Internet Programming

Week 6

Instructor:
Daniel Slack, P. Eng.
Applied Computer Science
University of Winnipeg

Web Services

Chapter 16

Introduction

- Despite prevalence of internet usage across the world, web site usability has lagged behind desktop applications
  - Often because of relative immaturity compared to standard desktop patterns
- Any significant interaction results in waiting
  - Due to communication delay over the internet

Rich Internet Applications (RIA)

- RIA refers to web-based applications that simulate a desktop-style feel
  - Look, feel and usability
- Traditionally refers to enhancements to browser
  - Flash, Silverlight, Java Applets, etc
- With browsers now implementing newer specifications, plugins not necessary

RIA and Ajax

- The primary reason for this is Ajax
  - Asynchronous Javascript and XML
- We can use client side scripting to make web applications more responsive
- Allows us to separate client-side user interaction and server communication
  - Run in parallel
  - Frees up the Client-side UI thread for processing
  - Reduces server load

Implementation

- Different ways to implement Ajax style functionality
  - Iframe
  - XMLHttpRequest Object
- Primary way is XMLHttpRequest
  - Javascript object sends asynchronous requests to server
  - We update the DOM when result returns
- Low Level API
  - Fine for simple apps
  - More complicated, not really suitable for larger applications

XMLHttpRequest Object

- Also known as XHR
- Layer between the client and the server that manages asynchronous requests
- Supported by all browsers*
  - Well.. IE10+, IE7+ for basic stuff
- Initialized by the following:

```
var xhr = new XMLHttpRequest();
```

XMLHttpRequest Object

- Open method used to set up the request
  - Two mandatory parameters
    - `method`
      - Specifies the type of Request (GET/POST/etc)
    - `url`
      - Specifies the address to the server that will generate a response
      - Could be a script, servlet, html file, text file, etc.
    - `Optional Parameter`
      - Specifies whether the request is asynchronous
        - Defaults to true

XMLHttpRequest Object

- The send method is used to initiate the request
  - *sends* a request to the server
- Has an optional parameter
  - `data`
    - Specifies the data to be POSTed to the server
    - Set to null by default

XMLHttpRequest Object

- `onreadystatechange`
  - Stores the callback function (event handler) that gets called when the server responds
  - Where response data would be typically handled and DOM manipulated in asynchronous calls

XMLHttpRequest Object

- `readystate`
  - Keeps track of requests progress
  - Usually called from callback function to determine when the code should actually be executed.
    - 0: Signifies that the request is uninitialized
    - 1: Signifies that the request is loading
    - 2: Signifies that the request had been loaded
    - 3: Signifies that data is actively being sent from the server
    - 4: Signifies that the request has been completed

XMLHttpRequest Object

- `status`
  - HTTP status code of the request
    - 200: Request was successful
    - 404: Means that the requested resources was not found
    - 500: There was an error while the server was processing the request
  - Typically, 2XX status codes are considered “good”
  - 4XX typically resource related (missing, etc) and are “bad”
  - 5XX are errors from the server processing itself are are “bad”

JSON

XML vs JSON

- Extensible Markup Language (XML)
- XML was hyped a lot
- A data format that was:
  - human readable
  - machine parseable
- XML was the standard when XMLHttpRequest object was created
  - Hence the name

XML vs JSON

- JavaScript Object Notation (JSON)
- Coined by Douglas Crockford
- JSON has gained in popularity over XML
- Quickly becoming the format of choice for HTML5 apps
- Advantages:
  - Human readable
  - Parsed easily and quickly into JavaScript values and objects

JSON

- Using JSON involves two simple method calls
- To exchange or store an object in JSON:
  1. Call the JSON.stringify method
     - Pass the object as the argument

```
JSON.stringify(myObject);
```

  2. The result is a string that represents the object
     - We can store string, pass it to a function, or send it over the network

JSON

- To convert a JSON string to an object
  1. Call the JSON.parse method
     - Pass the string as the argument

```
JSON.parse(jsonString);
```

  2. The result is a copy of the original object

Example 01

- Using JSON parse/stringify

Example 2

- Load data from server and parse

Accessing the Server

- We have tested our code locally
  - *i*.e. we have served the data from a server running on our development machine
- Goal:
  - Retrieve live data from a server
  - Can we do it?

Browser Security Policy

- Answer: No
- The browser enforce security around the XMLRequest object
- Policy
  - Can’t retrieve data from a domain different from the domain the page was served from

Browser Security Policy

- Solution 1:
  - Copy web application files to the same server that is hosting the data file
  - Developers typically have access to the servers that will host the pages
  - Like Example 2
- Solution 2:
  - Use JSONP to get the data

JSONP

- XMLHttpRequest is a great way to get data into an app when the data is hosted on the same domain as the web app
- What about getting data from a third party?
  - e.g. Google or Twitter?
    - Must find a way around this security problem

JSONP

- Stands for JSON with Padding
- JSONP is a way of getting the script tag to do the work of retrieving the data

Example 03

- JSONP demonstration

JSONP

1. The source for the script is the URL of a web service
   - Service supplies us with JSON data
2. Browser encounters script element in the page
   - Sends HTTP request to the src URL
3. Server sends back JSON, BUT …
   - Before the server sends back the JSON string, it first wraps it in a function call
     - Like a call to updateSales
4. The JSON is parsed and interpreted
   - Then, it is wrapped in a function call
   - This function is then called with object created above

Callback Function

- Is approach useful?
  - How does web service know to call updateSales?
  - What if another function is needed?
- Web services require you to specify what you want the function to be named
  - Look at Example 3 server code

http://localhost:8080/?callback=updateSales

JSONP Security

- Is JSONP a security problem?
  - It is not any more or less secure than using <script> to load JavaScript
- JSONP requests to a malicious server could include malicious JavaScript
  - But it is no different than linking to libraries hosted on other servers
- Either way, be sure you trust the service
- Do not use this approach if your web app is handling sensitive data

JSONP and the DOM

- The user need to hit refresh to update the page!
  - The point of this route was to avoid hitting refresh
- Solution: Use the DOM
- You can use the DOM with the script element too
  - *i*.e. You can create a new script element in the DOM any time you want to retrieve more data

Example 04

- Name/age web service without refresh