookies: `name` and `value` Arguments

- Cookies created with only the `name` and `value` arguments of the `setcookie()` function are *temporary cookies* because they are available for only the current browser session

No “expires” argument, for temporary cookies

```
<?php
setcookie("firstName", "Don");
?>
<!DOCTYPE html>
<head>
<title>Skyward Aviation</title>
...
```

Using Cookies: `name` and `value` Arguments (continued)

- The `setcookie()` function can be called multiple times to create additional cookies – as long as the `setcookie()` statements come ***before*** any output on a Web page

```
setcookie("firstName", "Don");
setcookie("lastName", "Gosselin");
setcookie("occupation", "writer");
```

Using Cookies: `expires` Argument

- The `expires` argument determines how long a cookie can remain on a client system before it is deleted
- Cookies created without an `expires` argument are available for only the current browser session
- To specify a cookie's expiration time, use PHP's `time()` function

```
setcookie("firstName", "Don", time()+3600);
```

This "expires" argument, is set to current time + 3600 seconds

Using Cookies: `path` Argument

- The `path` argument determines the availability of a cookie to other Web pages on a server
- Using the `path` argument allows cookies to be shared across a server
- A cookie is available to all Web pages in a specified path as well as all subdirectories in the specified path

```
setcookie("firstName", "Don", time()+3600,
   "/marketing/");

setcookie("firstName", "Don", time()+3600, "/");
```

Using Cookies: `domain` Argument

- The `domain` argument is used for sharing cookies across multiple servers in the same domain
- Cookies *cannot* be shared outside of a domain

```
setcookie("firstName", "Don", time()+3600,
  "/", ".gosselin.com");
```

Using Cookies: `secure` Argument

- The `secure` argument indicates that a cookie can only be transmitted across a secure Internet connection using HTTPS or another security protocol
- To use this argument, assign a value of 1 (for true) or 0 (for false) as the last argument of the `setcookie()` function

```
setcookie("firstName", "Don", time()+3600,
  "/", ".gosselin.com", 1);
```

Using Cookies: Reading Cookies

- Cookies that are available to the current Web page are automatically assigned to the `$_COOKIE` autoglobal
- Access each cookie by using the cookie name as a key in the associative `$_COOKIE[]` array

```
echo $_COOKIE['firstName'];
```

- Newly created cookies are *not available* until after the current Web page is reloaded

Using Cookies: Reading Cookies (continued)

- To ensure that a cookie is set before you attempt to use it, use the `isset()` function

```
setcookie("firstName", "Don");
setcookie("lastName", "Gosselin");
setcookie("occupation", "writer");
if (isset($_COOKIE['firstName'])
     && isset($_COOKIE['lastName'])
     && isset($_COOKIE['occupation']))
     echo "{$_COOKIE['firstName']}
       {$_COOKIE['lastName']}
       is a {$_COOKIE['occupation']}.";
```

Using Cookies: Reading Cookies (continued)

- Can use multidimensional array syntax to set and read cookie values

```
setcookie("professional[0]", "Don");
setcookie("professional[1]", "Gosselin");
setcookie("professional[2]", "writer");
if (isset($_COOKIE['professional']))
     echo "{$_COOKIE['professional'][0]}
          {$_COOKIE['professional'][1]} is a
          {$_COOKIE['professional'][2]}.";
```

Using Cookies: Deleting Cookies

- To delete a persistent cookie before the time assigned to the `expires` argument elapses, assign a new expiration value that is sometime in the past
- Do this by subtracting any number of seconds from the `time()` function

```
setcookie("firstName", "", time()-3600);
setcookie("lastName", "", time()-3600);
setcookie("occupation", "", time()-3600);
```

Using Sessions to Save State Information

- **Spyware** can gather user information from a client computer, such as cookies, for marketing and advertising purposes without the user's knowledge
- A **session** refers to a period of activity when a PHP script stores *state information on a Web server*
- **Sessions** allow you to maintain state information even when clients disable cookies in their Web browsers

Starting a Session

- The `session_start()` function starts a new session or continues an existing one
- The `session_start()` function generates a unique session ID to identify the session
- A **session ID** is a random alphanumeric string that looks something like:

  ```
  7f39d7dd020773f115d753c71290e11f
  ```

- The `session_start()` function creates a text file on the Web server that is the same name as the session ID, preceded by `sess_`

Starting a Session (continued)

- Session ID text files are stored in the Web server directory specified by the `session.save_path` directive in your php.ini configuration file
- The `session_start()` function does not accept any functions, nor does it return a value that you can use in your script

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

Starting a Session (continued)

- You must call the `session_start()` function ***before*** you send the Web browser any output
- If a client's Web browser is configured to accept cookies, the session ID is assigned to a temporary cookie named `PHPSESSID`
- Pass the session ID as a query string or hidden form field to any Web pages that are called as part of the current session

Starting a Session (continued)

```
<?php
session_start();
...
?>
<p><a href='<?php echo "occupation.php?PHPSESSID="
     . session_id() ?>'>Occupation</a></p>
```

Working with Session Variables

- Session state information is stored in the `$_SESSION` autoglobal
- When the `session_start()` function is called, PHP either initializes a new `$_SESSION` autoglobal or retrieves any variables for the current session (based on the session ID) into the `$_SESSION` autoglobal

Working with Session Variables (continued)

```
<?php
session_set_cookie_params(3600);
session_start();
$_SESSION['firstName'] = "Don";
$_SESSION['lastName'] = "Gosselin";
$_SESSION['occupation'] = "writer";
?>
<p><a href='<?php echo "Occupation.php?"
 . session_id() ?>'>Occupation</a></p>
```

Sets the "lifetime" argument to 3600 seconds

Working with Session Variables (continued)

- Use the `isset()` function to ensure that a session variable is set before you attempt to use it

```
<?php
session_start();
if (isset($_SESSION['firstName']) &&
   isset($_SESSION['lastName'])
      && isset($_SESSION['occupation']))
    echo "<p>" . $_SESSION['firstName'] . " "
           . $_SESSION['lastName'] . " is a "
           . $_SESSION['occupation'] . "</p>";
?>
```

Deleting a Session

- To delete a session manually, perform the following steps:

1. Execute the `session_start()` function
2. Use the `array()` construct to reinitialize the `$_SESSION` autoglobal
3. Use the `session_destroy()` function to delete the session

Deleting a Session (continued)

```
<?php
session_start();
$_SESSION = array();//unset all session variables
session_destroy();
?>
```

4. Modify a “Registration” / “Log In” page so it deletes any existing user sessions whenever a user opens it.

UNDERSTANDING PHP SECURITY ISSUES

Understanding PHP Security Issues

- Viruses, worms, data theft by hackers, and other types of security threats to Web-based applications.
- Web server security issues
  - Firewalls
  - **Secure Sockets Layer** protocol to encrypt data
- Secure coding issues
  - Refers to the writing of code in such a way that it minimizes any intentional or accidental security issues.
  - No magic formula for writing secure code, although there are various **secure coding techniques** to minimize security threats in programs.

Some Secure Coding Techniques (continued)

- Disable the register_globals directive in php.ini
  - **On** – client, server and environment information are automatically available as global variables.
    - For example, $email instead of $_GET["email"].
    - Security issue that an unscrupulous hacker can take advantage of
  - **Off** (recommended; turned off after PHP4.2.0)
    - Use autoglobal arrays such as $_GET and $POST

Some Secure Coding Techniques (continued)

- Validate submitted form data
  - Unscrupulous hackers can falsify submissions by bypassing JavaScript validation code or by constructing HTTP headers.
  - Validate data in php scripts
    - isset() function
    - empty() function
    - is_numeric() function
- Use sessions to validate user identities
  - Randomly generated alphanumeric session id is extremely difficult to guess.

Some Secure Coding Techniques (continued)

- Store code in external files
  - Helps to secure your scripts by hiding the code from hackers and other programmers who might steal and claim your scripts as their own.
- Access databases through a proxy user
  - Create a single account that a PHP script uses to access the database for a user by proxy rather than for each visitor.

```
$DBConnect = @new mysqli ("localhost",
"proxy_user", "password");
   if (mysqli_connect_errno()) ...
```

Some Secure Coding Techniques (continued)

- Handle magic quotes
  - ☐ Handle single and double quotes before writing to a data source, such as a file or database.
  - ☐ Magic quotes in PHP automatically adds a backslash (\) to any single quote, double quote or NULL character.
  - ☐ 'Magic Quotes' is ***deprecated*** in PHP 5.3 and ***removed*** in PHP 5.4
  - ☐ If magic quotes are disabled (magic_quotes_gpc directive in php.ini is disabled rather than enabled by default), use `addslashes()` function.
  - ☐ `stripslashes()` function removes the slashes.

Some Secure Coding Techniques (continued)

- Report errors
  - **display_errors** directive in php.ini
    - On (by default) – print error messages to a web browser
    - Off – do not print error messages to a web browser
    - Off is recommended when running in production environments
  - **display_startup_errors** directive in php.ini
    - On – display errors that occur when PHP first starts.
    - Off (by default) – do not display errors that occur when PHP first starts.
    - Off is recommended. On can be assigned only when debugging a script.

Summary

- Information about individual visits to a Web site is called state information
- Maintaining state means to store persistent information about Web site visits with hidden form fields, query strings, cookies, and sessions
- The four tools for maintaining state information with PHP are: hidden form fields, query strings, cookies, and sessions
- A query string is a set of name=value pairs appended to a target URL

Summary (continued)

- Cookies, or magic cookies, are small pieces of information about a user that are stored by a Web server in text files on the user's computer
- Cookies cannot include semicolons or other special characters, such as commas or spaces, that are transmitted between Web browsers and Web servers using HTTP but can using PHP
- The path argument determines the availability of a cookie to other Web pages on a server

Summary (continued)

- The **domain** argument is used for sharing cookies across multiple servers in the same domain
- The **secure** argument indicates that a cookie can only be transmitted across a secure Internet connection using HTTPS or another security protocol
- A **session** refers to a period of activity when a PHP script stores state information on a Web server

Summary (continued)

- Viruses, worms, data theft by hackers, and other types of security threats to Web-based applications.
  - Web server security issues
  - Secure coding issues
- Secure coding techniques
  - Disable the register_globals directive in php.ini
  - Validate submitted form data
  - Use sessions to validate user identities
  - Store code in external files
  - Access databases through a proxy user
  - Handle magic quotes
  - Report errors