

CSE 154

LECTURE 13: SESSIONS

How long does a cookie exist?

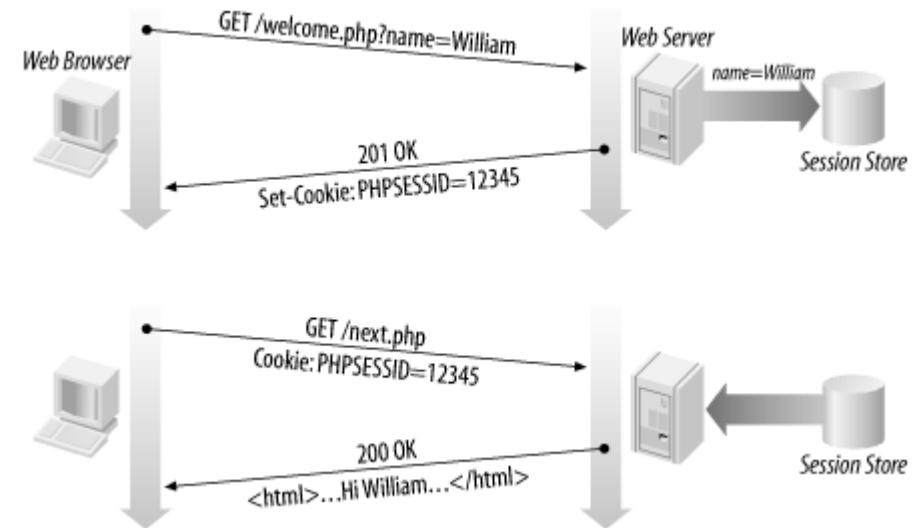
- **session cookie** : the default type; a temporary cookie that is stored only in the browser's memory
 - when the browser is closed, temporary cookies will be erased
 - can not be used for tracking long-term information
 - safer, because no programs other than the browser can access them
- **persistent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

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 (as a cookie)
- client sends that same session ID (cookie) back to server on future requests
- server uses session ID cookie to retrieve its data for the client's session later (like a ticket given at a coat-check room)



Cookies vs. sessions

- **duration:** sessions live on until the user logs out or closes the browser; cookies can live that long, or until a given fixed timeout (persistent)
- **data storage location:** sessions store data on the server (other than a session ID cookie); cookies store data on the user's browser
- **security:** sessions are hard for malicious users to tamper with or remove; cookies are easy
- **privacy:** sessions protect private information from being seen by other users of your computer; cookies do not



Sessions in PHP: session_start

```
session_start();
```

PHP

- `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

```
$_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
}
```

PHP

- 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 [session_cache_expire](#) function
 - you can explicitly delete a session by calling [session_destroy](#)

Ending a session

```
session_destroy();
```

PHP

- `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:

```
session_destroy();
```

```
session_regenerate_id(TRUE); # flushes out session
```

```
ID number
```

```
session_start();
```

PHP

Common session bugs

- `session_start` doesn't just begin a session; it also reloads any existing session for this user. So it must be called in every page that uses your session data:

```
# the user has a session from a previous page
print $_SESSION["name"];    # undefined

session_start();
print $_SESSION["name"];    # joe
```

PHP

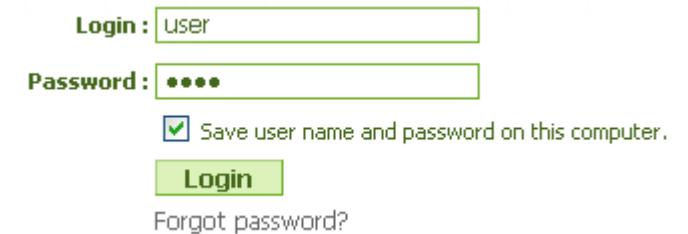
- previous sessions will linger unless you destroy them and regenerate the user's session ID:

```
session_destroy();
session_regenerate_id(TRUE);
session_start();
```

PHP

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



Login :

Password :

Save user name and password on this computer.

[Forgot password?](#)

"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?



The image shows a login form with the following elements:

- Login :** A text input field containing the text "user".
- Password :** A text input field containing four dots "••••".
- Save user name and password on this computer. (This checkbox and its label are circled in red in the original image.)
- Login** (A green button)
- Forgot password?

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.

