Test Review

- Review returned tests
Web Service Communication

To obtain information from the City of Winston-Salem about *open work orders*, we will request information from their exposed *web-service*.
XML Parsing: Key Ideas

• Tags:
  Start: <tag>
  End: </tag>

• Content:
  <tag>content</tag>

• Nesting:
  <tag1>
    <tag2>content</tag2>
  </tag1>

  Close most recently opened tag!

• Attributes:
  <tag label="value">

• Auto-closing tag:
  <tag label="value" /> 

  Equivalent to no-content
  <tag label="value"></tag>

A Simpler XML Example

<meals>
  <meal>
    <id>1</id>
    <main>Hamburger</main>
    <side>French fries</side>
    <drink>Coke</drink>
    <price>3.99</price>
  </meal>
  <meal>
    <id>2</id>
    <main>Nuggets</main>
    <side>Onion rings</side>
    <drink>Sweet tea</drink>
    <price>4.39</price>
  </meal>
</meals>
Alternative Parsing Methods

• We don’t have to hand-create a parser

• There already exist in Java XML parsers
  – DOM: Document Object Model
    • Parse document to build model of whole document, allow queries against model
  – SAX: Simple API for XML
    • Trigger events on-fly as encounter XML elements
    • We will ignore this
A DOM Parser reads an entire XML document and builds a tree-structure out of it. We then ask to view parts of the tree.

A Node is an entity in the tree. An Element is a component of the XML document which represents all data associated with a node (essentially, corresponds to a sub-tree starting from a node).
DOM Parser: Meals Example

<MEALS>

<MEAL>
  <ID>1</ID>
  <MAIN>Ham burger</MAIN>
  <SIDE>French Fries</SIDE>
  <DRINK>Coke</DRINK>
  <PRICE>3.99</PRICE>

<MEAL>
  <ID>2</ID>
  <MAIN>Nuggets</MAIN>
  <SIDE>Onion Rings</SIDE>
  <DRINK>Sweet Tea</DRINK>
  <PRICE>4.39</PRICE>
DOM Parser

We will request for all elements with a particular tag ("meal")

- Returns us a node list of "meal" elements
  - There are 2 in our example
- For each meal, we will ask for elements with five tags of interest ("id", "main", "side", "drink", "price")
  - Still get a list (though of size 1 here)
  - Have to ask for the child of these elements to get the actual text data
DOM-Based XML Parser

• See example in Eclipse/on website
Hand Parsing vs. DOM

• Why might hand-parsing be useful?
  – Can often be more efficient in execution
  – Requires less library overhead

• Why might DOM be useful?
  – More general (applicable to other situations)
  – Works out-of-the-box
    • Less up-front-time coding
  – Built-in error handling
Web Services

Web Service Server

Function call w/ parameters

Return value

Web Service Client

Exposes an API of functionality it can provide

API is very much like the Java or Android API: a description of functions, return types, and parameter types

(can be shared via a WSDL [Web Service Description Language] file – which is in XML and machine-parseable)

Employs a generic web service client that allows indication of which webservice to employ, which function to call, and parameters to send

“Let me call function X on webservice Y passing parameters A and B”
Web Services

I, mathserver, can do:

- int add(int, int)
- int subtract(int, int)
- int multiply(int, int)
- int divide(int, int)

I would like to do:
- function multiply
- parameters 3, 2
- on mathserver

Notion of: Remote Procedure Call
Calling a procedure (function/method) processed on another computer
Web Services

Web service:
- Reads XML function call request
- Extracts parameters
- Does actual computation
- Generates a return XML string

Web service client:
- Takes desired function call specification
- Generates appropriate XML to represent function call and data
- Passes to web service server
- Parses XML return string for answer
Web Services

Notion of: **SOAP (Simple Object Access Protocol)**
Specifies, among other things, how to encode function call in XML, including how to represent (potentially large) datatypes.

This XML is hidden by Java function calls

```xml
<function name="multiply">
  <param><type>int</type><value>2</value></param>
  <param><type>int</type><value>2</value></param>
</function>

<result><type>int</type><value>6</value></result>

This we have to parse
SOAP Elements

- SOAP Encoding
- Envelope package
- Header/Body pattern
  - Similar to how HTTP works
Simple Example

We will ignore the header!

c = Add(n1, n2)
SOAP Request in XML

```xml
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <t:transId xmlns:t="http://a.com/trans">345</t:transId>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <m:Add xmlns:m="http://a.com/Calculator">
      <n1>3</n1>
      <n2>4</n2>
    </m:Add>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
City Web Service

- City of Winston-Salem provides a web service to request information they have decided to make publicly available
- There is one function exposed: `Request`
- Takes 14 parameters, of which 7 need to be set by us with data

```javascript
g2ws.Request("BuildingPermits", "Apno, OwnerName, JobAddress, PropertyUse, Status", "OwnerName", "0", ",", ",", ",", ",", ",", ",", ",");
```

<table>
<thead>
<tr>
<th>Dataset name Required</th>
<th>Columns to return Empty string for all columns</th>
<th>Order By column</th>
<th>Number of rows to return &quot;0&quot; or &quot;&quot; for all rows</th>
</tr>
</thead>
</table>
datasetName

"WORKORDERS"

columns

The available columns are listed below. Passing an empty string will return all columns.

columns examples: ("") or ("WORK_NO, INITIATED, ASSET").

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK_NO</td>
<td>CHAR(9)</td>
<td>Work order number</td>
</tr>
<tr>
<td>INITIATED</td>
<td>DATE</td>
<td>Date Work Order was created</td>
</tr>
<tr>
<td>ADDRESS</td>
<td>CHAR(36)</td>
<td>Street address where work will be done</td>
</tr>
<tr>
<td>CITY</td>
<td>CHAR(20)</td>
<td>City where work will be done</td>
</tr>
<tr>
<td>ZIP</td>
<td>CHAR(5)</td>
<td>Zip code where work will be done</td>
</tr>
<tr>
<td>ASSET</td>
<td>CHAR(30)</td>
<td>Sewer Main, Storm Inlet, Sewer Meter, Water Backflow, etc.</td>
</tr>
<tr>
<td>ACTIVITY</td>
<td>CHAR(30)</td>
<td>BACKFLOW TEST, INLET MANHOLE INSPECT, WATER MAIN FLUSH, etc.</td>
</tr>
<tr>
<td>MAINTENANCE_TYPE</td>
<td>CHAR(30)</td>
<td>MONTHLY MAIN FLUSHING, CORRECTIVE, PREVENTIVE, INSPECTION, etc.</td>
</tr>
<tr>
<td>AUTHORIZED</td>
<td>CHAR(6)</td>
<td>Person who approved the work order</td>
</tr>
<tr>
<td>PROBLEM</td>
<td>CHAR(30)</td>
<td>SERVICE STOPPED, WATER CONNECTION, POTHOLE, etc.</td>
</tr>
<tr>
<td>PRIORITY</td>
<td>CHAR(4)</td>
<td>EMER, HIGH, NORM</td>
</tr>
<tr>
<td>COMPLETED_FLAG</td>
<td>CHAR(1)</td>
<td>Whether work completed (Y,N)</td>
</tr>
<tr>
<td>ASSIGNED_TO</td>
<td>CHAR(34)</td>
<td>Person responsible for work</td>
</tr>
<tr>
<td>CREW</td>
<td>CHAR(9)</td>
<td>Crew assigned to work (crew number)</td>
</tr>
<tr>
<td>XCOORD</td>
<td>NUMBER</td>
<td>GPS Longitude</td>
</tr>
<tr>
<td>YCOORD</td>
<td>NUMBER</td>
<td>GPS Latitude</td>
</tr>
</tbody>
</table>
City Web Service

**orderBy**

The result set can be ordered by any of the columns listed above. By default when an `orderBy` column is passed the resultset is returned in ASCENDING order. To order in DESCENDING order append `DESC` to the end of your `orderBy` string.

`orderBy` examples: (""), ("WORK_NO"), ("WORK_NO, ADDRESS"), or ("ASSET, ADDRESS DESC")

**numRows**

When used in conjunction with the `orderBy` parameter, providing a value for `numRows` will limit the number of rows returned. When a value of "0" or "" is passed then all rows are returned. The maximum number of rows returned for the "WORKORDERS" dataset is

`numRows` Examples: (""), ("0"), or ("1000")

**Dataset Specific Parameters**

The following information is specific to the "WORKORDERS" data request. All ten parameters are string datatypes and must be passed even if empty strings.

**param1**

`param1` (xCoord/Longitude)
Example: "-80.2567894"

**param2**

`param2` (yCoord/Latitude)
Example: "36.7997894"

**param3**

`param3` (Radius) Value is in feet and the maximum value is 10560 (2 miles)
Example: "1000"

**param4**

`param4` (Not Used)
This parameter is not used by the "WORKORDERS" view. Pass an empty string (""")
City Web Service


- Need a web services client library: KSOAP2
  Acts as our function calling proxy
  Allows us to generate SOAP envelope, SOAP request, and “mail” the envelope & request
  Generates the XML for us
KSOAP2

• KSOAP2
  – Comes in a JAR file you can add to your project
    • I will upload the JAR file to our tumblr website
    • Import using:
      – Your Project -> right click -> Import -> File System -> yourjar.jar
      – Your Project -> right click -> Properties -> Java Build Path -> Libraries -> Add Jar -> yourjar.jar
City WebService Client

• I have built a Java class which abstracts away the details of dealing with KSOAP2
• Employ as is, or tweak as necessary
• See code in Eclipse

• Usage:
  
  CityWebServiceClient webServiceClient = new CityWebServiceClient();
  String queryResponse = webServiceClient.getWorkOrdersXML(latitude,longitude,radiusInFeet);
  // parse the query response “by-hand” or using DOM parser