CSE 1710

Lecture 22

Net-Centric Programming, Part III

Part2

Learning Outcomes

– Understand the notion of a URL that contains a query string.

– Use string processing to manipulate query strings

– Retrieve content from a web server

  - with safe guards in place
  - understand the purpose of the safe guards

– Scrape data from a remote source

– Scrape data from a remote source repeatedly, introduce a pause

Smallish Recap

HTTP is a client-server protocol.

The client sends requests for remote resources; the server retrieves the requested resources (if nothing goes wrong).

These resources are named using the URL convention.

The requests can be either a GET or a POST request.

The server listens and responds to requests.

If the server can provide the requested resource, it will return a code of 200 and accompany the response with a payload that contains the requested resource.

If the server cannot provide the requested resource, if will return a descriptive code.

Examples: code 404 (resource not found), code 500 (resource is there but insufficient permissions).

Why do we care?

We can *programmatically* retrieve remote resources.

We can *scrape* the retrieved data for information.

Your apps can make use of whatever resources you need from the web.

It is important for application development to allow Internet communication between programs, even if the applications are running on different operating systems, with different technologies and programming languages.

There are many challenges: security, firewall, proxy server problems

Aside:

- presently, many applications communicate using Remote Procedure Calls (RPC) between objects, such as DCOM and CORBA.
- Other approaches include: Simple Object Access Protocol (SOAP) and JavaScript Object Notation (JSON)

A crash course in the stock market

(We will use a pretend stock market for this module)

A public company is a company that offers its stock/shares for sale to the general public, typically through a stock exchange,

A public company is represented by a two-character symbol e.g., ".AB", ".PP"

At any given point in time, the company's shares have a selling price.

"buy low, sell high"

Query Strings

Let's have a look at http://www.cse.yorku.ca/~roumani/jba/ase/

This html code contains a form element.
The form gathers together elements into a meaningful whole.
The form elements can consist of any number of input elements: text fields, radio buttons, checkboxes. In addition, the form elements typically consists of a "submit" button.

what happens when we press the form's submit button?

Query Strings

When we invoke the form's submit button, two things happen.
1) the following is composed:

```
"http://www.cse.yorku.ca/~roumani/jba/ase/se.cgi"
+ "?" + "hrss" + "=" + ".NN"
```

This evaluates to the following URL:

```
http://www.cse.yorku.ca/~roumani/jba/ase/se.cgi?hrss=.NN
```

2) The browser sends a "GET" request with the above-named URL

Now, what does the server do?

Serving Content as per the HTTP Protocol

1. Listen on port 80
2. If and when an HTTP request arrives ("GET" or "POST"), start a process to handle it ("fork")
3. Extract the path/file from the URL
4. Check whether file exists
   If not, return status 404.
5. Check whether file is reachable & readable (file permissions?)
   If not, return status 403
6. Determine the content type (static vs dynamic)

Static Content

7. Return with status 200 (OK) and a type header.

8. Serve file as the payload.

9. Close HTTP session.
   Or, on keep-alive, wait brief time for another request.

Dynamic Content (CGI)

7. Masquerade as file owner.
8. Check that file is executable by owner.
   If not, return status 500.
9. Run the file and capture its output.
10. Check the validity of the output.
   Not valid? Return status 500.
   Valid? Return status 200 (OK), and the output as the payload.

Serving Content as per the HTTP Protocol

1. Listen on port 80
2. The HTTP request arrives
“GET http://www.cse.yorku.ca/~roumani/jba/ase/se.cgi?hrss=er”
start a process to handle it (“fork”)
3. Extract the path/file from the URL: “~roumani/jba/ase/se.cgi?hrss=er”
4. Check whether file exists (yes, it does)
5. Check whether file is reachable & readable (yes, it is)
6. The content type is dynamic – the file is se.cgi
7. Masquerade as file owner.
8. Check that file is executable by owner (yes, it is)
9. Run the file and capture its output.
10. Check the validity of the output – it is valid html, so return status 200 (OK), and the output as the payload.

Programmatically fetching content, with all safe guards

L22App1

Programmatically fetching content, without safe guards

L22App1b

L22App1b2

Fetching Content Repeatedly, with a Pause

- L22App1c

Scraping Content

- L22App2

Scraping Content Repeatedly

- L22App3