28

Designing Apps Using The WebView Control

Victor Matos
Cleveland State University

Notes are based on:

Android Developers
http://developer.android.com/index.html

The Busy Coder's Guide to Advanced Android Development
by Mark L. Murphy. Ed. CommonsWare.

WebView

An interesting class of Android applications could be created by combining **JavaScripted HTML** pages and the **WebView** control.

- The WebView widget is typically used by the WebKit browser engine to display pages accessed from the Internet.
- However, you could also display **local HTML pages** on a WebView.

The Android application interacts with the WebView through **user created objects** which are passed back and forth between the WebView and the Android Activities.

WebView

Architecture:

HTML & Javascript pages

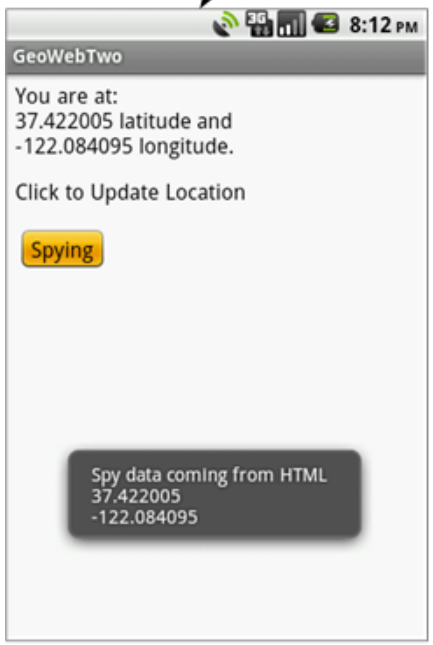
+

Android Activities

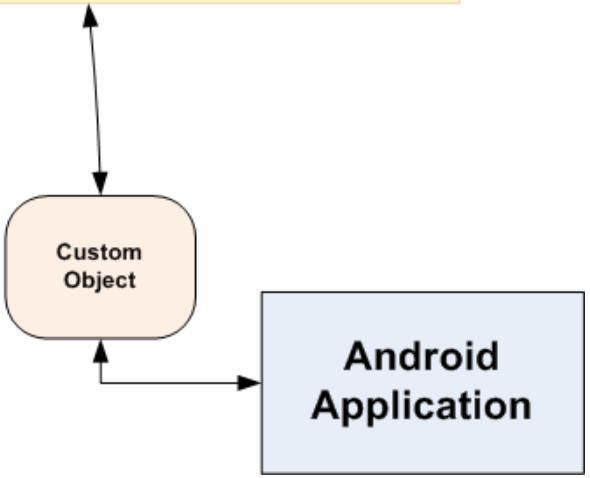
```
<head>
    <title>Android GeoWebTwo Demo</title>
    <script language="javascript">
            function whereami() {
                    ...
            }
            function whereami2() {
                    ...
            }
    </script>
</head>
    <body>
        <p>
        You are at: <br/>
        <span id="lat">(unknown)</span> latitude
        and <br/>
        <span id="lon">(unknown)</span> longitude.
        </p>
        <p><a onClick="whereami()">Click to Update
        Location</a></p>
        <p> <input type="button" onclick=
        "whereami2()" value="Spying">
        </p>
    </body>
</html>
```

Javascript uses custom object

UI designed using HTML



WebView



WebView

What is New?

The **addJavascriptInterface()** method on WebView allows you to pass a Java object from Android activities to the WebView, exposing its methods.

The various getters/setters methods defined in the object allow data exchange between the HTML-UI and Android activities.

In addition Javascript events (clicking buttons, making selections, filling boxes, etc) could be used to react to the user requests and correspondingly pass data from the UI to the Android classes.

WebView

Example: How is the HTML-Android exchange done?

In this example the current coordinates (latitude, longitude) of the device will be displayed on the screen. Assume:

1. The application's UI consists of a WebView called "browser", also in the **Assets** folder the developer has introduced a Javascripted web-page called **geoweb2.html**.

2. The Android application has defined a custom object called "locater" with get/set methods exposing the [lat, lon] values.

Cont.

WebView

Example: How is the HTML-Android exchange done?
(continuation)

The following statements are held in the Android main activity

```
browser.getSettings().setJavaScriptEnabled(true);
browser.addJavascriptInterface(new Locater(), "locater");
browser.loadUrl("file:///android_asset/geoweb2.html");
```

1. The first allows the use of Javascript on the WebView
2. The second statement passes the object type and name.
3. The last stat. loads the local HTML page on the WebView.

Cont.

WebView

Example: How does the HTML page uses the object ?
(continuation)

The HTML page could manage the “locater” object through its accessors such as in the following lines:

```
document.getElementById("lat").innerHTML=locater.getLatitude();
locater.setAddress (  document.getElementById(“address").innerHTML  );
Locater.doSomething();
```

Where “lat” (and “address”) are HTML placeholders defined using

```
<span id="lat"> (unknown) </span>
```

Cont.

WebView

Example: How does the HTML page uses the object ?
(continuation)

Consider the JavaScript expression:

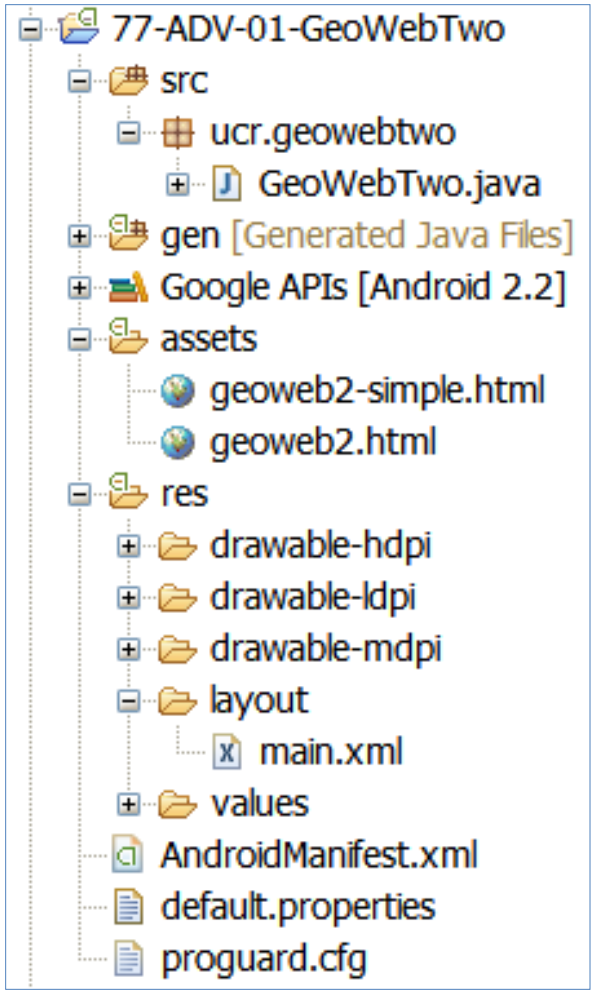
```
document.getElementById("lat").innerHTML
```

- The **innerHTML** property is used along with **getElementById** within your JavaScript code to refer to an HTML element and change its contents.
- **innerHTML** allows you to change the page's content without refreshing the page (*this makes your website feel quicker and more responsive to user input*).
- The innerHTML property is not actually part of the official DOM specification, despite this, it is supported in all major browsers including *Android's* ***WebKit***.

Cont.

WebView

Complete Example: Get Location – Show on a Local WebView



main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
>
<WebView
        android:id="@+id/browser"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
/>
</LinearLayout>
```

Add to the Manifest

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
```

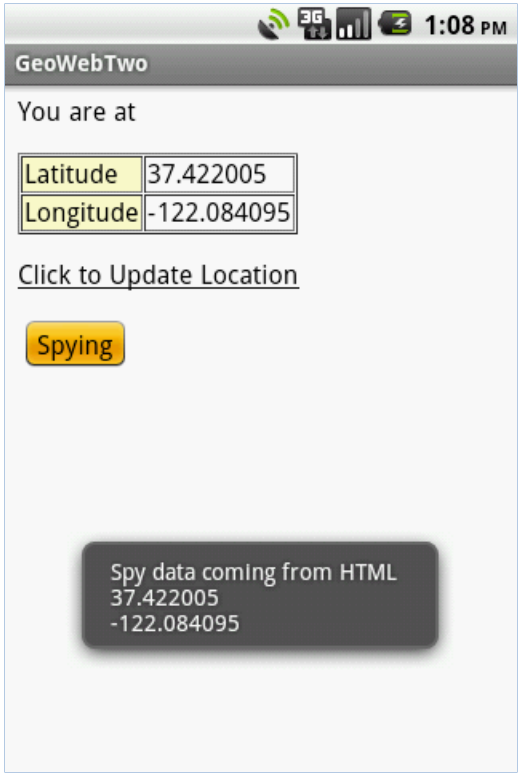

Cont.

WebView

Complete Example: Get Location – Show on a Local WebView

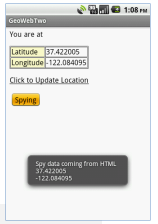
assets: geoweb2.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
        <title>Android GeoWebOne Demo</title>
        <script language="javascript">
        function whereami() {
          document.getElementById("lat").innerHTML=locater.getLatitude();
          document.getElementById("lon").innerHTML=locater.getLongitude();
          var spy = "Spy data coming from HTML\n"
                  + document.getElementById("lat").innerHTML
                  + "\n"
                  + document.getElementById("lon").innerHTML;
          locater.setValue(spy);
        }
        function whereami2() {
            var spy = "Spy data coming from HTML\n"
                        + document.getElementById("lat").innerHTML
                        + "\n"
                        + document.getElementById("lon").innerHTML;
          locater.htmlPassing2Android(spy);
        }
        </script>
</head>

<body>
        <p>You are at</p>
        <table  border="1" cellspacing="1" cellpadding="1">
          <tr>
            <td bgcolor="#FFFFCC">Latitude</td>
            <td><span id="lat">(unknown)</span></td>
          </tr>
          <tr>
            <td bgcolor="#FFFFCC">Longitude</td>
            <td><span id="lon">(unknown)</span></td>
          </tr>
        </table>
        <p><a onClick="whereami()"><u>Click to Update Location</u></a></p>
        <p> <input type="button" onclick= "whereami2()" value="Spying">
</body>
</html>
```



Cont.

WebView

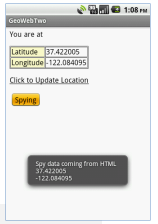


Complete Example: Get Location – Show on a Local WebView

```
package ucr.geowebtwo;
// code based on M. Murphy - CommonsWare, V. Matos

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.webkit.WebView;
import android.widget.Toast;

public class GeoWebTwo extends Activity {

private String PROVIDER = "gps";
private WebView browser;
private LocationManager myLocationManager = null;
Locater locater = new Locater();

@Override
public void onCreate(Bundle icicle) {
      super.onCreate(icicle);
      setContentView(R.layout.main);
      browser = (WebView) findViewById(R.id.browser);
      // request GPS location services
      myLocationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
      // enable JavaScript, pass user's object, load page
      browser.getSettings().setJavaScriptEnabled(true);
      browser.addJavascriptInterface(locater, "locater");
      browser.loadUrl("file:///android_asset/geoweb2.html");
}
```

Cont.

WebView

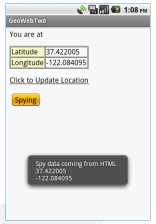


Complete Example: Get Location – Show on a Local WebView

```
@Override
public void onResume() {
     super.onResume();
     myLocationManager.requestLocationUpdates(
     PROVIDER, 3000, 10,onLocationChange);
}

@Override
public void onPause() {
     super.onPause();
     myLocationManager.removeUpdates(onLocationChange);
}
// ---------------------------------------------------------------------
LocationListener onLocationChange = new LocationListener() {
     // passing the actual values of lat & lon. Waiting for the function
     // whereami(...) to drop the arguments into HTML placeholders
     public void onLocationChanged(Location location) {
     StringBuilder buf = new StringBuilder("javascript:whereami(");
     buf.append(String.valueOf(location.getLatitude()));
     buf.append(",");
     buf.append(String.valueOf(location.getLongitude()));
     buf.append(")");
     browser.loadUrl(buf.toString());
}

public void onProviderDisabled(String provider) {
     // required for interface, not used
}
```

Cont.

WebView

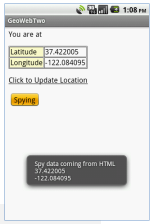


Complete Example: Get Location – Show on a Local WebView

```
  public void onProviderEnabled(String provider) {
     // required for interface, not used
  }

  public void onStatusChanged(String provider, int status, Bundle extras) {
     // required for interface, not used
  }
};
```

Cont.

WebView



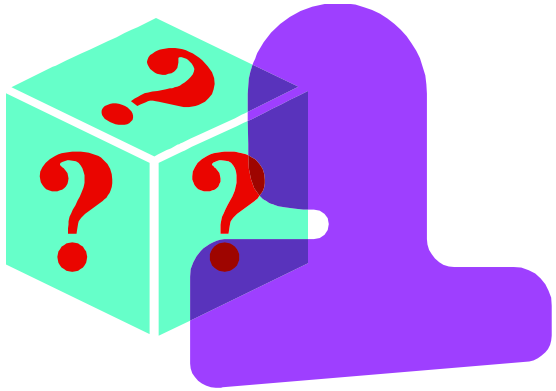
Complete Example: Get Location – Show on a Local WebView

```
// ////////////////////////////////////////////////////////////////
public class Locater {
          public String spy = "";

          public double getLatitude() {
          Location loc = myLocationManager.getLastKnownLocation(PROVIDER);
          if (loc == null) {
          return (0);
          }
          return (loc.getLatitude());
     }

     public double getLongitude() {
          Location loc = myLocationManager.getLastKnownLocation(PROVIDER);
          if (loc == null) {
          return (0);
          }
          return (loc.getLongitude());
     }

     public void htmlPassing2Android(String dataFromHtml) {
          // changes to the HTML place-holders lat & lon can be seen here.
          // There is an HTML button that when clicked calls this Android method.
          spy = dataFromHtml;
          Toast.makeText(getApplicationContext(), spy, 1).show();
          }
     } // Locater
} // GeoWebTwo
```

Last

WebView

Questions



Zipped code:

77-ADV-01-GeoWebTwo.zip