Intro to Web App Development

CS 4640
Programming Languages for Web Applications

[Robert W. Sebesta, “Programming the World Wide Web
Jon Duckett, Interactive Frontend Web Development]
Web and Internet

A set of standards or infrastructures for distributing information to the world

Web
- HTML, CSS, Browser

Internet
- Application layer: Specifies the shared communication protocols → over network: DNS, FTP, HTTP, POP, SSH, Telnet, ...
- Transport layer: Provides host-to-host communication services: TCP, ...
- Internet layer: Transports packets from host across network boundaries: IP, ICMP, ...
- Link layer: Operates on the link that a host is physically connected to: MAC (Ethernet, DSL, ...)
Hypertext and the WWW

1945: Vannevar Bush proposes hypertext
1965: Ted Nelson coins the term “Hypertext” – beyond the linear constraints of text
1969: ARPANET comes online
1980: Tim Berners-Lee writes ENQUIRE, a notebook program allowing links to be made between nodes with titles
1989: Tim Berners-Lee’s Information Management proposal became WWW
1990: HTML defined
1992: CERN (Switzerland) releases WWW
1993: First browser: NCSA Mosaic
1994: First widely used commercial browser: Netscape
1997: More than 31,000,000 pages
2000: More than 100,000,000 hosts, more back-end programming than front-end hypertext
2004: 3,307,998,701 pages (Google)
2017: More than 46,200,000,000 pages (Google), more than 1,943,000,000 websites (www.internetlivestats.com), and large number of web app failures
Aspects to Consider

Software engineering aspect

• How can we design for change and reuse?
• Many developers may involve
  • What happens when a new developer joins the team?
  • How can a developer successfully maintain / change / refactor the code without understanding the whole system?

Usability aspect

• How can we design web apps that are usable for their intended purpose?
• A web app may serve millions of users with different needs
  • What happens when a new user interacts with the app?
  • How can we make a web app less frustrating to use?
URI: Uniform Resource Identifier

URI: `<scheme>://<domain><path>?<query>`

http://www.cs.virginia.edu/~up3f/cs4640/syllabus.html

Use HTTP scheme
(Other popular schemes: ftp, mailto, file)

Connect to cs.virginia.edu
May be host name or an IP address
Optional port number (e.g., :8080 for port 8080)
e.g., http://localhost:8080/myproject/register.php

Request resource
Stored in up3f/cs4640 folder

**URI, URL, and URN**

**URI**: Uniform Resource Identifier
Specify resource either by location or by name, or both

**URL**: Uniform Resource Locator
Specify resource by location

**URN**: Uniform Resource Name
Specify resource by name

http://www.domain.com:1234/path/to/resource?p1=v1&p2=v2
DNS: Domain Name System

Hierarchical Domain Name System for class Internet, organized into zones, each served by a name server

Mapping from names to IP addresses

[ref: https://en.wikipedia.org/wiki/Domain_Name_System]
HTTP: HyperText Transfer Protocol

GET request
Other popular types:
POST, PUT, DELETE, HEAD

www.cs.virginia.edu/~up3f/cs4640/syllabus.html

HTTP Request
GET /~up3f/cs4640/syllabus.html HTTP/1.1
Host: cs.virginia.edu
Accept: text/html
...

HTTP Response
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
...

<html>
<head><title>CS 4640</title></head>
<body>...

HTML returned content
Common MIME types:
application/json
application/pdf
image/png

Protocol and versions
Resource
Web server
Response status codes:
1xx Informational
2xx Success
3xx Redirection
4xx Client error
5xx Server error

Request line
Request header
Blank line then request message body

Response line
Response header
Blank line then response message body

Request header
Request message body

Resource
Protocol and versions

HTML document

Request line
Request header
Blank line then request message body

Response line
Response header
Blank line then response message body

Web server

HTTP/1.1
200_OK
Content-Type: text/html; charset=UTF-8

<html>
<head><title>CS 4640</title></head>
<body>...
Properties of HTTP

• Request-response
  • Interactions always initiated by client request to server
  • Server responds with results

• Stateless
  • Each request-response cycle independent from other
  • Any state information (login credentials, shopping carts, exam scores, ...) needs to be maintained somehow
Client-side vs. Server-side

HTTP Request
GET /~up3f/cs4640/syllabus.html HTTP/1.1
Host: cs.virginia.edu
Accept: text/html
...

HTTP Response
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
...
<html>
<head><title>CS 4640</title></head>
<body>...
General Web Terminology

• **Web page**: Data that fits in one browser screen
  • **Static**: HTML exists as a file on a computer
  • **Dynamic**: Created as needed

• **Web site**: A collection of connected web pages

• **Web application**: A program that is deployed on the web
  • User interface (UI) is in HTML
  • User interacts through HTTP’s request / response cycle
Static Web Pages

- URL corresponds to directory location on server
- Server responds to HTTP request by returning requested files or documents

- Advantages
  - Simple

- Disadvantages
  - No interactivity
Dynamic Web Pages

• Server responds to HTTP request by running a program that processes the request and produces the response

• Different content is displayed each time the web page is viewed

• Two types of dynamic web page
  • Client-side scripting
  • Server-side scripting
Client-Side Scripting

- Generate HTML on the client through scripts
- Example: JavaScript

```html
<html>
<head>
<title>Example: JavaScript to create a table of factorials</title>
</head>
<body>
<script type="text/javascript">
    document.write("<h2>Table of Factorials</h2>");
    for (i = 1, fact = 1; i < 10; i++, fact *= i) {
        document.write(i + "! = " + fact);
        document.write("<br>");
    }
</script>
</body>
</html>
```

- Advantages
  - Interactivity, input validation, customization, improving usability

- Disadvantages
  - Browser compatibility
Server-Side Scripting

- Generate HTML on the server through scripts
- Early approaches emphasized embedding server code inside HTML pages
- Examples: PHP, JSP

```php
<!doctype html>
<html>
<head>
    <title>Login example</title>
</head>
<body>
    You logged in as <font color="green">b><i><?php echo $_POST["name"]}}</b></font><br />
    with password <font color="green">b><i><?php echo $_POST["pwd"]</i></b></font>
</body>
</html>
```

```jsp
<html>
<head>
    <title>Counting with a JSP</title>
</head>
<body>
    <!-- Set global information for the page -->
    <%! page language="java" %>
    <!-- Declare the variable -->
    <! int count = 0; %>
    <!-- Scriptlet - Java code -->
    <! %
    for (int i = 0; i < 10; i++)
    {
        count = count+1;
    %>
    The counter value is: <%= count %>
    <! %>
</body>
</html>
```
The Web Today

• Increasingly reliance

• Modern web applications are
  • Distributed (world-wide)
  • Heterogeneous (hardware and software)
  • Highly user interactive
  • Built on new technology
  • Evolve from one architecture style to another, combine multiple styles
    • Newer architectural styles are not always better – more complex and may be overkill for simple sites

• The software is
  • Very loosely coupled
  • Written in multiple languages
  • Often generated dynamically
Important Quality Attributes

1. Reliability
2. Usability
3. Security

Customers have little “site loyalty” and will switch quickly, thus time to market is much less important than in other application areas.

(but still important!)

4. Availability
5. Scalability
6. Maintainability
7. Performance & Time to market
8. ...
Summary

• Web sites and web apps are now too complicated for *individuals* to manage.

• They need to be *engineered* by teams of people with diverse talents:
  • Programming skills
  • Graphics design
  • Usability
  • Information layout and engineering
  • Data communications
  • Database

We need web site engineering
What’s Next

• Architectural styles and web software models
• How to partition a system
• How components identify and communicate with each other
• How information is communicated
• How elements of a system can evolve