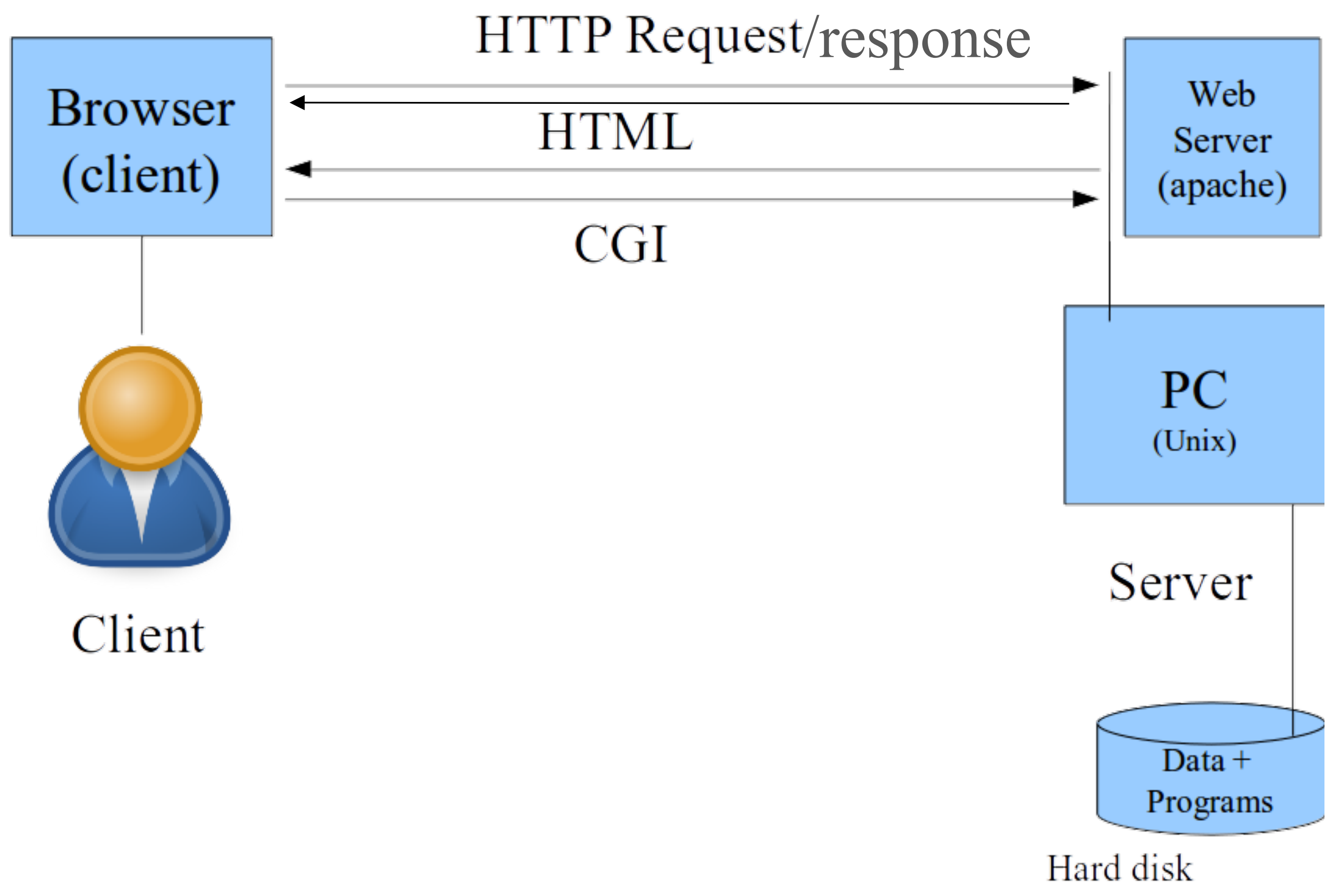
COMP 206 – Intro to Software Systems

Lecture 20 – Running your code through a web server

November 19th, 2018

Web Software Protocols:
Last time static HTML. Today dynamic pages.

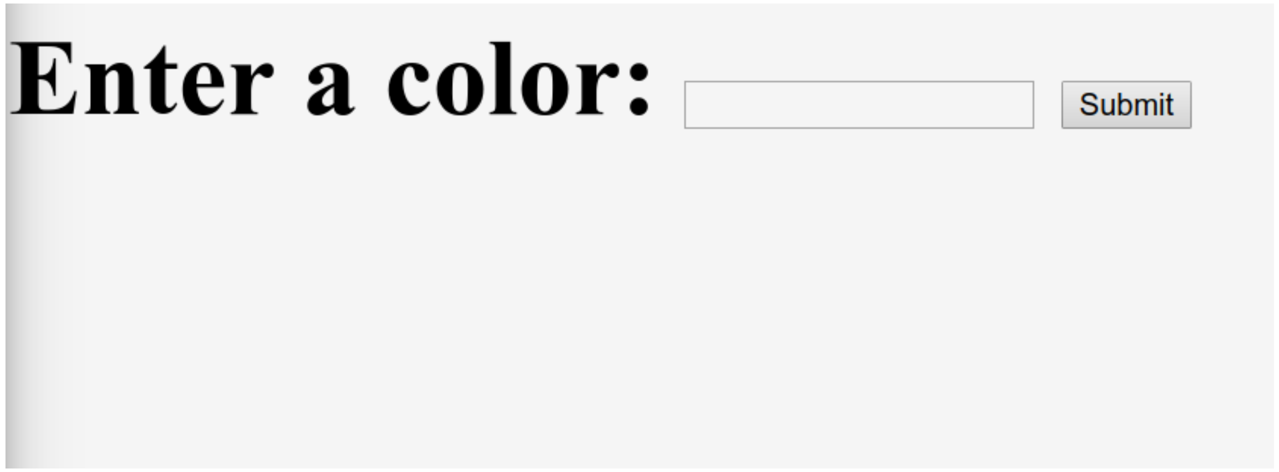


Common Gateway Interface (CGI)

- Sometimes we want more than static code.
  - Example: Enter banking info into a form, expect a purchase to be made when we press enter
- CGI is a method the server to be offer such applications to web clients and to mediate information flow between the client and the program
- CGI connects with both HTML and HTTP:
  - HTML is used to specify which programs are available and describe what user interface elements should be used to interface with this program
  - HTTP is used to transmit the user's inputs to the server.
  - The HTTP command used here is typically "POST”, which means the browser is sending some data (whatever we entered)

CGI Example

```
<FORM ACTION="color.cgi" METHOD="POST”>
Enter a color:
<INPUT TYPE="text" NAME="color">
<INPUT TYPE="submit">
</FORM>
```



CGI: How does it work? Part 1, prepare and run the program

- The web server receives the user's inputs within the HTTP request

- Instead of grabbing a page from disk, instead it looks for the program named in the FORM ACTION. This can be C, Python, Perl, and many more.

- The server runs the program in a special way:
  - It passes it the data that the user entered using environment variables, command-line arguments and standard input (not too different from “<“ on the shell).
  - It captures the standard output (think about the ">" operator on the shell).

CGI: How does it work? Part 2: send back the response

- The standard output is passed back to the browser as the HTTP response.

- This means CGI must print both some HTTP fields and the full HTML file as its output. That defines the nature of most CGI (it looks like a lot of structured printing).

- If the page generated by CGI contains another FORM tag, this can then call the next program in line, or the same program again with new arguments.

Some typical use cases for CGI

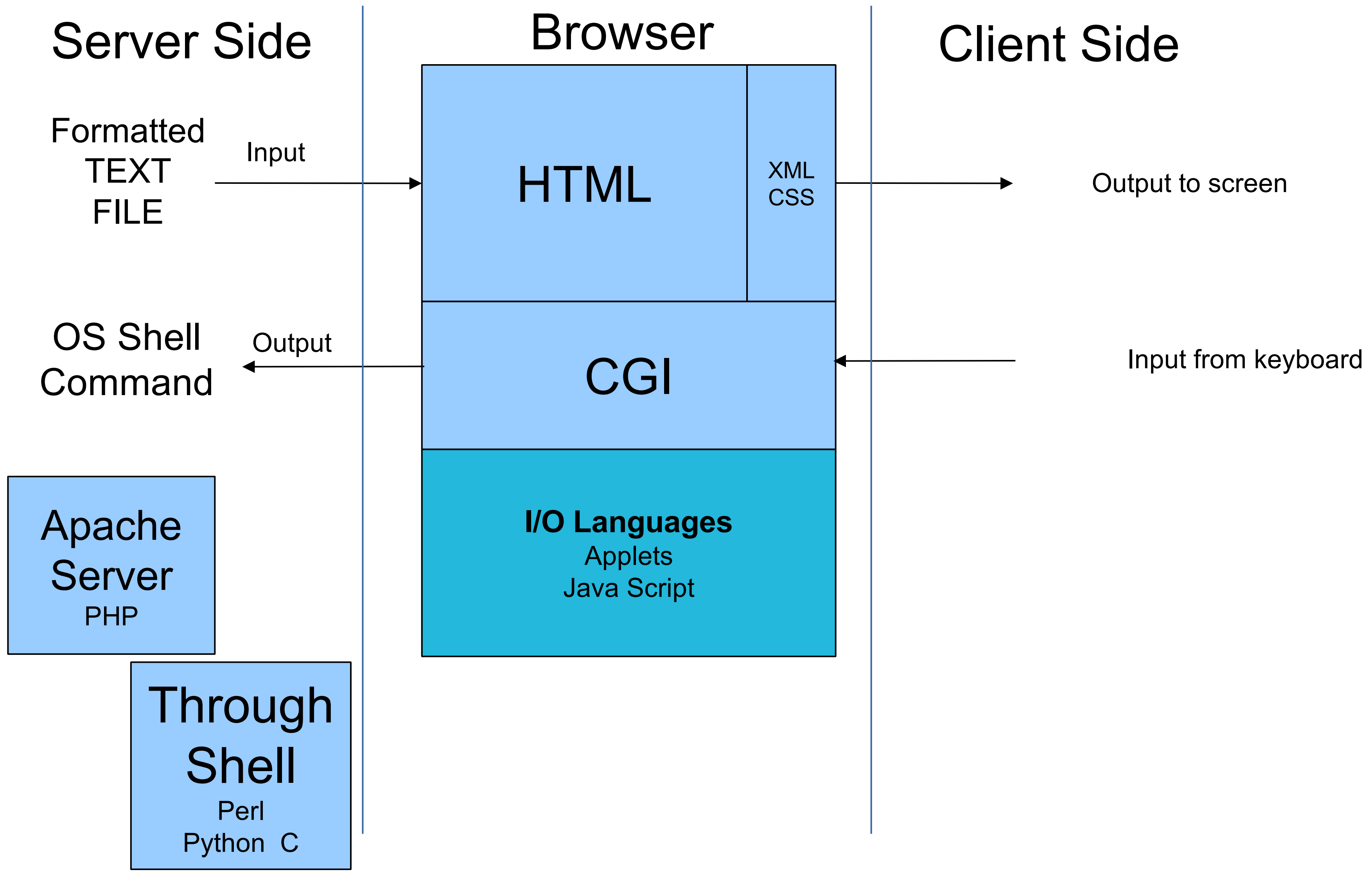
- Read a database file:
  - Instead of having a pre-coded page with our bank balances, we can run the program “check_balance.c” that grabs our latest transactions ($$$!)
- Connect to a live running program on the server (or elsewhere):
  - Our browser can only connect to the web server, which should not do anything except servicing web connections, but the CGI can be customized to do chat, play games, share files, etc.
- Make the appearance of our pages more interesting:
  - Instead of picking only one color for the background, CGI can auto-generate a different version of the page for each time of day

Think about a CGI-executed program

- It is not like typing "$ ./a.out" on our terminal

- Since no terminal is attached to this program:
    - Where does the program get its standard input?
    - Where should the text from standard output be displayed?

Your Browser as I/O



CGI Technicals

- “Script” (program) must reside in a web-visible directory
- Some web servers require that it ends in .cgi. Others allow .py, a.out, etc.
- World executable (mode 755, for example) (the directory too).
- Information for setting it up at mimi is here:
  - https://cs.mcgill.ca/docs/labs/webservers/

CGI Program Structure (typically)

- Check form fields
  - Ensure all input is of the correct form, otherwise just return a “retry” page, which is the same as before but instructs what to change
- Perform action
  - This is up to you and totally depends on the application!
  - What logic must run to decide on the next page the user sees?
  - Database interfaces available
- Generate HTTP + HTML output
  - (HTTP is the way to returning data, HTML is the formatting.)
  - Print statements are simplest
  - Often we will not generate everything from scratch. Rather, load a file with most of the contents and “find/replace” placeholders with the dynamic content

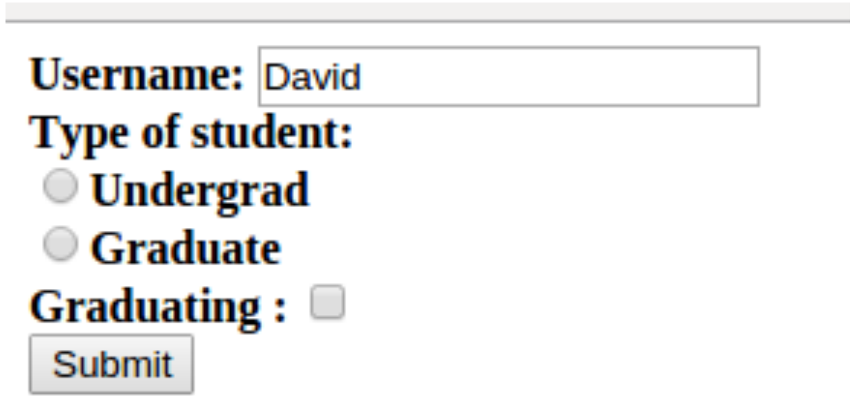
Example Dynamic HTML

Form tag signals dynamic content

Name of script to execute

Method to pass data: get or post

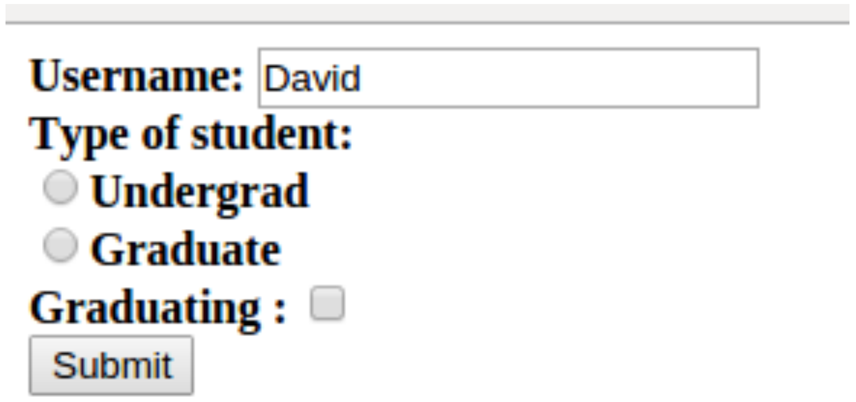
```
<html>
<body>
<form name="input" action="./cgi-bin/week10c_sample_cgi.cgi"
method="get">
  <b>Username: <input type="text"   name="user"><br />
  Type of student: <br />
  <input type="radio" name="student" value="ugrad">Undergrad<br />
  <input type="radio" name="student" value="grad">Graduate<br />
  Graduating :
  <input type="checkbox" name="graduating"> <br />
  <input type="submit" value="Submit">
</form>
</body>
</html>
```


Username: David
Type of student:
Undergrad
Graduate
Graduating :

Example Dynamic HTML

```
<html>
<body>
<form name="input" action=”lecture20_sample_cgi.cgi" method="get">
  <b>Username: <input type="text"   name="user"><br />
  Type of student: <br />
  <input type="radio" name="student" value="ugrad">Undergrad<br />
  <input type="radio" name="student" value="grad">Graduate<br />
  Graduating :
  <input type="checkbox" name="graduating"> <br />
  <input type="submit" value="Submit">
</form>
</body>
</html>
```

User interface elements


Username: David
Type of student:
Undergrad
Graduate
Graduating :

A Long Running Dynamic Page

- Original HTML file displayed in browser, leads to first call of CGI script
- Repeat forever:
  - Script outputs HTML for display that includes a form field linking to a CGI script (can be the same one again or different)
  - This is displayed in the browser, user interacts
  - The CGI is called again

Structure Refinement Pseudo-code

```
Get the data from the user’s previous action on the form
If( form has not been filled out yet )
    ...display blank form...
Else If( we have valid form data )
    ...perform action, display results (or next form)...
Else
    ...display error message (maybe repeating form)...
```

CGI Example 3

- More significant computations and dealing with uploaded files:
  - week11_image_page.html
  - week11_image_cgi.cgi

Security and CGI: A Big Downfall

- CGI scripts need guard against malicious entry.
- Watch out when passing fields from the web to the shell
  - e.g. system(command_string)
  - what if the value is "; cat /etc/passwd" ...
- Example (imperfect) solutions:
  - Quote calls to program names. Ensures white-space characters are not interpreted incorrectly
  - Refuse to handle strings that are longer than available buffer
  - Sanitize by removing special characters

Multi-step Interactions

- HTTP is "stateless"
  - Each page/web request is independent.
  - There is no natural notion of the next interaction or the last one.
    - » When a request arrives, it could be from the same person who made the previous one, or maybe not.
- An *connected set of interactions* must somehow implement
  - persistence (information that is remembered)
  - identity (an ID to something like it)
  -
- Approaches: manually, cookies, hidden form fields, URL encoding.

More state: Trivial (bad) idea

- Continuity via ID
- On each web form, include 2 manual fields
  - ID number
  - step number (in a series of steps)
    - e.g. first register, then pick and item, then fill in credit card, then fill in shipping address, then …

- Problem: don't want to have to fill this out repeatedly,
- Problem: could lie (too easily).

Automatic Transfer of ID

- Use fact that form fields are automatically sent to sever when forms are submitted.

- The server can pre-fill fields that hold state information (like your ID).
- These can, further, be hidden from the user
  - to make it more attractive
  - to reduce chances of tampering

Session Maintenance

- Correlate requests from same user
  - Assign session key on first contact
  - Incorporate session key in form or in URL
- Options:
  1. In form: use hidden input field:
     1. <input type="hidden" name="session" value="1f9a2">
  2. In URL:
     - » http://myhost.com/cgi-bin/myprog.py/1f9a2
     - » passed in environment (os.environ[...]):
       - » PATH_INFO=/1f9a2
       - » PATH_TRANSLATED=<rootdir>/1f9a2

What are cookies?

- Way for web site to store data on client browser.
- Cookies are sent in invisible header the precedes actual HTML data.
- Web site send you cookies.
- You send cookies back.
- Each cookies has: name, value, web site binding.

Integrated Example

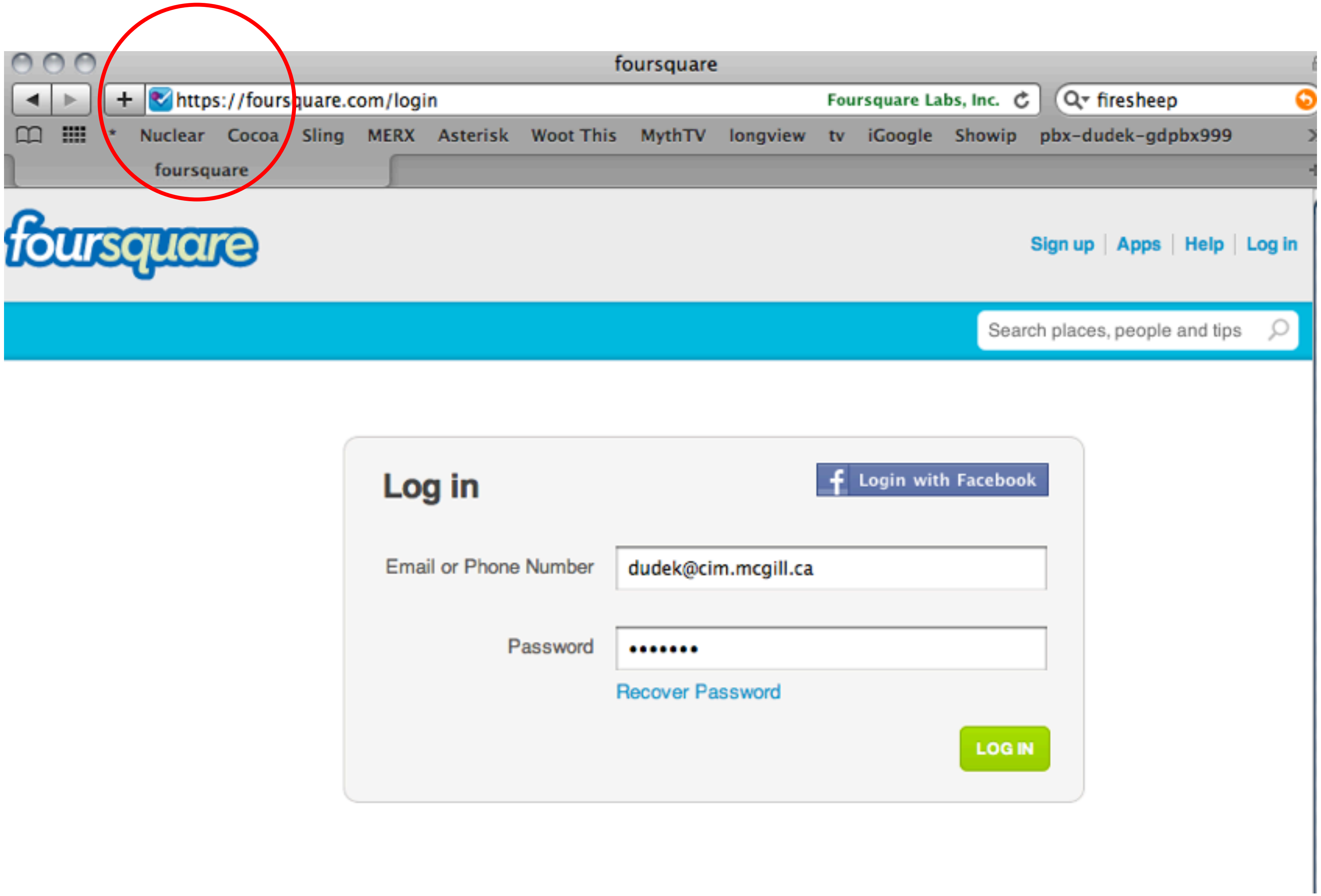
How do Facebook, Twitter, FourSquare,etc. manage your "state"?

1) Login via a secure (encrypted) page

- Your password is safe.
- Encryption is costly though.

2) Store session data via cookies

- Provide temporary authorization to access you account
- Often not encrypted.
- If the network connection is also not encrypted, we might be at risk.


foursquare

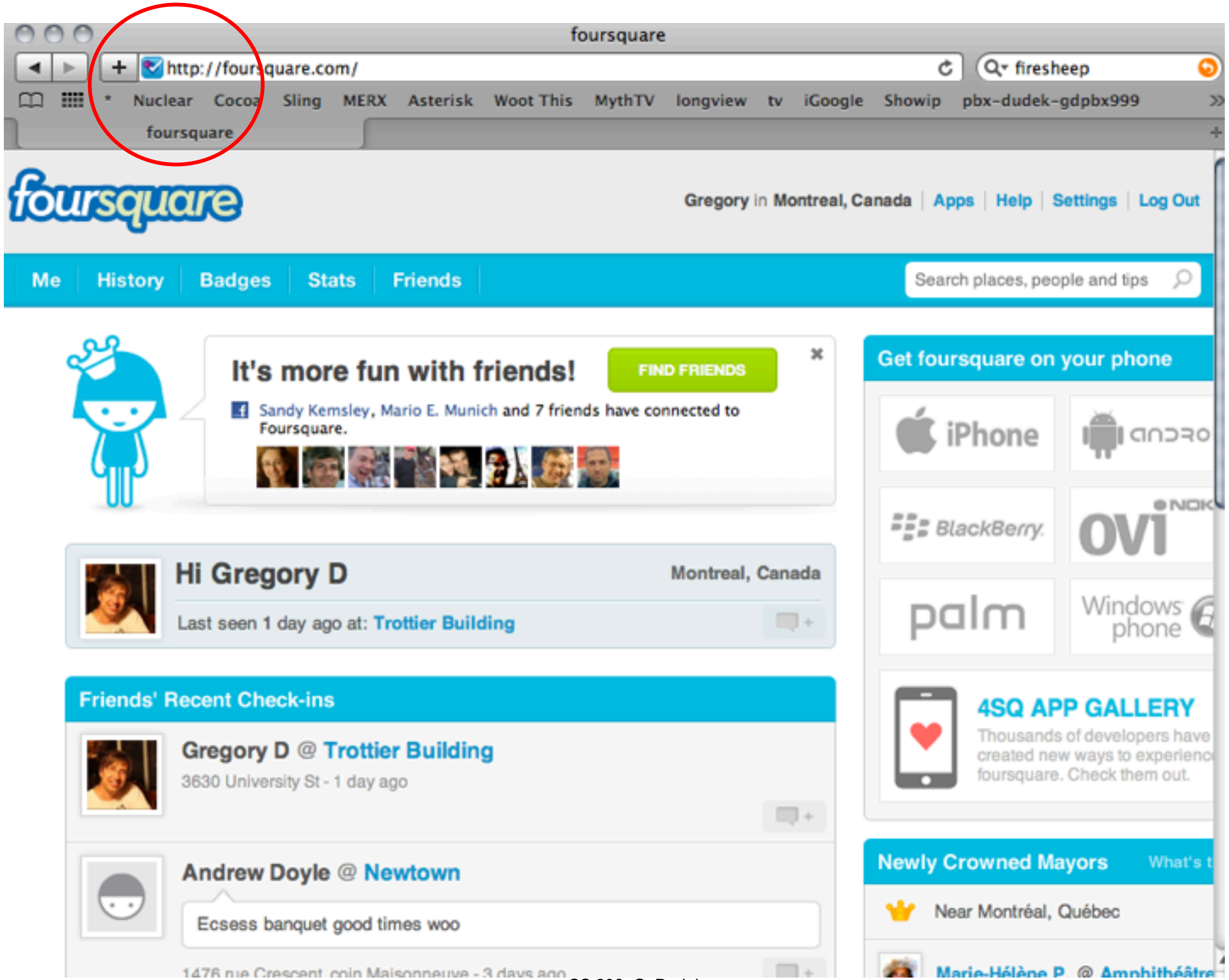
https://foursquare.com/login Foursquare Labs, Inc.

firesheep

Log in

Email or Phone Number: dudek@cim.mcgill.ca

Password: •••••••



foursquare

http://foursquare.com/

firesheep

Nuclear Cocoa Sling MERX Asterisk Woot This MythTV longview tv iGoogle Showip pbx-dudek-gdpbx999

foursquare

Gregory in Montreal, Canada | Apps | Help | Settings | Log Out

Me History Badges Stats Friends

Search places, people and tips

It's more fun with friends! FIND FRIENDS

Sandy Kemsley, Mario E. Munich and 7 friends have connected to Foursquare.

Get foursquare on your phone

iPhone

BlackBerry

OVI

palm

Windows phone

4SQ APP GALLERY

Thousands of developers have created new ways to experienc foursquare. Check them out.

Hi Gregory D Montreal, Canada

Last seen 1 day ago at: Trottier Building

Friends' Recent Check-ins

Gregory D @ Trottier Building

3630 University St - 1 day ago

Andrew Doyle @ Newtown

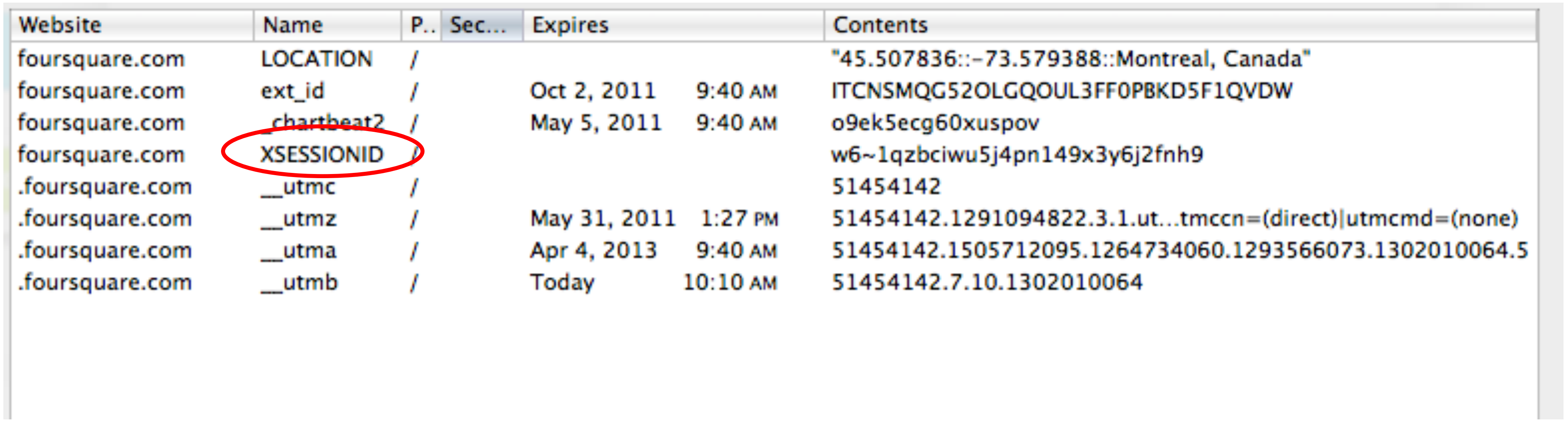
Ecsess banquet good times woo

1476 rue Crescent, coin Maisonneuve - 3 days ago

Newly Crowned Mayors What's t

Near Montréal, Québec

Marie-Hélène P. @ Amphithéâtre



Website	Name	P..	Sec...	Expires		Contents
foursquare.com	LOCATION	/				"45.507836::-73.579388::Montreal, Canada"
foursquare.com	ext_id	/		Oct 2, 2011	9:40 AM	ITCNSMQG52OLGQOUL3FF0PBKD5F1QVDW
foursquare.com	_chartbeat2	/		May 5, 2011	9:40 AM	o9ek5ecg60xuspov
foursquare.com	XSESSIONID	/				w6~1qzbciwu5j4pn149x3y6j2fnh9
.foursquare.com	__utmc	/				51454142
.foursquare.com	__utmz	/		May 31, 2011	1:27 PM	51454142.1291094822.3.1.ut...tmccn=(direct)|utmcmd=(none)
.foursquare.com	__utma	/		Apr 4, 2013	9:40 AM	51454142.1505712095.1264734060.1293566073.1302010064.5
.foursquare.com	__utmb	/		Today	10:10 AM	51454142.7.10.1302010064

Can this be sniffed?

- Normal network interactions are broadcast to all computers on local chunk of the network.
- Your computer ignores all the ones not meant for you.

- Filtered by IP address.
- We can turn off filtering, so processes can “see” all traffic nearby.
  - **Promiscuous** mode.

Exercises

- Try to compile a CGI program and get it connected to a simple website on www.mimi.cs.mcgill.ca/~<your_username>
- Recall:
  - You may need to do some configuration on your account to ensure this is allowed at all for you. Don't get stuck looking for code bugs if the "hello world" version fails, especially if the warning is about "unauthorized"
  - Make sure that everything is in the correct path (within public_html/cgi-bin) and that all programs are executable by the server (chmod a+x)

- See if you can accomplish a browser/server "session" that maintains state.