Stateful client/server interaction

Sites like amazon.com seem to "know who I am." How do they do this? How does a client uniquely identify itself to a server, and how does the server provide specific content to each client?

• HTTP is a stateless protocol; it simply allows a browser to request a single document from a web server

• today we'll learn about pieces of data called cookies used to work around this problem, which are used as the basis of higher-level sessions between clients and servers
What is a cookie?

- **cookie**: a small amount of information sent by a server to a browser, and then sent back by the browser on future page requests.

- Cookies have many uses:
  - Authentication
  - User tracking
  - Maintaining user preferences, shopping carts, etc.

- A cookie's data consists of a single name/value pair, sent in the header of the client's HTTP GET or POST request.
How cookies are sent

• when the browser requests a page, the server may send back a cookie(s) with it

• if your server has previously sent any cookies to the browser, the browser will send them back on subsequent requests

• alternate model: client-side JavaScript code can set/get cookies
Myths about cookies

• Myths:
  • Cookies are like worms/viruses and can erase data from the user's hard disk.
  • Cookies are a form of spyware and can steal your personal information.
  • Cookies generate popups and spam.
  • Cookies are only used for advertising.

• Facts:
  • Cookies are only data, not program code.
  • Cookies cannot erase or read information from the user's computer.
  • Cookies are usually anonymous (do not contain personal information).
  • Cookies CAN be used to track your viewing habits on a particular site.
A "tracking cookie"

- an advertising company can put a cookie on your machine when you visit one site, and see it when you visit another site that also uses that advertising company
- therefore they can tell that the same person (you) visited both sites
- can be thwarted by telling your browser not to accept "third-party cookies"
Where are the cookies on my computer?

• IE: *HomeDirectory*\Cookies
  • e.g. C:\Documents and Settings\jsmith\Cookies
  • each is stored as a .txt file similar to the site's domain name

• Chrome:
  C:\Users\*username*\AppData\Local\Google\Chrome\User Data\Default

• Firefox: *HomeDirectory*\.mozilla\firefox\***.default\cookies.txt
  • view cookies in Firefox preferences: Privacy, Show Cookies...
How long does a cookie exist?

- **session cookie**: the default type; a temporary cookie that is stored only in the browser's memory
  - when the browser is closed, temporary cookies will be erased
  - can not be used for tracking long-term information
  - safer, because no programs other than the browser can access them

- **persistent cookie**: one that is stored in a file on the browser's computer
  - can track long-term information
  - potentially less secure, because users (or programs they run) can open cookie files, see/change the cookie values, etc.
### Setting a cookie in PHP

<table>
<thead>
<tr>
<th>setcookie(&quot;name&quot;, &quot;value&quot;);</th>
<th>PHP</th>
</tr>
</thead>
<tbody>
<tr>
<td>setcookie(&quot;username&quot;, &quot;alllllison&quot;);</td>
<td></td>
</tr>
<tr>
<td>setcookie(&quot;age&quot;, 19);</td>
<td></td>
</tr>
</tbody>
</table>

- **setcookie** causes your script to send a cookie to the user's browser
- **setcookie** must be called before any output statements (HTML blocks, `print`, or `echo`)
- you can set multiple cookies (20-50) per user, each up to 3-4K bytes
- by default, the cookie expires when browser is closed (a "session cookie")
Retrieving information from a cookie

```php
$variable = $_COOKIE["name"];  // retrieve value of the cookie

if (isset($_COOKIE["username"])) {
    $username = $_COOKIE["username"];  
    print("Welcome back, $username.\n");
} else {
    print("Never heard of you.\n");
}

print("All cookies received:\n");
print_r($_COOKIE);```
What cookies have been set?

- **Chrome**: F12 → Resources → Cookies; **Firefox**: F12 → Cookies
Expiration / persistent cookies

```php
setcookie("name", "value", expiration);

$expireTime = time() + 60*60*24*7; // 1 week from now
setcookie("CouponNumber", "389752", $expireTime);
setcookie("CouponValue", "100.00", $expireTime);
```

- to set a persistent cookie, pass a third parameter for when it should expire
- indicated as an integer representing a number of seconds, often relative to current timestamp
- if no expiration passed, cookie is a session cookie; expires when browser is closed
- `time` function returns the current time in seconds
  - `date` function can convert a time in seconds to a readable date
Deleting a cookie

```php
setcookie("name", FALSE);
setcookie("CouponNumber", FALSE);
```

- setting the cookie to FALSE erases it

- you can also set the cookie but with an expiration that is before the present time:

  ```php
  setcookie("count", 42, time() - 1);
  ```

- remember that the cookie will also be deleted automatically when it expires, or can be deleted manually by the user by clearing their browser cookies
Clearing cookies in your browser

- **Chrome**: Wrench → History → Clear all browsing data...
- **Firefox**: Firefox menu → Options → Privacy → Show Cookies... → Remove (All) Cookies
Cookie scope and attributes

```javascript
setcookie("name", "value", expire, "path", "domain", secure, httponly);
```

- A given cookie is associated only with one particular **domain** (e.g. www.example.com)
- You can also specify a **path** URL to indicate that the cookie should only be sent on certain subsets of pages within that site (e.g. /users/accounts/ will bind to www.example.com/users/accounts)
- A cookie can be specified as **Secure** to indicate that it should only be sent when using HTTPS secure requests
- A cookie can be specified as **HTTP Only** to indicate that it should be sent by HTTP/HTTPS requests only (not JavaScript, Ajax, etc.; seen later); this is to help avoid JavaScript security attacks
Common cookie bugs

When you call `setcookie`, the cookie will be available in `_COOKIE` on the next page load, but not the current one. If you need the value during the current page request, also store it in a variable:

```php
setcookie("name", "joe");
print $_COOKIE["name"];  // # undefined
$name = "joe";
setcookie("name", $name);
print $name;              // # joe
```

- `setcookie` must be called before your code prints any output or HTML content:

```php
<!DOCTYPE html><html>
<?php
setcookie("name", "joe");  // # should precede HTML content!
```