

Sessions

ian.mcloughlin@gmit.ie

Sessions

HTTP treats each request–response individually.

How can we let users identify themselves to the server as the same user who made a previous request?

Needed to enable such things as shopping carts.

Sessions provide a mechanism for this.

Servers can provide a unique key in a response, which can be used in the next request.

Amazon were one of the first companies to design this functionality.

Cookies

Set-Cookie is a header that a server can send in a response.
name=value; is typically what Set-Cookie is set to.

Browsers store the information on the client side.

Cookie is a header the client can send in a request.

Usually it is set to something a server has sent before.

Set-Cookie example

HTTP/1.0 200 OK

Content-type: text/html

Set-Cookie: yummy_cookie=choco

Set-Cookie: tasty_cookie=strawberry

<p>Hello, world!</p>

Cookie example

GET /sample_page.html HTTP/1.1

Host: www.example.org

Cookie: yummy_cookie=choco; tasty_cookie=strawberry

Cookie Lifetimes

Cookies usually don't last forever.

Session cookies typically expire when a browser window is closed.

Sometimes browsers will restore session cookies, however, as a feature.

Persistent cookies aren't quite permanent – the server sets an expiration date and time.

```
Set-Cookie: id=a3; Expires=Wed, 21 Oct 2015 07:28:00 GMT;
```

Using Cookies in JavaScript

document.cookie is used to get and set cookies in JavaScript.

```
document.cookie = "yummy_cookie=choco";  
document.cookie = "tasty_cookie=strawberry";  
console.log(document.cookie);  
// logs "yummy_cookie=choco; tasty_cookie=strawberry"
```

Cookie security

User identification is often done through the use of Cookies.

Servers give a user a unique cookie to be sent with each request.

Stealing cookies is a good way to hijack a user session.

HttpOnly can be tacked onto cookies so that they are not available in JavaScript.

Secure will ensure cookies are only sent over secure connections (HTTPS).

```
Set-Cookie: id=a3f; Expires=...; Secure; HttpOnly
```


Data in web applications

Databases are usually used to store user data on the server side.

Interactive web applications can trigger lots of HTTP requests in short periods of time.

Traditional databases will (often) write to the hard disk.

Hard-disk access is slow.

RAM is fast.

In-memory databases can make the user experience better.

Data in web applications

Databases are usually used to store user data on the server side.

Interactive web applications can trigger lots of HTTP requests in short periods of time.

Traditional databases will (often) write to the hard disk.

Hard-disks are slow.

In-Memory Data Stores

In-Memory databases can make the user experience better.

RAM is much faster than hard-disks.

Good for data that is not hanging around for long, and can easily be discarded.

Identification credentials are a good example.

Worst-case scenario, the user is asked to login again.

Not great for large chunks of data that needs long-term persistence, like photos.

Redis

Redis is an in-memory data structure store.

Supports data structures such as strings, hashes, lists, sets, sorted sets, etc.

Persistence is supported, just not emphasised.

A lot a like your usual database.

Difficult to appreciate until you have a high-traffic web application.