

Homework: TV Listings Google Maps mashup – an Ajax/JSON Exercise

1. Objectives

- Become familiar with the Ajax, JSON & XML technologies;
- Use a combination of HTML, CSS, DOM, XMLHttpRequest, XML and Java Servlets;
- Provide a Google Maps-based interface to display the current TV Listing of Los Angeles TV stations.

2. Background

2.1 AJAX & JSON

Ajax (Asynchronous JavaScript + XML) incorporates several technologies:

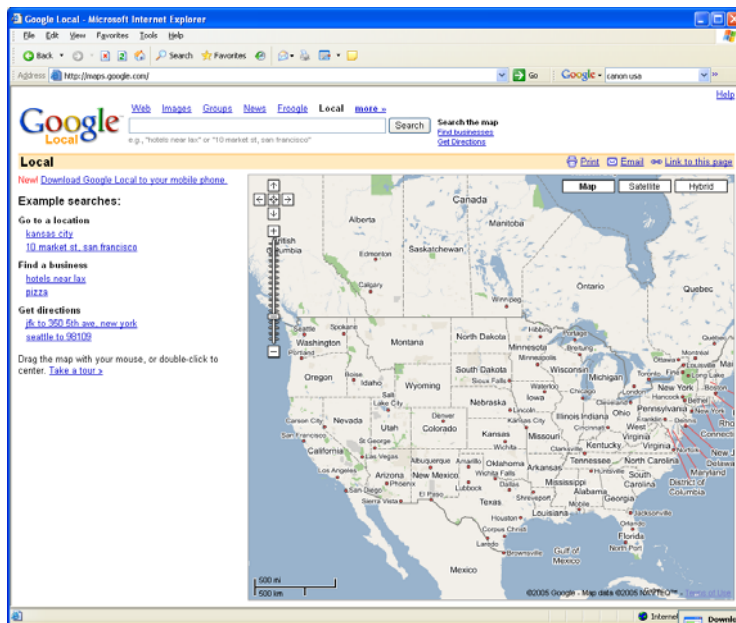
- Standards-based presentation using XHTML and CSS;
- Dynamic display and interaction using the Document Object Model (DOM);
- Data interchange and manipulation using XML and XSLT;
- Asynchronous data retrieval using XMLHttpRequest;
- JavaScript binding everything together.

See the class slides at <http://www-scf.usc.edu/~csci571/Slides/ajax.ppt>.

JSON, short for JavaScript Object Notation, is a lightweight data interchange format. Its main application in Ajax web application programming, where it serves as an alternative to the use of the XML format for data exchange between client and server. See the class slides at <http://www-scf.usc.edu/~csci571/Slides/JSON1.ppt>.

2.2 Google Maps

Google Maps (a.k.a. Google Local) is a technology from Google that provides access to street maps, satellite maps and a combination of the two (called “hybrid” maps). The Google Local homepage looks as follows:



The Google Maps homepage is available at:

<http://maps.google.com/>

The Google Maps API is a technology that lets developers embed Google Maps in their own web pages with JavaScript. You can add overlays to the map (including markers and polylines) and display shadowed "info windows" just like Google Maps. The Google Maps API homepage is available at:

<http://www.google.com/apis/maps/>

Note that as of April 3, 2006, Google has released version 2 of the Google Maps API, with additional support for Geocoding (translating an address into latitude and longitude coordinates). You will need to use this version of the API, as the input data will be provided as addresses and not as coordinates. Version 2 of the API is documented at:

<http://www.google.com/apis/maps/documentation/>

3. Description of the Exercise

To implement a Google Maps interface to the TV Listings that you developed in Homework #6, you are required to write a combination of HTML, JavaScript and Java Servlet programs. The top-level interface consists of three areas:

- A Form area including a “Download Listings” button;
- A Map of the United States, implemented using the Google Maps API, and indicating, using Map Overlays (also known as “markers”), the location of 8 TV stations local to the Los Angeles area TV market: KCBS, KNBC, KTLA, KCET, KABC, KCAL, KTTV and CNN. The overlays are “clickable”, and when clicked

provide information about the TV station in an “info” window (a “bubble”), including:

- a. TV station name
- b. TV station address
- c. TV station phone number
- d. Current TV listing (time slot, show name and link) [showing the “complete schedule” from HW #6]

Figure 1 below shows the complete interface window, including the details of an info window after TV listing data has been retrieved and a marker has been clicked.

Important Notes:

- a. The markers should use an appropriate custom PNG bitmap. Small bitmaps for the logos of KCBS, KNBC, KTLA, KCET, KABC, KCAL, KTTV and CNN are available at:

<http://www-scf.usc.edu/~csci571/2008Fall/hw8/kcbs.png>

<http://www-scf.usc.edu/~csci571/2008Fall/hw8/knbc.png>

<http://www-scf.usc.edu/~csci571/2008Fall/hw8/ktla.png>

<http://www-scf.usc.edu/~csci571/2008Fall/hw8/kcet.png>

<http://www-scf.usc.edu/~csci571/2008Fall/hw8/kabc.png>

<http://www-scf.usc.edu/~csci571/2008Fall/hw8/kcal.png>

<http://www-scf.usc.edu/~csci571/2008Fall/hw8/kttv.png>

<http://www-scf.usc.edu/~csci571/2008Fall/hw8/cnn.png>

- b. The map should be initially centered on the “Los Angeles, CA” area;
- c. The call names, network name, addresses, phone numbers and link to logos of the local TV stations are stored in a XML file located at:

<http://www-scf.usc.edu/~csci571/2008Fall/hw8/tvstations.xml>

- d. Note that the map contains the Zooming and Panning controls (on the top left) and the default 3 map type buttons (map, satellite, hybrid or terrain, on the top right). Your application should handle these controls appropriately;
- e. Your program should work on both IE and Firefox.

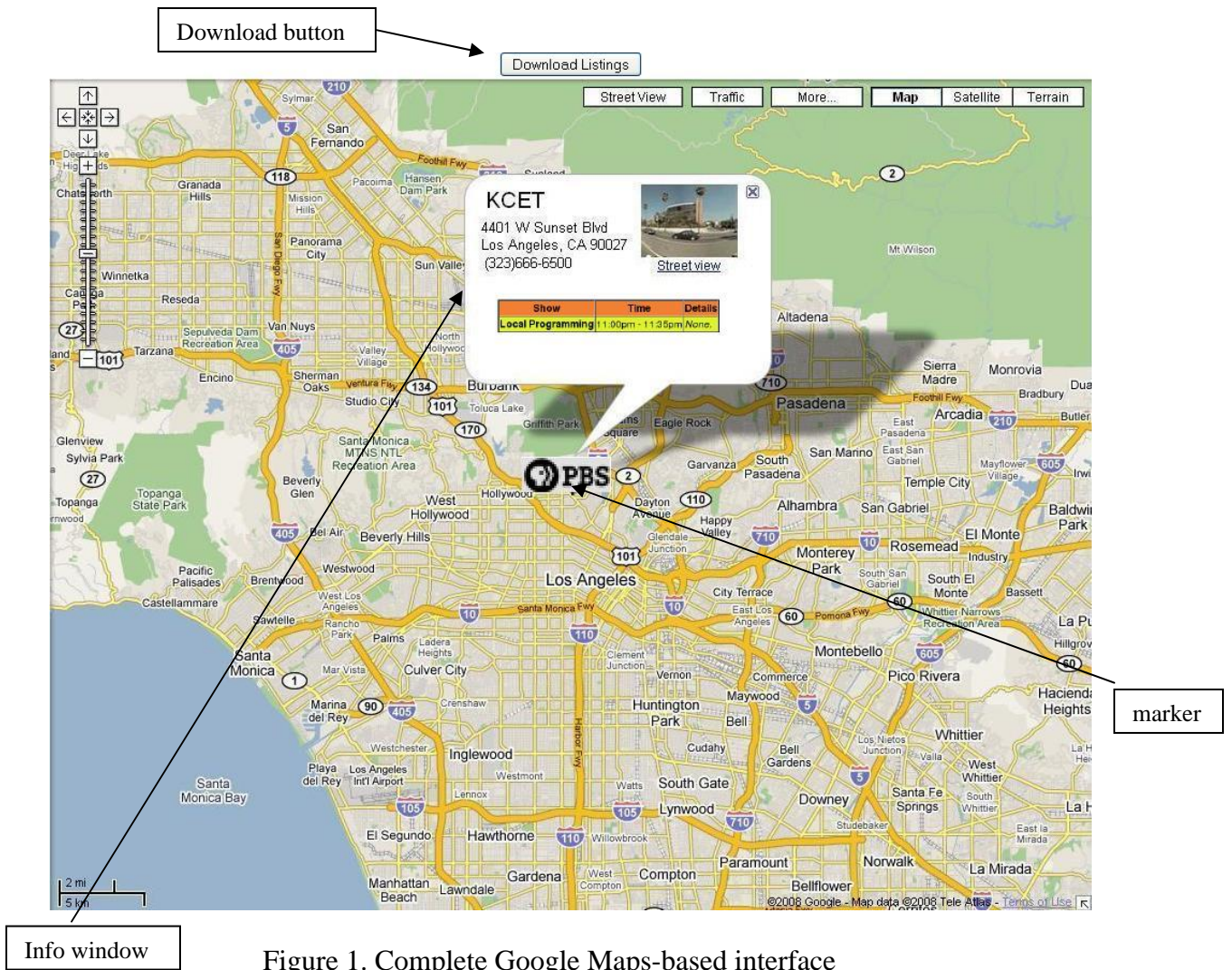


Figure 1. Complete Google Maps-based interface

You should implement the page shown in Figure 1 using HTML, JavaScript and the Google Maps API.

Once the Google Maps page is displayed (initially with no markers) and the “Download Listings” button is pressed, the form would call a JavaScript function that executes an XMLHttpRequest to start an asynchronous transaction with a Java Servlet running under Tomcat. When using the Google Maps API, you can either use XMLHttpRequest or GXmlHttp (which shields you from the code to detect the browser), as described in the API documents at:

<http://code.google.com/apis/maps/documentation/index.html>

and

<http://code.google.com/apis/maps/documentation/reference.html#GXmlHttp>

The Java Servlet in turn retrieves the current TV listings data from <http://tv.yahoo.com/listings> by initiating a connection with your Apache server and executing a CGI program at a location similar to this one:

http://csci571.usc.edu:YourApachePort/cgi-bin/hw6_listings.pl

and then retrieving the XML file generated on your student account, identical to the file that you created in Exercise #6, using a URL such as:

<http://csci571.usc.edu/~YourStudentId/listings.xml>

The listings.xml file contains the same XML data as described in Homework #6, as shown in the partial sample below:

```
<?xml version="1.0" encoding="UTF-8"?>
<TVLISTINGSRESULTS>
  <CHANNELS>
    <CHANNEL>
      <NAME>ABC</NAME>
      <SHOW>
        <TIMESLOT>3:00pm -4:00pm</TIMESLOT>
        <LINK>http://tv.yahoo.com/listings/general-hospital/show/86
        </LINK>
        <SHOWNAME>General Hospital</SHOWNAME>
      </SHOW>
      <SHOW>
        <TIMESLOT>4:00pm -6:30pm</TIMESLOT>
        <LINK>N/A</LINK>
        <SHOWNAME>Local Programming</SHOWNAME>
      </SHOW>
    </CHANNEL>
    <CHANNEL>
      . . . .
    </CHANNEL>
  </CHANNELS>
</TVLISTINGSRESULTS>
```

The Java Servlet then extracts from this file only the listing information for the 8 local stations (KCBS, KNBC, KTLA, KCET, KABC, KCAL, KTTV and CNN) and returns the listings information in a JSON file that is obtained asynchronously through XMLHttpRequest.

The format of the JSON file is as follows:

```
{feed:
  {title: "TV Listings",
  link: "http://csci571.usc.edu:8080/feed.xml",
  channels: [
    {name: "ABC",
    shows: [
      {timeslot: "3:00pm -4:00pm",
      link: "http://tv.yahoo.com/listings/general-hospital/show/86",
```

```

        showname: "General Hospital"},
    {timeslot: "4:00pm -6:30pm",
    link: "N/A",
    showname: "Local Programming"}
  ]},
  {name:"CBS",
  shows: [
    {timeslot: "3:00pm -4:00pm",
    link:"http://tv.yahoo.com/listings/guiding-light/show/30848",
    showname: "Guiding Light"},
    {timeslot: "4:00pm -6:30pm",
    link: "N/A",
    showname: "Local Programming"}
  ]},
  {name:"FOX",
  shows: [
    {timeslot: "N/A",
    showname: "Local Programming"}
  ]}
]}
}

```

After obtaining the query results, the JavaScript program places the overlays at the appropriate locations on the Los Angeles map (after translating the TV station address from the `tvstations.xml` file into latitude and longitude values, using the Google Maps API version 2 Geocoder), for each one of the returned “markers”. The bitmaps for the markers are also retrieved from the `tvstations.xml` file which has the following format:

```

<?xml version="1.0" encoding="UTF-8"?>
<STATIONS>
  <STATION>
    <NAME>KCBS</NAME>
    <NETWORK>CBS</NETWORK>
    <ADDRESS>4200 Radford Avenue</ADDRESS>
    <CITY>Studio City</CITY>
    <STATE>CA</STATE>
    <ZIP>91604</ZIP>
    <PHONE>(818)655-2000</PHONE>
    <IMAGE>http://www-scf.usc.edu/~csci571/2008Fall/hw8/kcbs.png</IMAGE>
  </STATION>
  .
  .
  .
  </STATION>
  .
  .
  .
</STATIONS>

```

Notice that the `<NETWORK>` tag in the `tvstations.xml` file corresponds to the “channel name”, as returned by `tv.yahoo.com`. The map should be “centered” to downtown LA.

The markers are implemented as “balloons” which, when clicked, bring up a custom “info window”, displaying the TV station “call” name, address, phone number and current TV listing, as shown in Figure 1.

4. Implementation Hints

1. *Step 1: Writing Your HTML program – Set up use of Google Maps API*

Your HTML should include the prologue required by Google Maps, including proper DOCTYPE, namespaces, style and a script that passes your Google maps API key, such as:

```
<!DOCTYPE html "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-
8"/>
<title>Google Maps JavaScript API Example</title>
<script
src="http://maps.google.com/maps?file=api&v=2&key=
ABQIAAAA6gpvPmzg5cB8NSnt79F-
xBTpzY8jYW9xcm5_L9s0QIAeskItExQQMQUtJYbbBL5rt4GrrTPHPTp22g "
type="text/javascript"></script>
<script type="text/javascript">

</script>
```

Important Note: You need to replace the key

```
ABQIAAAA6gpvPmzg5cB8NSnt79F-
xBTpzY8jYW9xcm5_L9s0QIAeskItExQQMQUtJYbbBL5rt4GrrTPHPTp22g
```

with your own Google Maps API key. See the API documentation at:

http://code.google.com/apis/maps/documentation/introduction.html#The_Hello_World_of_Google_Maps

2. Step 3: Writing your JavaScript Program – set up the base US Map

Use the Google Maps API to create a map of the United States and center it initially on the LA area, using the `GMap2` and `setCenter` properties, and display the map when the HTML loads, as in:

```
function onLoad() {
    var map = new GMap2(document.getElementById("map"));
    map.setCenter(new GLatLng(33.9300, -118.4000), 11);
    <other javascript code .. ..>

<body onload="onLoad()">
    <div id="map"></div>
    <div id="message"></div>
</body>
```

The above sample code centers the map on the LAX Airport. You should center your map to the Downtown LA area. You can use Google Earth to find latitude and longitude coordinates of any location in the Los Angeles area. Information on how to download and run Google Earth is available at:

<http://www.google.com/earth/>

3. Step 4: Writing your JavaScript Program – set up Google Ajax transaction

The JavaScript invoked by the Download Listings button click event should do all of the following:

- a. Assign the “callback” function;
- b. Assemble the `url` parameter of the GET as a reference to the Java Servlet to be invoked:

```
BASE_URL + "/servlet/ajax_maps"
```

- c. Call the `XMLHttpRequest` method (see Ajax Slide 31) and create the request object. Alternatively you can use the Google Maps version of `XMLHttpRequest`, called `GXMLHttpRequest`, that works on most browsers:

```
var req = GXMLHttpRequest.create();
```

- d. Prepare the GET `XMLHttpRequest` using the `setRequestHeader` method:

```
req.open("GET", url, true);  
req.onreadystatechange = myCallback;  
req.setRequestHeader("Connection", "Close");  
req.setRequestHeader("Method", "GET" + url + "HTTP/1.1");
```

4. Step 5: Writing your JavaScript Program – Execute Ajax Transaction

The JavaScript should finally invoke the `XMLHttpRequest` `send` method (see Ajax slide 31).

The “callback” function should check for completion of the transaction (request `readyState` equal to 4 and `status` equal to 200 (see AJAX slide 34 and JSON slide 5); use `eval()` and the `responseText` method to retrieve the resulting JSON data (see JSON slide 5), and use the appropriate Google Maps APIs to (a) use the Geocoder API to translate addresses into latitude and longitude coordinates and (b) place the markers on the US map (see `GMarker` object) and respond to marker “clicks” by displaying the “info windows” (see the marker property `openInfoWindowHtml`).

5. Step 6: Use the Java Servlet to respond to XMLHttpRequest and retrieve the TV listings

The Java Servlet referred above as `/servlet/ajax_maps` (see 3.c above) should be invoked using `doGet()`.

The Java Servlet should do all of the following:

- a. Initiating a connection with your Apache server and executing a CGI program to retrieve the TV listings from tv.yahoo.com;
- b. Wait until the CGI program finishes and creates the XML output file;
- c. Open an input data stream with the URL:

<http://csci571.usc.edu/~YourStudentId/listings.xml>

6. *Step 7: Use the Java Servlet to retrieve the XML file content*

You may have to use an XML parser (JAXP, for example). If you are hand coding using JAXP, the steps to retrieve the XML file content may be as follows:

Step 1: Get the XML content based on the URL above in section 5.c.

- You need to open a URL connection to get the file you want. To create a URL connection:

```
URL url = new URL(urlString);  
  
URLConnection urlConnection = url.openConnection();  
  
urlConnection.setAllowUserInteraction(false);  
  
InputStream urlStream = url.openStream();  
  
//read content
```

Step 2: Parse the XML file using an XML parser

- Any XML parser can be used to parse the XML file. You can use methods like `getNodeName()` to access these elements.

7. *Step 8: Use the Java Servlet to process the XML data*

As you parse the data, you will build an output string, converting the XML data into JSON format, as described in section 3.

Finally you will return the JSON as a single string to the calling JavaScript program.

The Java Servlet should handle exceptions such as `MalformedURLException` and `IOException`.

5. Prerequisites

This homework requires the use of the following components:

1. A servlet-based web server, Tomcat 4.1.27. Instructions on how to load Tomcat 4.1.27 can be found here: <http://www->

scf.usc.edu/~csci571/2006Spring/tomcatinstall.html. A tar version of Tomcat 4.1.27 can be found here: <http://www-scf.usc.edu/~csci571/download/jakarta-tomcat-4.1.27.tar>.

2. The Java Servlet library, which has functionality similar to Perl's LWP library, to perform HTTP transactions using methods such as `doGet()` or `doPost()` from Java.
3. A Java XML parser library. You may use the JDOM 1.0, an object model that uses XML parsers to build documents, available in the Download section of the class website. Additional information on JDOM is available at <http://www.jdom.org/>. You may also use JAXP, the Java API for XML Parsing, version 1.1, included in the Java JDK 1.4 (import `javax.xml.parsers.*`) and documented at: <http://java.sun.com/xml/jaxp/dist/1.1/docs/api/>. A good tutorial on JAXP is available at <http://www-106.ibm.com/developerworks/xml/library/x-jaxp1.html>.
4. You need to obtain a single Google Maps API key, which is valid for a single "directory" on your web server. For example, if your Tomcat port is 8834, you would sign up for the URL `http://csci571.usc.edu:8834/examples`, and the key you get will be good for all URLs in the `http://csci571.usc.edu:8834/examples/` directory and its subdirectories. You can sign up for your private Google Maps API key at <http://code.google.com/apis/maps/signup.html>.

6. Deployment Structure

To write your own Java Servlets program using Tomcat 4.1.27, you need to:

1. Successfully install Tomcat 4.1.27 on your machine.
2. Go to `$CATALINA_HOME/webapps/examples` directory.
3. Place the HTML, CSS and JavaScript (`.js`) files in the Tomcat `servlets` subdirectory.
4. Place your Java Servlets file (`.java`) in the `/WEB-INF/classes` folder. So the path of your Servlets file is http://server_name:port/examples/servlet/your_servlet_name
5. Add appropriate sections to the `WEB-INF/web.xml` file, as in:

```
<servlet>
  <servlet-name>ajax_maps</servlet-name>
  <display-name>AJAX Maps</display-name>
  <servlet-class>AJAXMaps</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>ajax_maps</servlet-name>
  <url-pattern>/servlet/ajax_maps</url-pattern>
</servlet-mapping>
```

6. To avoid `UTFDataFormatException` during file IO operation, you have to use JDK 1.3 or later for Tomcat. In the `.cshrc` file under your home directory, add the entries:

```
setenv JAVA_HOME /usr/j2se
```

```
setenv PATH /usr/j2se/bin:${PATH}
```

7. Before you issue a request to your Java Servlet file, you need to compile it. You might need a Java Servlet class to compile your code, so open the `.cshrc` file, and add `"/home/scf-22/csci571/servlet_classfiles/j2ee.jar"` to your `CLASSPATH` variable.
8. Then run `"source .cshrc"` and restart your Tomcat server.

7. Material You Need to Submit

On your course homework page, your link for this homework should go to a page that includes your JavaScript/HTML program (a page similar to the one depicted in the picture in section 3). You should submit all source code files including HTML (.html), Cascading Style Sheets (.CSS), JavaScript (.js), Java Servlets (.java) and a README file electronically to the csci571 account so that it can be graded and compared to all other students' code via the MOSS code comparison tool.

8. Extra Credit

Extra credit will be given any one of two items is implemented:

1. Google Maps street view. Search for "street view" in the demo gallery at:
<http://code.google.com/apis/maps/documentation/demogallery.html>
2. Local Stations listings. Instead of showing the listing for the "East coast" stations (the default in HW #6), show the listings for the 90089 ZIP code, Cable, "Time Warner – Hollywood/Wilshire – Digital". Implementing this item will require changes in your Perl program.

There will be NO HELP provided by the TAs and in the newsgroup for the optional "extra credit" items.