

CSE 135 Server Side Web Languages Lecture # 6

State and Session Management

UCSD
Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Addressing HTTP Statelessness

- HTTP being stateless, in other words having no memory from page view to page view, can make Web programming a hassle. A variety of techniques to address this are used including:
 - Hidden Form fields
 - Dirty URLs
 - Cookies
 - Memory or session cookies
 - Persistent cookies
 - Wrongly - client tech
- Fortunately, most modern programming environments such as PHP, ColdFusion, ASP.NET, JSP, etc. provide a session concept which will abstract away many of the details of state preservation.
 - Sessions are most often implemented as cookies which store a unique ID referencing data stored on the server

UCSD
Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Hidden Form Fields

- Forms contain `<input type="hidden" name="cartid" value="78Ccad786">` this data is passed along with submission
- Subsequent page loads are rewritten with hidden fields in them

UCSD
Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Hidden Form Field Notes

- Open for easy parameter tampering
 - Countermeasure: integrity checks
- Does not preserve state information across a user visit
- Requires that a form be used to trigger the pass back of the hidden data
 - .NET uses hidden form fields (a viewstate value) and wrap the whole page in a single giant form

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

State Preservation via "Dirty URLs"

- All links are rewritten with query strings in them
`Page 1`
`Page 2`
- All forms pass the value via the query string as well (either direct action change or via hidden field)

The diagram illustrates the process of state preservation using dirty URLs. At the top, a server icon labeled 'www.xyz.com' is connected to a database icon. A text box states: 'Server references stored data for session as identified by passed value'. Below the server, two boxes represent 'Page 1 loaded and links annotated' and 'Page 2 returned URLs with session info (or modified values if state stored in string itself)'. A central laptop icon represents the user's browser. Arrows show the flow of data: from the server to Page 1, from Page 1 to the browser, from the browser to Page 2, and from Page 2 back to the server.

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Dirty URL state management

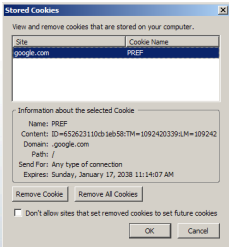
- Open for easy tampering
 - Countermeasure: integrity checks
- Does not preserve state information across a user visit
- Does not promote URL usability, marketing, etc.
- Without dereferencing to server stored data has significant limit to the amount of data that can be saved (URL limit around 2K)

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Cookies

- Cookies are simple short pieces of data stored on a user's system often in a file (cookies.txt) or a directory C:\Documents and Settings\userid\Cookies with individual cookie files
 - Example contents of a Google cookie
 - PREFID=6b476a56502ce137;TM=1088552548;LM=1088552548;S=FUE7V3k3CtQY]BeHgGoogle.com/15362618878336321163440547774429646386*
 - Notice that often cookies do not store the actual data to be tracked but an id that references data stored on a server



UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Cookies

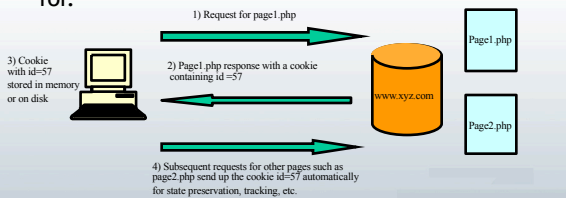
- A cookie is typically issued to a browser via the HTTP header
 - Set-Cookie: NAME=VALUE; expires=DATE; path=PATH; domain=DOMAIN_NAME; secure
- Most of the time we would not set cookies manually in an HTTP response (though you could), instead the Web programming environment you employ will provide some command.
 - To issue a cookie in PHP use `setcookie(name, value);`
 - Example: `setcookie('leadstooge', 'moe');`
- However, not that JavaScript can also easily set cookies client-side via the `document.cookie` object

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

First Party Cookie Example

- Once the cookie is set it is transferred back and forth from the browser to the site for every request within the domain and path that it is set for.



UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Third Party Cookie Example

- Cookies can be set by any HTTP request including images, scripts, CSS, etc.
 - This is often used in the case of Third-party cookie issued from a banner ad or tracking site

1) Request for page1.php
 2) Page1.php response with a first-party cookie containing id=57
 3) Cookie with id=57 stored in memory or on disk
 4) Response is parsed and an image request to www.banner.com/ad.gif is made
 5) Image response sets cookie for track=A7b
 6) Future request to abc.com that also uses banner.com allows tracking cookie to show cross-site usage

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Privacy Concern with Cookies

- Third party cookies represent a possibility to track user activity across a range of sites
 - Typically these sites are part of some "network" - like an ad network
- The cookies are also set often via a clear pixel GIF so the user may not see what they should be trying to block
 - These cookie setting clear pixels are dubbed "Web bugs" or just "bugs"
 - Typically they are used in a hosted Web analytics services like Hitbox from WebSideStory
- The privacy concern with cookies come from the fact that you may register at a particular site like the last example xyz.com and they then associate your personal information with the cookie-id. If the personal information is then shared to other network participating sites they can start building a detailed profile on your usage habits
 - Because of this many users want to block third party cookies, but the real problem is giving out your private information and knowing what happens to it afterwards
 - We lack "informational self-determination" online which is unfortunate
 - Browsers, tools, and technology like P3P help but fundamentally once again we see a social issue and not a technical one.

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

IE Cookie Related Settings

Internet Options - Privacy tab: Move the slider to select a privacy setting for the Internet zone. Medium. Settings: - Blocks third-party cookies that do not have a compact privacy policy. - Blocks third-party cookies that use personally identifiable information without your explicit consent. - Restricts first-party cookies that use personally identifiable information without explicit consent.

Advanced Privacy Settings: You can choose how cookies are handled in the Internet zone. Cookies: Override automatic cookie handling. First Party Cookies: Accept, Block, Prompt. Third Party Cookies: Accept, Block, Prompt. Always allow session cookies.

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Firefox Cookie Related Settings

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Cookie Example - cookie1.php

```

<?php
    setcookie('name','Noe');
    setcookie('numstooges','3');
?>
<html>
<head><title>Simple Cookie</title></head>
<body>
<h1>I just "baked" a cookie</h1>

<a href="cookie2.php">See what's in the cookie</a>

<form>
    <input type="button"
        value="JS Cookie Check"
        onclick="alert(document.cookie);">
</form>
</body>
</html>
    
```

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Cookie Example - cookie2.php

```

<html>
<head><title>Show Cookie</title></head>
<body>
<?php
    print ("There are " . $_COOKIE['numstooges'] . " stooges and the
        main one is named " . $_COOKIE['name']);
?>
</body>
</html>
    
```

- The cookie set in this particular case is a *session cookie* or *memory cookie*
 - It will be destroyed once the browser session is closed down
 - It is not shared between browsers, but it is likely shared between browser windows
 - These types of cookies are not as troubling from a privacy point of view and are necessary to make a typical modern Web site (unfortunately some users still deny them)

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Cookie Limits

- The browser and server may put some limits on cookies
 - Commonly held that browser:
 - Can store a limited number of cookies (300?)
 - Cookies received after browser limit are deleted starting the least used
 - A single cookie can be no more than 4K in size
 - Cookies are trimmed down to 4K if they exceed that size
 - Commonly held that a browser store -or- server will accept no more than 20 cookies for a particular host.domain combo
 - Though web.xyz.com and store.xyz.com are different and thus could have 20 each
 - The reality is these limits may not in fact be true as they are implementation dependent
- Commonly held development practice suggests using as few cookies per domain and only storing a sessionid or related value with the bulk of tracked information stored server side.

Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Setting Persistent Cookies

- Specify some expiration date
 - In PHP `setcookie('name', 'value', expiration, 'path', 'domain', secure);`
 - Expiration stored in number of seconds since Epoch
 - Examples


```
setcookie('color', 'green', time()+3600);
// expires in 1 hour

setcookie('userid', 'admin', time()+3600, '/admin');
// limit to admin directory path

setcookie('userid', 'admin', time()+3600, '/admin',
        '/', 1);
// use security in transmit
```

Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Setting Persistent Cookies

```
<?php
setcookie('favoritestooge', 'Moe', time()+3600);

?>
<html>
<head><title>Simple Cookie</title></head>
<body>
<h1>I just "baked" a persistent cookie so check your disk</h1>
</body>
```

```
C:\Documents and Settings\Thomas Powell\Application Data\Mozilla\Firefox\Profile
> netstat -s | findstr cookies.txt
HTTP Cookie File
C:\inetpub\wwwroot\php\sessions\php\sessions\php\cookie_spec.html
This is a generated file, do not edit.
To delete cookies, use the Cookie Manager.
192.168.0.3 FALSE /php/sessions/ FALSE 1092815507 favoritestooge
From
google.com TRUE / FALSE 2147368447 PREP ID=652623110cbie
65871f-10924203371f-1092420337-0-v011mwh1m-v0check
```

Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Deleting Cookies

- Session cookies go away on browser close and persistent cookies are blasted upon expiration you can do it early by setting a value to blank or modifying the expiration or even better both!

```
setcookie('favoritestooqe','');
setcookie('favoritestooqe','',time()-3600);
```

- You do want to encourage a user to logout of a Web application rather a click away because you want to improve security
 - Consider the problem of a public use system with still active cookies from banking sites, secure extranets, etc.

UCSD Jacobs School
Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Sessions

- While it is possible to do all the management of most sites using cookies (or even hidden fields or dirty URLs) it is much better to use sessions
- A well implemented session management system abstracts away how data is preserved from page to page (typically in a cookie) and stores the saved information server side only referencing via a SESSIONID value in the cookie
 - You could of course build this easily enough by storing some database-id in a cookie yourself and every page load doing a query against the database reading or modifying data in regards to the user's state
- A significant benefit of sessions not storing the data in the cookie is that it helps guard against session hijacking
 - Assuming that session ids are not easily guessable
 - Even better to "brand" the session to the user somehow

UCSD Jacobs School
Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Sessions in PHP

- Use `session_start()` and PHP will issue a cookie with name PHPSESSID and some value
- You will use `session_start()` most likely on every page in your app
- It is possible to set a `php.ini` setting of `session.auto_start` so that you don't need to use `session_start()` all the time
- You can specify a different value other than PHPSESSID before by calling `session_name('MYID')`; where MYID is your value and then calling `session_start()`;
- To create a session variable use `$_SESSION['var'] = 'value';`

Confirms setting cookie

The site 192.168.0.3 wants to set a second cookie.

Use my choice for all cookies from this site

Name: PHPSESSID

Cookie: 1397234c4a0592798bf028b332c4129

Host: 192.168.0.3

Path: /

Send For: Any type of connection

Expires: at end of session

UCSD Jacobs School
Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Session Example in PHP

```

<?php
    session_start();
    $_SESSION['fav_stooge']='Moe';
    $_SESSION['num_stooges']=3;
?>
<html>
<head>
<title>Session Fun</title>
</head>
<body>
<h1>Just set some session values</h1>

<a href="session2.html">Next page</a>
  <!-- eventually read this later on session3.php -->

</body>
</html>

```

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Session Example in PHP Contd.

```

<html>
<head>
<title>Session Fun</title>
</head>
<body>
<h1>Reading session variables</h1>
<?php
    session_start();
    print "There are ".$_SESSION['num_stooges'] ." and
    my favorite is ".$_SESSION['fav_stooge'];
?>
</body>
</html>

```

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Deleting Session Info from PHP

- To delete an individual session variable in PHP use `unset($_SESSION['thevar']);` just as you would unset an arbitrary value in PHP
- To delete all session variables then just blow out the entire array like so `$_SESSION = array();`
- Removing all session data from the server can be accomplished with `session_destroy();`
- *Note: That to do anything with session values including destroying them you still have to call `session_start()` first.*
- *Just like cookies you really ought to clean up sessions upon "logout"*

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

PHP Session Details

- By default sessions uses memory cookies that expire upon browser close. You can use the `session_set_cookie_params(expiration, 'path', 'domain', ,secure)` function to set the typical cookie values you might want to do
 - Example:


```
session_name('mysid');
session_set_cookie_params(3600, '/whoismgr', 'www.xyz.com');
session_start();
```
- Note that you do not have to manually calculate the time from EPOCH here you specify the number of seconds into the future before the cookie expires.

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Session Considerations

- Session Hijack
 - Make the IDs unpredictable and tie to a browser
 - HttpOnly Cookies
- Cookies off
 - Default to a dirty that puts the SessionID into every link dynamically as a query string
 - Careful: not automatic and assume all pages get passed via script engine unlike a cookie
- Sessions in a Server Farm
 - How do you share the server stored data?
 - Don't
 - Sticky Server
 - Share it using
 - A backend db
 - Letting the load balancer hold it

UCSD Jacobs School Department of Computer Science and Engineering

CSE 135 Server Side Web Languages Lecture # 6

Some other important ideas

- Reassociating cookies
 - Why your blank Vons club card doesn't save you as much as you think
- Using Flash persistence, browser specific or HTML5 persistence - we can abstract this - careful here
- Tricks using if-modified-since values for Web bugs
- Tracking with JavaScript, fingerprinting and simple or esoteric ideas like frames or window.name
- If you are going to worry - be consistent

UCSD Jacobs School Department of Computer Science and Engineering
