14.1: Cookie Basics

- 14.1: Cookie Basics
- 14.2: Programming with Cookies
- 14.3: Sessions
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...

![Cookies](image)
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.

### 14.2: Programming with Cookies

- 14.1: Cookie Basics
- **14.2: Programming with Cookies**
- 14.3: Sessions
Cookies in JavaScript

```javascript
document.cookie = "username=smith"; // setting two cookies
document.cookie = "password=12345";
document.cookie = "age=29; expires=Thu, 01-Jan-1970 00:00:01 GMT"; // deleting a cookie
...
```

```javascript
var allCookies = document.cookie.split("; "); // ["username=smith", "password=12345"]
for (var i = 0; i < allCookies.length; i++) {
    var eachCookie = allCookies[i].split("="); // ["username", "smith"]
    var cookieName = eachCookie[0]; // "username"
    var cookieValue = eachCookie[1]; // "smith"
    ...
}
```

- JS has a global `document.cookie` field (a string)
- you can manually set/get cookie data from this field (sep. by ;), and it will be saved in the browser
- to delete a cookie, set it to 'expire' in the past

Provided Cookie library

```html
<!-- using the instructor-provided Cookie.js class -->
<script src="http://www.webstepbook.com/Cookie.js" type="text/javascript"></script>
```

```javascript
Cookie.set("username", "smith");
// (later)
alert(Cookie.get("username")); // smith
```

- we have written a Cookie.js helper class with methods set, get, exists, remove, and remember
Setting a cookie in PHP

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

setcookie("username", "martay");
setcookie("favoritecolor", "blue");
```

- `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
- technically, a cookie is just part of an HTTP header, and it could be set using PHP's `header` function (but this is less convenient, so you would not want to do this):
  ```php
  header("Set-Cookie: username=martay; path=/; secure");
  ```

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.
    
} else {
    print("Never heard of you.
    
}  
print("All cookies received:
    
print_r($_COOKIE);

  ```

- any cookies sent by client are stored in `$_COOKIES` associative array
- use `isset` function to see whether a given cookie name exists
- `unset` function deletes a cookie
Setting a persistent cookie in PHP

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

$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 its timeout in seconds
  - `time` function returns the current time in seconds
    - `date` function can convert a time in seconds to a readable date

Removing a persistent cookie

```php
setcookie("name", "", time() - 1);

setcookie("CouponNumber", "", time() - 1);
```

- if the server wants to remove a persistent cookie, it should set it again, passing a timeout that is prior to the present time
14.3: Sessions

- 14.1: Cookie Basics
- 14.2: Programming with Cookies
- 14.3: Sessions

What is a session?

- **session**: an abstract concept to represent a series of HTTP requests and responses between a specific Web browser and server
  - HTTP doesn't support the notion of a session, but PHP does
- sessions vs. cookies:
  - a cookie is data stored on the client
  - a session's data is stored on the server (only 1 session per client)
- sessions are often built on top of cookies:
  - the only data the client stores is a cookie holding a unique **session ID**
  - on each page request, the client sends its session ID cookie, and the server uses this to find and retrieve the client's session data
How sessions are established

- client's browser makes an initial request to the server
- server notes client's IP address/browser, stores some local session data, and sends a session ID back to client
- client sends that same session ID back to server on future requests
- server uses session ID to retrieve the data for the client's session later, like a ticket given at a coat-check room

Sessions in PHP: session_start

```php
session_start();
```

- **session_start** signifies your script wants a session with the user
  - must be called at the top of your script, before any HTML output is produced
- when you call **session_start**:
  - if the server hasn’t seen this user before, a new session is created
  - otherwise, existing session data is loaded into **$_SESSION** associative array
  - you can store data in **$_SESSION** and retrieve it on future pages
- complete list of PHP session functions
### Accessing session data

```php
$_SESSION["name"] = value;  # store session data
$variable = $_SESSION["name"];  # read session data
if (isset($_SESSION["name"])) {  # check for session data
    #store session data
    $variable = $_SESSION["name"];  # read session data
    if (isset($_SESSION["points"])) {
        $points = $_SESSION["points"];  # default
        print("You've earned $points points.\n");
    } else {
        $_SESSION["points"] = 0;  # default
    }
}
```

- the $_SESSION associative array reads/stores all session data
- use isset function to see whether a given value is in the session

### Where is session data stored?

- on the client, the session ID is stored as a cookie with the name PHPSESSID
- on the server, session data are stored as temporary files such as /tmp/sess_fcc17f071...
- you can find out (or change) the folder where session data is saved using the session_save_path function
- for very large applications, session data can be stored into a SQL database (or other destination) instead using the session_set_save_handler function
Session timeout

- because HTTP is stateless, it is hard for the server to know when a user has finished a session
- ideally, user explicitly logs out, but many users don't
- client deletes session cookies when browser closes
- server automatically cleans up old sessions after a period of time
  - old session data consumes resources and may present a security risk
  - adjustable in PHP server settings or with \texttt{session\_cache\_expire} function
  - you can explicitly delete a session by calling \texttt{session\_destroy}

Browsers that don't support cookies

```php
session_start();  # same as usual

# Generate a URL to link to one of our site's pages
# (you probably won't ever need to do this)
$orderUrl = "/order.php?PHPSESSID=\" . session_id() . "
```

- if a client's browser doesn't support cookies, it can still send a session ID as a query string parameter named \texttt{PHPSESSID}
  - this is done automatically; \texttt{session\_start} detects whether the browser supports cookies and chooses the right method
- if necessary (such as to build a URL for a link on the page), the server can find out the client's session ID by calling the \texttt{session\_id} function
Ending a session

```php
session_destroy();
```

- **session_destroy** ends your current session
- potential problem: if you call **session_start** again later, it sometimes reuses the same session ID/data you used before
- if you may want to start a completely new empty session later, it is best to flush out the old one:

```php
session_destroy();
session_regenerate_id(TRUE);  // flushes out session ID number
session_start();
```

Practice problem: Power Animal

- Write a page `poweranimal.php` that chooses a random "power animal" for the user.
- The page should remember what animal was chosen for the user and show it again each time they visit the page.
- It should also count the number of times that user has visited the page.
- If the user selects to "start over," the animal and number of page visits should be forgotten.
Implementing user logins

- many sites have the ability to create accounts and log in users
- most apps have a database of user accounts
- when you try to log in, your name/pw are compared to those in the database

Example simpsons database

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>email</th>
<th>password</th>
<th>id</th>
<th>name</th>
<th>teacher_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Bart</td>
<td><a href="mailto:bart@fox.com">bart@fox.com</a></td>
<td>bartman</td>
<td>1234</td>
<td>Krabappel</td>
<td></td>
</tr>
<tr>
<td>456</td>
<td>Milhouse</td>
<td><a href="mailto:milhouse@fox.com">milhouse@fox.com</a></td>
<td>fallout</td>
<td>5678</td>
<td>Hoover</td>
<td></td>
</tr>
<tr>
<td>888</td>
<td>Lisa</td>
<td><a href="mailto:lisa@fox.com">lisa@fox.com</a></td>
<td>vegan</td>
<td>9012</td>
<td>Stepp</td>
<td></td>
</tr>
<tr>
<td>404</td>
<td>Ralph</td>
<td><a href="mailto:ralph@fox.com">ralph@fox.com</a></td>
<td>catfood</td>
<td>10001</td>
<td>Computer Science 142</td>
<td>1234</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10002</td>
<td>Computer Science 143</td>
<td>5678</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10003</td>
<td>Computer Science 190M</td>
<td>9012</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10004</td>
<td>Informatics 100</td>
<td>123</td>
</tr>
</tbody>
</table>

- to test queries on this database, use username homer, password dOughnUt
Exercise: Simpsons grade display

Springfield Elementary

Grades for Bart:

<table>
<thead>
<tr>
<th>Course Name</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Science 142</td>
<td>B-</td>
</tr>
<tr>
<td>Computer Science 143</td>
<td>C</td>
</tr>
</tbody>
</table>

- Write a site where Springfield Elementary students can log in to check their grades.
- Implement a user login system that verifies proper user names and passwords.
- A student should only be able to view his/her own grades.

"Remember Me" feature

- How might an app implement a "Remember Me" feature, where the user's login info is remembered and reused when the user comes back later?
- Is this stored as session data? Why or why not?
- What concerns come up when trying to remember data about the user who has logged in?