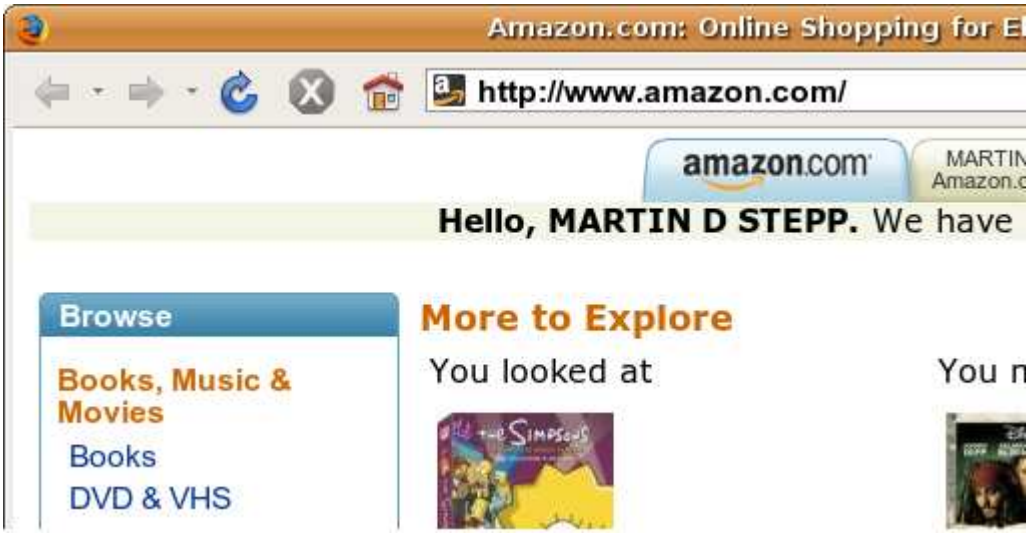
Cookies and Sessions

CSE 190 M (Web Programming) Spring 2008
University of Washington

Reading: Sebesta 12.12 - 12.13, 10.6

References: tizag.com sessions, cookies; Codewalkers

Stateful client/server interaction



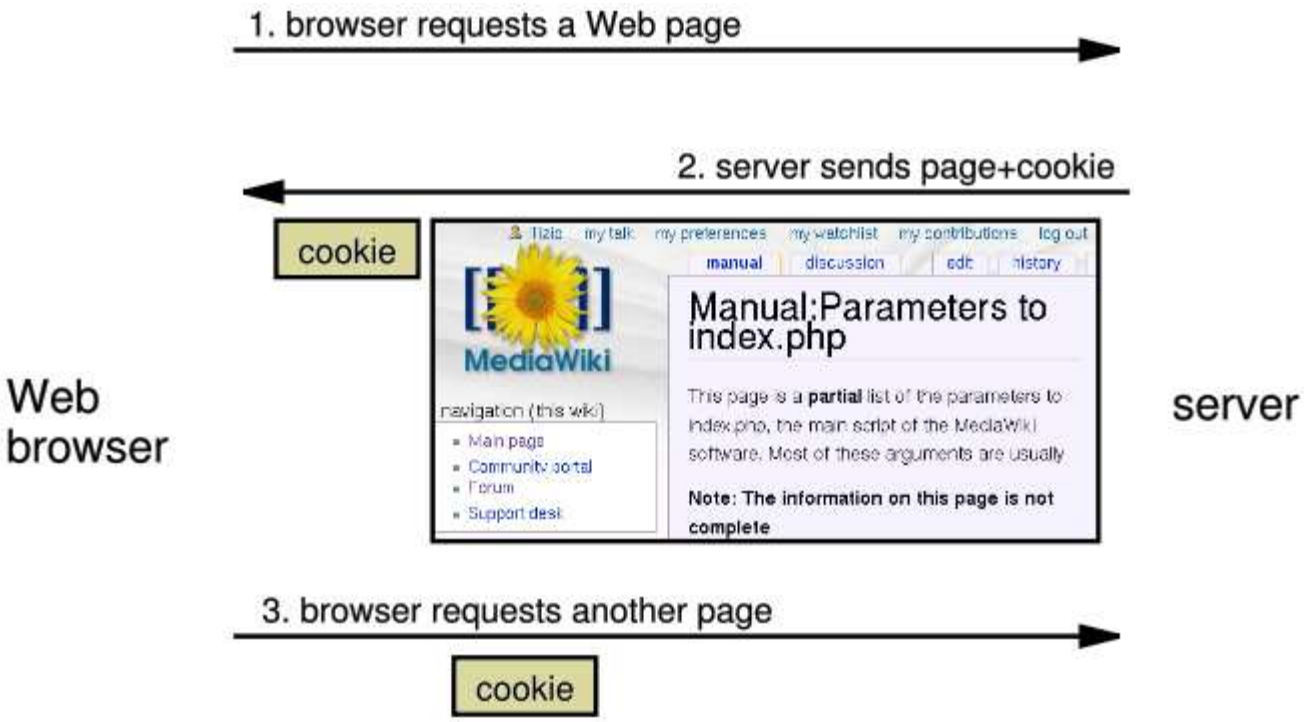
Sites like amazon.com seem to "know who I am." How do they do this? How does a client uniquely identify itself to a server, and how does the server provide specific content to each client?

- HTTP is a **stateless** protocol; it simply allows a browser to request a single document from a web server
- in these slides, 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

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.

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.

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
- Firefox: *HomeDirectory*\.mozilla\firefox*???*.default\cookies.txt
  - view cookies in Firefox preferences: Privacy, Show Cookies...

Setting a cookie in PHP

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

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

- `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):

```
header("Set-Cookie: username=martay; path=/; secure");
```

Retrieving information from a cookie

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

```php
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);
```

- 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);
```

```php
$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);
```

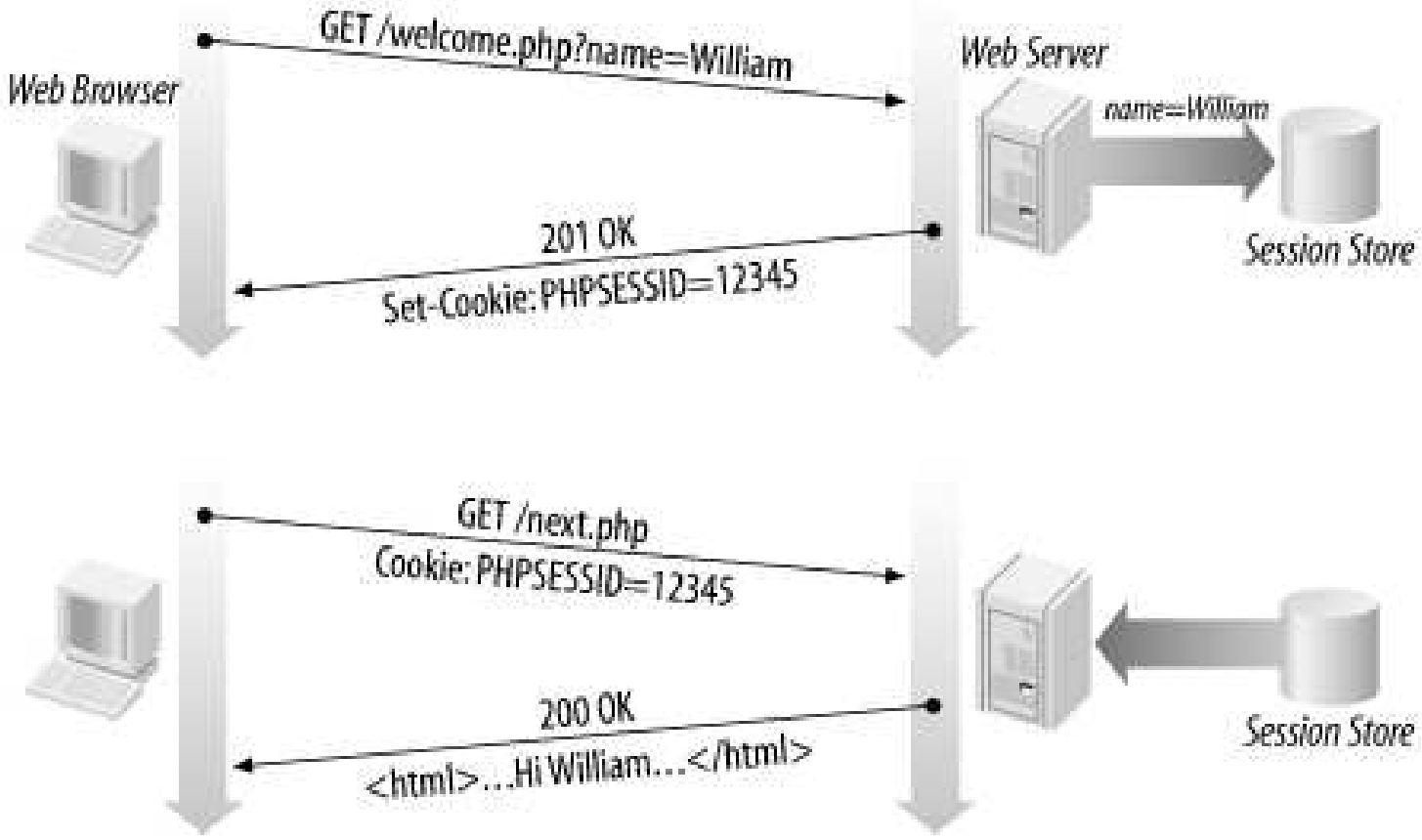
```php
setcookie("CouponNumber", "", time() - 1);
```

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

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
```

```php
if (isset($_SESSION["points"])) {
    $points = $_SESSION["points"];
    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

Browsers that don't support cookies

```php
session_start();

# Generate a URL to link to one of our site's pages
$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 `PHPSESSID`
  - this is done automatically; `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 `session_id` 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 `session_cache_expire` function
    - you can explicitly delete a session by calling `session_destroy`

Practice problem: remembering query

- Modify the `movie.php` movie search script from previous lectures so that it remembers the current user's last query (if any), and offers the user a chance to search for it again, such as:
    - Welcome back! Would you like to repeat your recent search for Fight Club?
- Pretend that the movie-search program is running on a system that wants to limit repeated usage by particular users. Add code so that a given user can only conduct one session per day.