PHP Cookies, Sessions, and Session Variables

Fall 2009

CSCI 2910 Server-Side Web Programming

Objectives

- Understand and use Cookies in PHP scripts.
- Understand and use Sessions and Session variables in PHP scripts.

HTTP

HTTP is a stateless protocol

- Each page "stands alone" and has no memory of past actions.

Addressed in Netscape 3.0 with cookies.

Cookies allow us to write data to user's computer and read that data as user traverses site.

Cookies can only be *written* as part of header information, therefore **cannot create or add to a cookie after writing to browser**.

Using cookies

To create a cookie, use **`setcookie()`**

```
setcookie(cookiename, value, [expire]);
setcookie("cook","27");
```

Expiration—expressed using time. If not set, cookie is valid for this user session only.

```
setcookie("other","1", time()+60*60*24*30);
```

http://einstein.etsu.edu/~pittares/CSCI2910/examples/8-1.php

Retrieved similar to $_POST variables:

- $_COOKIE['cookiename']

http://einstein.etsu.edu/~pittares/CSCI2910/examples/8-2.php

Deleting and checking cookies

To delete: overwrite cookie with expiration time in the past.

```
setcookie("cook","",time()-100);
```

http://einstein.etsu.edu/~pittares/CSCI2910/examples/8-3.php

Actual cookie deletion done by user's browser.

To see if the user accepts cookies, write one and then check (*on another page or after a refresh*) to see if it exists.

Cookie tutorial:

http://einstein.etsu.edu/~pittares/CSCI2910/examples/8-4.php

Conclusion: Using Cookies

If user accepts cookies, *and* if you remember to **manage setting them prior to non-header output**, then they're fine.

If you use Sessions:

- PHP manages complexity.
- If the user doesn't support cookies, PHP has an automated "workaround".
- More complex data storage (arrays, etc.) easier to implement.
- But, you lose multi-visit persistence

What is session control?

Gives ability to track a user through site, and easily move data related to that user among pages.

- No need to move data through hidden form fields.
- Very useful for authentication, but can be used any time persistent data needed throughout a site visit.

How sessions work

Sessions are identified by a random number (Session ID) generated by PHP and stored on the client computer in 1 of 2 ways:

- Using a cookie, if the user's browser supports.
- Appending the session number to URLs as user traverses site
  - www.whatever.com?PHPSESSID=495294532459x

Session ID corresponds a session data store on server

A session will eventually expire--usually after a specified period of inactivity.

Progression of events

PHP script starts a session. *Done before any other page activity*.

```
session_start();
```

Session ID created and stored on **user's** computer. (if possible)

Session variables are created, and values stored on the **server**.

PHP script can use these variables from page to page throughout a site.

Using session variables

Some PHP servers automatically start a Session for every user when they visit the site.

May slow things down due to unnecessary overhead.

Controlled by **PHP.ini** file on the server.

http://einstein.etsu.edu/~pittares/PHPTest/phpinformation.php

Session operations changed in PHP 4.1, so be careful with older installations and reference books.

Starting a session

In *any script* using sessions, you must first call **`session_start()`**.

- If session has not been established, this will do that.
- If a session has been established, this will load session data.

You **must** start the session at the very beginning of the script--as part of header transmission.

Add or access session variables by using the **`$_SESSION`** superglobal array.

Session Handling

```
<?php
  session_start();
  $_SESSION['name'] = "Dr. Tony Pittarese";
  $_SESSION['office'] = "Nicks 484";
  $_SESSION['phone'] = 96951;
?>
```

http://einstein.etsu.edu/~pittares/CSCI2910/examples/8-5.php

```
<?php
  session_start();

  echo "Here's the session info:<br />";
  foreach ($_SESSION as $var=>$contents)
    echo "$var: $contents<br />";
?>
```

Manipulating Session ID

`session_id()` allows you to get or set the Session ID.

- If no parameter, returns the Session ID.
- If given a parameter, sets that as the Session ID.

http://einstein.etsu.edu/~pittares/CSCI2910/examples/8-7.php
http://einstein.etsu.edu/~pittares/CSCI2910/examples/8-8.php

Manipulating the Session data

`session_unset()` erases all session variables and data.

http://einstein.etsu.edu/~pittares/CSCI2910/examples/8-9.php

`unset()` can be used to erase a single variable and data.

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

`session_destroy()` destroys the session data (without destroying the session variables).

- Can be useful for "logging out" user.

http://einstein.etsu.edu/~pittares/CSCI2910/examples/8-10.php
http://einstein.etsu.edu/~pittares/CSCI2910/examples/8-11.php

Session variable arrays

Session variables can be arrays

```
<?php
  session_start();
  $_SESSION['list'][]="Hello";
  $_SESSION['list'][]="Wow";
  echo count($_SESSION['list'])."<br />";
  foreach ($_SESSION['list'] as $item)
    echo "$item<br />";
?>
```

Can be useful technique for shopping carts or other data that is accumulated over multiple page visits.

http://einstein.etsu.edu/~pittares/CSCI2910/examples/8-12.php

When and why to use Sessions

Performance

- When performing a slow operation, storing the results for use on several pages is better than repeating the calculation on each.
  - Example: storing results of SQL query

Sequence

- When a user process takes place over a sequence of screens, storing information saves time and user input.

Personalization

- Session variables can be used to store user color or layout preferences or facts about browsing activity. Pages can then adapt to that activity.

http://einstein.etsu.edu/~pittares/CSCI2910/examples/8-13.php

Potential problems with Sessions

Multiple Servers

- Since session information stored on server, harder to configure when multiple servers fulfill user requests.
  - Handled typically by using a DB to store session data.

Performance

- Additional workload for server to store and retrieve information.

Garbage Collection

- Since user may abandon site visit, must determine session timeout values and employ garbage collection.

Potential problems with Sessions

Bookmarking

- Unlike GET parameters which can be bookmarked, data moved from page to page is lost when the user bookmarks a page and returns later.

Security

- If a user can counterfeit a SESSION cookie, they could "hijack" another user's interaction session.

Session ID Numbers

If the user allows Cookies, this will be handled automatically.

If the user does not allow Cookies, then as you move from page to page you (the programmer) must manually keep up with the Session ID.

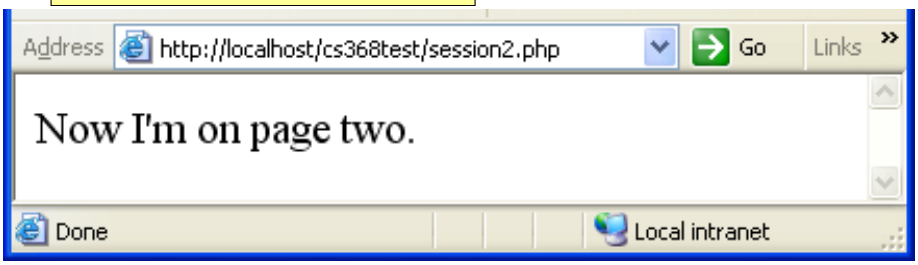
Append the SID to the URL.

```
<a href="session2.php?PHPSESSID=<?=SID?>">test</a>
```

Or turn on transparent SID support in the PHP configuration

If Transparent SID is on

If the user accepts cookies:



If the user does not accept cookies: