

Session Management in PHP

INFO 344 Winter 2007

Administrivia

- Lab 8 (MySQL & PHP) Due END OF DAY on Tuesday
- Lots of time today and Tuesday to work on lab
- Meet today and Tuesday about your project
- Privacy and Security Next Week
 - NO LABS! HOORAY!!

Today

- Session management with PHP
- Privacy issues

State management

- The problem: HTTP is a stateless protocol
 - Web server gets URL only
 - No idea what happened before
- How can we maintain application state between pages?
 - Between application pages
 - Between browser sessions

HTTP is stateless



Levels of state management

- Long term application state
 - Effectively last forever
 - User accounts, records
 - Use *database*
- User session
 - Between web pages
 - Single activity: shopping, filling out a form
 - User may close and restart browser
 - Can be stored on server or in browser

Technologies for managing state

- URL string parameters
- Cookies
- PHP session management

Passing session through URLs

- Scenario: User logs in on login page. How do we maintain the user's identity on other pages?
- Solution: Pass username as URL string or form parameter

```
<a href="link.html?userID=<?php echo  
$_REQUEST['userID']; ?>">Link text</a>
```

- Note that this needs to be added to all links

Using forms

- Values can be passed as a hidden input parameter

```
<form action="link.php" method="post">  
  <input type="hidden" name="userID"  
  value="<?php echo $_REQUEST['userID'];  
?>"/>  
  ...  
  <input type="submit" value="Next page"/>  
</form>
```

What value to use?

- What should we persist to be sure that a user is logged in?
 - Username: but can be tampered with, copied
 - Password: shouldn't be passed around
- Any value passed in the URL string can be tampered with by the user
- Login can create a token that is difficult to fake
 - Random ID generated by server
 - Store in database and verify each page
 - Timestamp
 - Disallow sessions after a set period of time

Using URL parameters

- Advantages
 - Works everywhere
 - Technically simple
 - Can be accessed client-side [JS] and server-side [PHP]
- Disadvantages
 - Can be viewed or altered by user
 - Must rewrite all links or use forms
 - Lost if user closes browser window

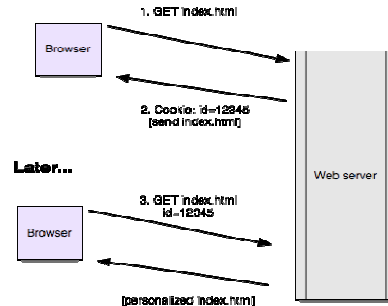
Cookies

- A cookie is a piece of text that is stored by the web browser between sessions
 - Set in the browser by a specific web server
 - Browser passes information back to that server only
- Commonly used to store user information
 - Or to tag individual browsers and identify them later
- Consists of four components
 - Source site
 - Attribute name
 - Attribute value
 - Expiration date (by default, when browser closes)

What cookies can do

- Store session data
 - Login info, pages visited, shopping cart
- Track users who visit your website
 - Assign them an ID, read it later
 - Even without logging in
- Allow users to personalize their experience
 - Set site settings, store in browser

How cookies work



The messy details

- Server at domain.com sends cookie in HTTP response

```
HTTP/1.0 200
Content-Length: 1276
Content-Type: text/html
Date: Tue, 06 Nov 2001 04:12:49 GMT
Expires: Tue, 06 Nov 2001 04:12:59 GMT
Set-Cookie: id=12345
<html>...</html>
```
- Browser stores cookie on local system
- When browser revisits domain, passes cookie back

```
GET /index.php HTTP/1.0
Connection: Keep-Alive
Cookie: id=12345
Host: www.test.com
Referer: http://www.test.com/
```

Using cookies with PHP

- Set cookies with `setcookie(name, value, [expiration, path, domain, secureOnly])`
- Retrieve cookies with `$_COOKIES["cookieName"]`

```
// if cookie is not set
if (!isset($_COOKIES["color"])) {
    // set cookie to last until (current time plus) 3 days
    setcookie("color", "green", time() + 3*60*60*24);
}
```
- See <http://us2.php.net/setcookie>

Using cookies in JavaScript

- Existing cookies can be viewed in most browsers
- Cookies can also be written and read in JavaScript
- See <http://www.quirksmode.org/js/cookies.html>

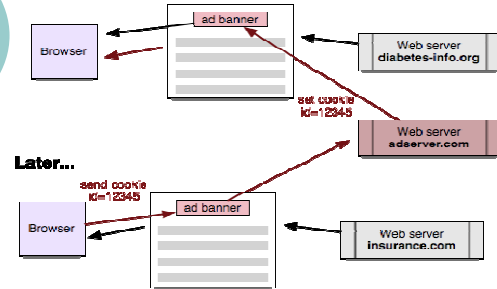
Using cookies

- Advantages
 - Easy
 - Can be used client- and server- side
 - A knowledgeable user has control over what is stored
 - Persistent across browser sessions
- Disadvantages
 - Can only store text
 - Cookies may only be up to a certain size [~4k]
 - May only be read by the site that set them
 - Some users turn cookies off in their browsers
 - Users may not realize cookies are being set

Third-party cookies

- In general, cookies may only be read by the website that sets them
- However, sometimes one web server provides content for multiple sites
 - Ad banners from advertising agencies
- These third-party (or "tracker") cookies can trace user through several sites

Third-party cookie example



adserver has seen the user at diabetes-info.org **and** insurance.com!

Session management with PHP

- PHP provides an abstraction layer for managing browser sessions
- Can attach PHP variables (including simple objects) to a user's session
 - Can retrieve variables on any page
 - Variables last until browser closed
- Underneath the hood
 - PHP creates a random ID
 - Sends ID to user as cookie
 - PHP stores ID and variables in server-side text file

Creating a PHP session

- Use `session_start()` to begin session
 - If stored values exist, retrieve them
- Set and get values in `$_SESSION` array

```
// load session
session_start();
// check to see if variable "color" is set
// if not, set it
if (!isset($_SESSION["color"])) {
    $_SESSION["color"] = "vermillion";
}
```

Ending a PHP session

- If we want to remove session data

```
// clear the SESSION array by setting it
to a new array
$_SESSION = array();
// call session_destroy() to delete files
session_destroy();
```