CS371m - Mobile Computing

WebView and Web Services
Using Built In Browser App

- To use the built in browser create an Intent and start the Activity

```java
public void showTop10(View v) {
    int day = datePicker.getDayOfMonth();
    int month = datePicker.getMonth() + 1;
    int year = datePicker.getYear();
    Log.d(TAG, "date: " + day + "\"\" + month + "\"\" + year);

    Intent i = new Intent(Intent.ACTION_VIEW,
                           Uri.parse("http://www.cbs.com/late_night" +
                                      "/late_show/top_ten/" +
                                      "top_ten_update_by_date.php?year=" + year +
                                      "&month=" + month + "&day=" + day));
    startActivity(i);
```
WebView

• A View that display web pages
  – basis for creating your own web browser
  – OR just display some online content inside of your Activity

• Uses WebKit rendering engine
WebView

- Android 4.4, API level 19 added an alternative to WebKit
- Chromium
- "Chromium WebView provides broad support for HTML5, CSS3, and JavaScript. It supports most of the HTML5 features available in Chrome for Android 30. It also brings an updated version of the JavaScript Engine (V8) that delivers dramatically improved JavaScript performance."

https://developer.android.com/about/versions/kitkat.html
WebView

• Built in functionality to:
• display page
• navigate forward and backwards through a history
• zoom in and out
• perform searches
• and more:
  – capture images of page, search page for string, deal with cookies on a per application basis,
More on WebView

- Scenarios for using WebView in app instead of built in browser:
  - provide info the app might need to update such as end user agreement or user guide (instead of doing app update)
    - display documents hosted online
  - OR application provides data that ALWAYS requires internet connect to retrieve data
    - as opposed to performing network request and parsing data to display in Android layout
- OR display ads (blah)
WebView Example

• Simple app to view and navigate web pages - demo WebView class
• res/layout/main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
    android:id="@+id/webview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
/>
```
WebView Activity

• override onCreate
• go to UT mobile site

```java
public class HelloWebView extends Activity {

    private WebView mWebView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        mWebView = (WebView) findViewById(R.id.webview);
        mWebView.setSettings().setJavaScriptEnabled(true);
        mWebView.loadUrl("http://m.utexas.edu");
    }
}
```
WebView Example

- Must add permission for app to use Internet
- Also change style so no title bar

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="scottm.examples"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="10" />

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

<application 
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name" > 
<activity 
    android:name=".HelloWebView"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.NoTitleBar" >
```
THINGS ON THE FRONT PAGE OF A UNIVERSITY WEBSITE

- Campus Photo Slideshow
- Alumni in the News
- Promotions for Campus Events
- Press Releases
- Statement of the School’s Philosophy
- Virtual Tour

THINGS PEOPLE GO TO THE SITE LOOKING FOR

- List of Faculty Phone Numbers and Emails
- Campus Address
- Application Forms
- Academic Calendar
- Campus Police Phone Number
- Department/Course Lists
- Parking Information
- Usable Campus Map

FULL NAME OF SCHOOL

- Letter from the President
Current Result

Clicking link actually leads to the default Android browser
Handling URL Requests

• To enable activity to handle its own URL requests create an inner class that extends WebViewClient

```java
private class HelloWebViewClient extends WebViewClient {
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        view.loadUrl(url);
        return true;
    }
}
```

• set client for mWebView

```java
mWebView.setWebViewClient(new HelloWebViewClient());
```
Navigating

• Making previous changes disables the back button
• Must override onKeyDown method
• Use WebView object to see if possible to go back

```java
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if ((keyCode == KeyEvent.KEYCODE_BACK)
        && mWebView.canGoBack()) {
        mWebView.goBack();
        return true;
    }
    return super.onKeyDown(keyCode, event);
}
```
Javascript

• If web page displayed in simple WebiView contains JavaScript ...

• enable JavaScript:

```java
WebView myWebView = (WebView) findViewById(R.id.webview);
WebSettings webSettings = myWebView.getSettings();
webSettings.setJavaScriptEnabled(true);
```

• create interface to act as bridge between JavaScript and Android
  – Example: convert JavaScript alert to Android Dialog
WEB SERVICES
Web Services

• "Web services are a means of exposing an API over a technology-neutral network endpoint."

• "They are a means to call a remote method or operation that's not tied to a specific platform or vendor and get a result."

  —Android in Action 3rd edition
# Web Services Sources


<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Maps</td>
<td>Mapping services</td>
<td>Mapping</td>
</tr>
<tr>
<td>Twitter</td>
<td>Microblogging service</td>
<td>Social</td>
</tr>
<tr>
<td>YouTube</td>
<td>Video sharing and search</td>
<td>Video</td>
</tr>
<tr>
<td>Flickr</td>
<td>Photo sharing service</td>
<td>Photos</td>
</tr>
<tr>
<td>Amazon eCommerce</td>
<td>Online retailer</td>
<td>Shopping</td>
</tr>
<tr>
<td>Facebook</td>
<td>Social networking service</td>
<td>Social</td>
</tr>
<tr>
<td>Twilio</td>
<td>Telephony service</td>
<td>Telephony</td>
</tr>
<tr>
<td>eBay</td>
<td>eBay Search service</td>
<td>Search</td>
</tr>
<tr>
<td>Last.fm</td>
<td>Online radio service</td>
<td>Music</td>
</tr>
<tr>
<td>Google Search</td>
<td>Search services</td>
<td>Search</td>
</tr>
<tr>
<td>Microsoft Bing Maps</td>
<td>Mapping services</td>
<td>Mapping</td>
</tr>
<tr>
<td>Twilio SMS</td>
<td>SMS messaging service</td>
<td>Messaging</td>
</tr>
<tr>
<td>del.icio.us</td>
<td>Social bookmarking</td>
<td>Bookmarks</td>
</tr>
<tr>
<td>Yahoo Search</td>
<td>Search services</td>
<td>Search</td>
</tr>
</tbody>
</table>
Example: Flickr API

- [http://www.flickr.com/services/api/](http://www.flickr.com/services/api/)
Flickr API Methods

- create url using API method name and parameters, plus api key
- 3 request formats
  - REST
  - XML-RPC
  - SOAP
- Multiple Response Formats
Flickr API Methods

flickr.photos.search

Return a list of photos matching some criteria. Only photos visible to the calling user will be returned. To return private or semi-private photos, the caller must be authenticated with 'read' permissions, and have permission to view the photos. Unauthenticated calls will only return public photos.

Authentication

This method does not require authentication.

Arguments

api_key (Required)
Your API application key. See here for more details.

user_id (Optional)
The NSID of the user who's photo to search. If this parameter isn't passed then everybody's public photos will be searched. A value of "me" will search against the calling user's photos for authenticated calls.

tags (Optional)
A comma-delimited list of tags. Photos with one or more of the tags listed will be returned. You can exclude results that match a term by prepending it with a - character.
Sample Request URL

- https://api.flickr.com/services/rest/?method=flickr.photos.search&api_key=754a89ad04e0b72f42fffb77f412c021e&tags=blue,cool,pretty
- https://api.flickr.com/services/rest/?method=flickr.photos.search&api_key=754a89ad04e0b72f42fffb77f412c021e&tags=blue,cool,pretty
Photo Source URLs

You can construct the source URL to a photo once you know its ID, server ID, farm ID and secret, as returned from the API.

The URL takes the following format:

http://farm{farm-id}.staticflickr.com/{server-id}/{id}_{secret}.jpg

or

http://farm{farm-id}.staticflickr.com/{server-id}/{id}_{secret}_{mstzb}.jpg

or

http://farm{farm-id}.staticflickr.com/{server-id}/{id}_{o-secret}_{o}.jpg

* Before November 18th, 2011 the API returned image URLs with hostnames like: "farm{farm-id}.static.flickr.com".

Size Suffixes

The letter suffixes are as follows:

s  small square 75x75
q  large square 150x150
t  thumbnail, 100 on longest side
m  small, 240 on longest side
Photo URL and Result

JSON Format

- Flickr allows request for response format
  – one example, JSON

```json
jsonFlickrApi(
  {"photos": {"page": 1, "pages": 99436, "perpage": 100,
    "total": "9943595", "photo": [{"id": "8987334232",
      "owner": "95960745@N05", "secret": "8ea7fa4884", "server": "3700",
      "farm": 4, "title": "ladder", "ispublic": 1, "isfriend": 0,
      "isfamily": 0}, {"id": "10922370846", "owner": "67877697@N07",
      "secret": "9ac9663673", "server": "3670", "farm": 4, "title": "Underneath
      U.F.O.s", "ispublic": 1, "isfriend": 0, "isfamily": 0},
      {"id": "10926178565", "owner": "34128007@N04", "secret": "f4212fe642",
      "server": "2875", "farm": 3, "title": "Los Angeles City Hall",
      "ispublic": 1, "isfriend": 0, "isfamily": 0}, {"id": "10926310046",
      "owner": "78618023@N04", "secret": "bc70fd4960", "server": "3678",
      "farm": 4, "title": "DSC_9235", "ispublic": 1, "isfriend": 0,
      "isfamily": 0}, {"id": "10926152545", "owner": "26870747@N04",
      "secret": "80ab7da855", "server": "7328", "farm": 8, "title": "Sunday
      Sunrise", "ispublic": 1, "isfriend": 0, "isfamily": 0},
      {"id": "10926228834", "owner": "69774579@N07", "secret": "aaf5eeced1",
      "server": "5533", "farm": 6, "title": "Production of light"},
  }
)}
```
WEATHER BUG EXAMPLE
WeatherBug Example

• From Deitel Android Programmers: An App-Driven Approach

• Example for Tablets
  – Fragments
  – tabbed navigation in Action Bar
  – Widget for home screen

• Our focus is on the use of Web Services
WeatherBug API

Build your own weather application using WeatherBug API tools

WeatherBug API lets novice to advanced developers create amazing weather applications using WeatherBug data.
Round Rock TX, 78681 United States

Temperature: 79°F
Feels like: 79°F
Humidity: 52%
Chance of Precipitation: 0%
WeatherView App - Five Day

Round Rock TX, 78681 United States

<table>
<thead>
<tr>
<th>Day</th>
<th>Weather Description</th>
<th>Highs</th>
<th>Lows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wednesday</td>
<td>Cloudy in the morning then becoming partly sunny. Highs in the upper 70s.</td>
<td>High: 81</td>
<td>Low: 60</td>
</tr>
<tr>
<td></td>
<td>South winds 5 to 10 mph.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thursday</td>
<td>Cloudy in the morning then becoming partly sunny. A 20 percent chance of showers and</td>
<td>High: 78</td>
<td>Low: 62</td>
</tr>
<tr>
<td></td>
<td>thunderstorms. Highs in the upper 70s.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>South winds 5 to 10 mph.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Friday</td>
<td>Cloudy with a 50 percent chance of showers and thunderstorms. Highs in the mid</td>
<td>High: 78</td>
<td>Low: 62</td>
</tr>
<tr>
<td></td>
<td>70s.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Saturday</td>
<td>Partly sunny with a 20 percent chance of showers and thunderstorms. Highs in the</td>
<td>High: 81</td>
<td>Low: 62</td>
</tr>
<tr>
<td></td>
<td>lower 60s.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sunday</td>
<td>Partly cloudy. Lows in the lower 60s. Highs in the lower 80s.</td>
<td>High: 82</td>
<td>Low: 62</td>
</tr>
</tbody>
</table>

Boston
Chicago
Dallas
Denver
New York
Round Rock
San Diego
San Francisco
Seattle
Use of API

• most API's require registration and a key value
• key used in requests

```java
// construct Weatherbug API URL
URL url = new URL(resources.getString(R.string.location_url_pre_zipcode) + zipcodeString + "&api_key=A5559065586");
```
WeatherBug Web Services

• Three classes deal with making requests via the WeatherBug API in WeatherView
  • ReadLocationTask
    – use zip to get location information
  • ReadForecastTask
    – read current forecast for given zip code
  • ReadFiveDayForecastTask
    – get forecast for next five days for given zip
Tasks

• All three class use AsyncTask
  – chose to extend
• constructors
• override doInBackground method
• override onPostExecute method
• define their own listeners
• Keep the UI thread responsive by using AsyncTask to perform potentially slow tasks
• "AsyncTask allows you to perform asynchronous work on your user interface. It performs the blocking operations in a worker thread and then publishes the results on the UI thread, without requiring you to handle threads and/or handlers yourself."

• Task started by invoking the execute method

ReadLocationTask

- Created with Context, zip code, and Listener
- Listener updated in postExecute method

```java
public ReadLocationTask(String zipCodeString, Context context, LocationLoadedListener listener) {
    this.zipcodeString = zipCodeString;
    this.context = context;
    this.resources = context.getResources();
    this.weatherLocationLoadedListener = listener;
}
```
ReadLocationTask - doInBackground

- Creates URL using zip code and API key

```java
// load city name in background thread
@override
protected String doInBackground(Object... params)
{
    try {
        // construct Weatherbug API URL
        URL url = new URL(resources.getString(
            R.string.location_url_pre_zipcode) + zipcodeString +
            "&api_key=A5559065586");

        Reader forecastReader = new InputStreamReader(url.openStream());
    }
```
JSON

• JavaScript Object Notation
• a way to represent JavaScript objects as Strings
• alternative to XML for passing data between servers and clients
• designed for data interchange format that humans can also read and write
JSON Format

• Built on two structures
  – collection of name-value pairs: a.k.a. objects, records, structs, etc.
  – an ordered list of values: a.k.a. an array

• objects
JSON Format

- arrays

- values
  - string, number, object, array, true, false, null
JSON Values

Syntax Diagrams for string and number: http://www.json.org/
JSON Strings

Any UNICODE character except " or \ or control character

- quotation mark
- reverse solidus
- solidus
- backspace
- formfeed
- newline
- carriage return
- horizontal tab
- 4 hexadecimal digits
JSON Numbers
JSON Examples

• value (String):
  – "Round Rock"

• array:
  – ["Round Rock", "Dallas", "Houston"]

• object
  – {"height":70,"weight":165}
Results For ReadLocationTask

• http://i.wxbug.net/REST/Direct/GetLocation.ashx?zip=78681&api_key=xxx
  – where xxxxxx is your API key

• Result:
Parsing JSON

- JsonReader class in Android API
- Read JSON encoded values as a stream of tokens
- ReadLocationTask uses a JsonReader to parse the JSON returned by the web request
- Pulls out city, state, and country string to display in View
Creating JsonReader

- and check given zip returns a valid location

```java
JsonReader forecastJsonReader = new JsonReader(forecastReader);
forecastJsonReader.beginObject(); // read the first Object

String name = forecastJsonReader.nextName();

// if the name indicates that the next item describes the
// zipcode's location
if (name.equals(resources.getString(R.string.location))) {
    forecastJsonReader.beginObject();
    String nextNameString;
```
while (forecastJsonReader.hasNext()) {
    nextNameString = forecastJsonReader.nextName();
    // if the name indicates that the next item describes the
    // zipcode's corresponding city name
    if (nextNameString.equals(
        resources.getString(R.string.city)))
        cityString = forecastJsonReader.nextString();
    else if (nextNameString.equals(resourcs.
        getString(R.string.state)))
        stateString = forecastJsonReader.nextString();
    else if (nextNameString.equals(resourcs.
        getString(R.string.country)))
        countryString = forecastJsonReader.nextString();
    else
        forecastJsonReader.skipValue();
}

forecastJsonReader.close();
onPostExecute

• Send the city, state, and country data to the listener

```java
// executed back on the UI thread after the city name loads
protected void onPostExecute(String nameString) {
    if (cityString != null)
        weatherLocationLoadedListener.onLocationLoaded(cityString, stateString, countryString);
    else {
        Toast errorToast = Toast.makeText(context, resources.getString(R.string.invalid_zipcode_error), Toast.LENGTH_LONG);
        errorToast.setGravity(Gravity.CENTER, 0, 0);
        errorToast.show();
    }
}
```
• Similar in nature to ReadLocationTask, but different url for different data

• 

```json
{"forecastHourlyList":

• and on for another 158 hours
```
ReadForecastTask

• Also downloads image for current condition

```java
// get the sky condition image Bitmap
public static Bitmap getIconBitmap(String conditionString,
    Resources resources, int bitmapSampleSize) {
    Bitmap iconBitmap = null;
    try {
        // create a URL pointing to the image on WeatherBug's site
        URL weatherURL = new URL(resources.getString(R.string.pre_condition_url) + conditionString +
            resources.getString(R.string.post_condition_url));

        Log.d(TAG, weatherURL.toString());

        BitmapFactory.Options options = new BitmapFactory.Options();
        if (bitmapSampleSize != -1)
            options.inSampleSize = bitmapSampleSize;

        // save the image as a Bitmap
        iconBitmap = BitmapFactory.decodeStream(weatherURL.openStream(), null, options);
    }
```
Icons Obtained From WeatherBug
ReadFiveDayForecastTask

• {
  "dateTime":1332892800000,
  "dayDesc":"Partly Cloudy","dayIcon":"cond003",
  "dayPred":"Cloudy in the morning...becoming partly cloudy. Patchy fog in the morning. Highs 61 to 66. Light winds becoming west 15 mph with gusts to 25 mph in the afternoon.",
  "dayTitle":"Wednesday","hasDay":true,
  "hasNight":true,"high":"66","hourly":null,"low":"54","nightDesc":"Drizzle","nightIcon":"cond162",
  "nightPred":"Partly cloudy in the evening...becoming cloudy. Patchy fog and patchy drizzle overnight. Lows 49 to 55. Areas of winds northwest 15 to 20 mph with gusts to 25 mph in the evening becoming light.",
  "nightTitle":"WednesdayNight","title":"Wednesday"},
Displaying Data

• App does not try and display all data, just chooses "most important"

• icon

• day of week

• day prediction

• high temp

• low temp