Cookies and Sessions
Maintaining State in HTTP
The web is “stateless” - the browser does not maintain a connection to the server while you are looking at a page. You may never come back to the same server - or it may be a long time - or it may be one second later.

So we need a way for servers to know “which browser is this?”

In the browser state is stored in “Cookies”

In the server state is stored in “Sessions”
Some Web sites always seem to want to know who you are!
Other Web sites always seem to know who you are!
You watch the YouTube video for an 30 seconds
How YouTube sees you...

Browser

Click  Draw

Server

GET  Whole Page

GET  Whole Page

Click  Draw
Multi-User

- When a server is interacting with many different browsers at the same time, the server needs to know *which* browser a particular request came from.

- Request / Response initially was stateless - all browsers looked identical - this was really really bad and did not last very long at all.
Web Cookies to the Rescue

Technically, cookies are arbitrary pieces of data chosen by the Web server and sent to the browser. The browser returns them unchanged to the server, introducing a state (memory of previous events) into otherwise stateless HTTP transactions. Without cookies, each retrieval of a Web page or component of a Web page is an isolated event, mostly unrelated to all other views of the pages of the same site.

http://en.wikipedia.org/wiki/HTTP_cookie
1. browser requests a Web page

2. server sends page+cookie

3. browser requests another page

http://en.wikipedia.org/wiki/HTTP_cookie
Cookies In the Browser

- Cookies are marked as to the web addresses they come from - the browser only sends back cookies that were originally set by the same web server.

- Cookies have an expiration date - some last for years - others are short-term and go away as soon as the browser is closed.
Playing with Cookies

- Firefox Developer Plugin has a set of cookie features
- Other browsers have a way to view or change cookies
Welcome to SI539

Welcome to the SI539 Sales Site

Here are two quotes from Chndi that look like they might reach the same conclusion.

*Power* is of two kinds. One is obtained over another by acts of love. *Power* based on effective and permanent then the one obtained.

I hope to demonstrate that the real source of authority is not a few, but by the acquisition of authority, when abused.

---

http://localhost:3000/One

<table>
<thead>
<tr>
<th>1 cookie</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NAME</strong></td>
</tr>
<tr>
<td><strong>VALUE</strong></td>
</tr>
<tr>
<td><strong>HOST</strong></td>
</tr>
<tr>
<td><strong>PATH</strong></td>
</tr>
<tr>
<td><strong>SECURE</strong></td>
</tr>
<tr>
<td><strong>EXPIRES</strong></td>
</tr>
</tbody>
</table>
Cookies

- Identifying Individual Users
- The Web is “stateless”
- How do we make the web seem not to be stateless
Request Response Again!
HTTP Request / Response Cycle

Web Server

HTTP Request

HTTP Response

Browser

Hello there my name is Chuck
Go ahead and click on here.

Internet Explorer, FireFox, Safari, etc.

http://www.oreilly.com/openbook/cgi/ch04_02.html
HTTP Request / Response Cycle

We do an initial GET to a server. The server checks to see if we have a cookie with a particular name set. Since this is our first interaction, we have not cookies set for this host.
Along with the rest of the response, the server sets a cookie with some name (sessid) and sends it back along with the rest of the response.
HTTP Request / Response Cycle

GET /index.html HTTP/1.1
Accept: www/source
Accept: text/html
Cookie: sessid=123
User-Agent: Lynx/2.4

From that point forward, each time we send a GET or POST to the server, we include any cookies which were set by that host.

http://www.oreilly.com/openbook/cgi/ch04_02.html
On each response, the server can change a cookie value or add another cookie.

HTTP Request / Response Cycle

Web Server

HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: name=chuck

<head> .. </head>
<body>
<h1>Welcome ....</h1>

Browser

host: sessid=123
host:name=chuck

HTTP Response

http://www.oreilly.com/openbook/cgi/ch04_02.html
HTTP Request / Response Cycle

From that point forward, each time we send a GET or POST to the server, we include all the cookies which were set by that host.

**HTTP Request**

- GET /index.html HTTP/1.1
- Accept: www/source
- Accept: text/html
- Cookie: sessid=123,name=Chuck
- User-Agent: Lynx/2.4

**Browser**

- host: sessid=123
- host: name=chuck

**Web Server**

http://www.oreilly.com/openbook/cgi/ch04_02.html
Security

• We only send cookies back to the host that originally set the cookie

• The browser has *lots* of cookies for lots of hosts

• To see all Cookies: Firefox -> Preferences -> Privacy -> Show Cookies
The Firefox Web Developer Plugin Shows Cookies for the Current Host.

<table>
<thead>
<tr>
<th>NAME</th>
<th>___umb</th>
</tr>
</thead>
<tbody>
<tr>
<td>VALUE</td>
<td>140904167</td>
</tr>
<tr>
<td>HOST</td>
<td>dr-chuck.com</td>
</tr>
<tr>
<td>PATH</td>
<td>/</td>
</tr>
<tr>
<td>SECURE</td>
<td>No</td>
</tr>
<tr>
<td>EXPIRES</td>
<td>Wed, 24 Sep 2008 10:28:02 GMT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAME</th>
<th>___umc</th>
</tr>
</thead>
<tbody>
<tr>
<td>VALUE</td>
<td>140904167</td>
</tr>
<tr>
<td>HOST</td>
<td>dr-chuck.com</td>
</tr>
<tr>
<td>PATH</td>
<td>/</td>
</tr>
<tr>
<td>SECURE</td>
<td>No</td>
</tr>
<tr>
<td>EXPIRES</td>
<td>At End Of Session</td>
</tr>
</tbody>
</table>
Two Kinds of Cookies

• Two kinds of cookie

• Long-lived - who you are - account name last access time - you can close and reopen your browser and it is still there

• Temporary - used to identify your session - it goes away when you close the browser
| NAME   | __lumb             |
| VALUE  | 140904167          |
| HOST   | .dr-chuck.com      |
| PATH   | /                 |
| SECURE | No                |
| EXPIRES| Wed, 24 Sep 2008 10:28:02 GMT |

| NAME   | __lmc              |
| VALUE  | 140904167          |
| HOST   | .dr-chuck.com      |
| PATH   | /                 |
| SECURE | No                |
| EXPIRES| At End Of Session |
Using Cookies to Support Sessions and Login / Logout
Some Web sites always seem to want to know who you are!
In most server applications, as soon as we meet a new browser - we create a session

We set a session cookie to be stored in the browser which indicates the session id in use

The creation and destruction of sessions is generally handled by a web framework or some utility code that we just use to manage the sessions
Session Identifier

- A large, random number that we place in a browser cookie the first time we encounter a browser.

- This number is used to pick from the many sessions that the server has active at any one time.

- Server software stores data in the session which it wants to have from one request to another from the same browser.

- Shopping cart or login information is stored in the session in the server.
Browser C: cook=97

Request

Server

Session 97

Create Session

index: "Please log in"

Response cook=97
Typing

We now have a session established but are not yet logged in.
Login / Logout

- Having a session is not the same as being logged in.
- Generally you have a session the instant you connect to a web site.
- The Session ID cookie is set when the first page is delivered.
- Login puts user information in the session (stored in the server).
- Logout removes user information from the session.
Click

Browser C

cook=97

Request

cook=97

Server

Session 97

login:
if good:
set user
Click
Browser C
  cook=97
Request
  cook=97
Response
  set user
  if good:
server
  login:
  user=phil
  Session 97
Server

Session 97
user=phil

Browser C
cook=97
Using Sessions for Other Stuff
Session 10
user=chuck
bal=$1000

Session 46
user=jan
bal=$500

withdraw:
bal=bal-100
Browser A
cook=10

Browser B
cook=46

Click

Server

Session 10
user=chuck
bal=$1000

Session 46
user=jan
bal=$500

withdraw:
bal=bal-100
Withdraw: 
bal = bal - 100
Review...
The web is “stateless” - the browser does not maintain a connection to the server while you are looking at a page. You may never come back to the same server - or it may be a long time - or it may be one second later.

So we need a way for servers to know “which browser is this?”

In the browser state is stored in “Cookies”

In the server state is stored in “Sessions”
You watch the YouTube video for 30 seconds.
Cookie/Session Summary

- Cookies take the stateless web and allow servers to store small “breadcrumbs” in each browser.

- Session IDs are large random numbers stored in a cookie and used to maintain a session on the server for each of the browsers connecting to the server.

- Server software stores sessions *somewhere* - each time a request comes back in, the right session is retrieved based on the cookie.

- Server uses the session as a scratch space for little things.