CS1116/CS5018

Web Development 2

Dr Derek Bridge

School of Computer Science & Information Technology
University College Cork

Statelessness

- HTTP is a **stateless protocol**:
  - each request is independent: by default, the server has no memory of previous requests
- Adequate for HTTP's original purpose:
  - a client contacts a server and requests a document
  - the server sends the requested document to the client
- Inadequate for situations where it can be useful to recognise repeat contacts, e.g.:
  - a client which has contacted the server in the past
  - a sequence of requests from the same client within a short period of time (a **session**)

State management

- **IP addresses**:
  - keep a record of clients' IP addresses
- **Cookies**:
  - send the client some data, which it sends back to you
- Other methods:
  - **URL rewriting**:
    - include a user identifier in the query part of a URL
  - **Hidden fields**:
    - include a non-visible field in forms so that a user identifier gets submitted with every request

IP addresses for state management

- An HTTP request header often contains the client's IP address
- A server-side program can obtain the IP address from the `os.environ` dictionary:

```
from os import environ

ip_address = environ.get('REMOTE_ADDR')
```

- Using IP addresses is sometimes adequate
  Q: What are the problems of trying to identify users using IP addresses?

Cookies for state management

- A cookie is a small amount of data (a name/value pair)
  - E.g. `id=cust123`.
  - Each cookie can be no more than 4kb in size
- If a browser has sent a request to a server, the server can include a cookie in its response (in a HTTP header line)
- If the browser has cookies enabled, it stores the cookie
- Next time the browser sends a request to the same server (or one in the same domain), it includes the cookie in its request (a HTTP header line)
- This enables the server to know that it has previously received requests from this client

Cookies example, part I

- Your browser sends a request to `www.amazon.co.uk`:

```
GET /index.html
```

- A server-side program stores information, e.g. in its database, about your visit
- The server's response includes a cookie:

```
HTTP/1.1 200 OK
Set-Cookie: id=cust123; path=/; domain=.amazon.co.uk
…
```

- If cookies are enabled in your browser, your browser stores the cookie

Cookies example, part II

- On a subsequent occasion, you visit `www.amazon.co.uk` again (or the `amazon.co.uk` domain)
- Your browser includes the cookie in the request:

```
GET /index.html
Cookie: id=cust123
```

- On detection of the cookie, server-side programs know that you have made requests on previous occasions and can use the cookie data, e.g. to look you up in the database
- Q: What are the problems of trying to identify users using cookies?

Two types of cookie: persistent and in-memory

- **Persistent cookies:**
  - The server includes an expiry date in the cookie:

```
HTTP/1.1 200 OK
Set-Cookie: id=cust123;
        expires=Sun, 17-Jan-2040 19:14:07 GMT;
        path=/; domain=.amazon.co.uk
…
```

  - The browser stores the cookie on the client's hard disk
  - The browser deletes the cookie when it expires
- Persistent cookies are useful for identifying clients which have contacted the server in the past

Two types of cookie: persistent and in-memory

- **In-memory cookies:**
  - The server does not include an expiry date:

```
HTTP/1.1 200 OK
Set-Cookie: id=cust123; path=/; domain=.amazon.co.uk
…
```

  - The browser (ordinarily) stores the cookie in main memory
  - The browser (ordinarily) deletes the cookie when the browser is shut down
- In-memory cookies are useful for sessions

Sending a cookie

- Create a `SimpleCookie` object, and print it:

```
from http.cookies import SimpleCookie

cookie = SimpleCookie()
cookie['id'] = 'cust123'
cookie['id']['expires'] = 157680000
print(cookie)
print('Content-Type: text/html')
print()
```

- Questions:
  - What kind of cookie is this?
  - When does it expire?
  - How did we make sure the cookie is in the HTTP header?

Receiving a cookie

- A server-side program can:
  - obtain the HTTP cookie header from the `environ` dictionary
  - test whether the HTTP cookie header is empty or not
  - if not empty, it can populate a `SimpleCookie` object with the cookie data, and then do things with that data

```
from os import environ
from http.cookies import SimpleCookie

cookie = SimpleCookie()
http_cookie_header = environ.get('HTTP_COOKIE')
if not http_cookie_header:
    # the client did not send a cookie
else:
    cookie.load(http_cookie_header)
    # now you can do things with the cookie
```

`count_visits_by_cookie.py`: a simple example

```
#!/usr/local/bin/python3

from cgitb import enable
enable()

from os import environ
from http.cookies import SimpleCookie

cookie = SimpleCookie()
http_cookie_header = environ.get('HTTP_COOKIE')
if not http_cookie_header:
    cookie['num_visits'] = 1
else:
    cookie.load(http_cookie_header)
    if 'num_visits' not in cookie:
        cookie['num_visits'] = 1
    else:
        cookie['num_visits'] = int(cookie['num_visits'].value) + 1
cookie['num_visits']['expires'] = 157680000
print(cookie)
print('Content-Type: text/html')
print()
print("""
    <!DOCTYPE html>
    <html lang="en">
        <head>
            <meta charset="utf-8" />
            <title>Welcome!</title>
        </head>
        <body>
            <p>
                You have visited this page %s times.
            </p>
        </body>
    </html>""" % (cookie['num_visits'].value))
```

Want to read more?

- Wikipedia's section on privacy and third-party cookies
- Wikipedia's section on cookie theft and session hijacking
  - Never send credit card or similar data in a cookie.
  - If data is the least bit sensitive (e.g. customer id), then use HTTPS (HTTP + SSL)