

# CS378 - Mobile Computing

Web - WebView and Web Services

# 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
  - <http://www.webkit.org/>



**The WebKit Open Source Project**

# WebView

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

# WebView Example

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

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

# WebView Activity

- override onCreate
- go to UT mobile site

```
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.getSettings().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 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" >

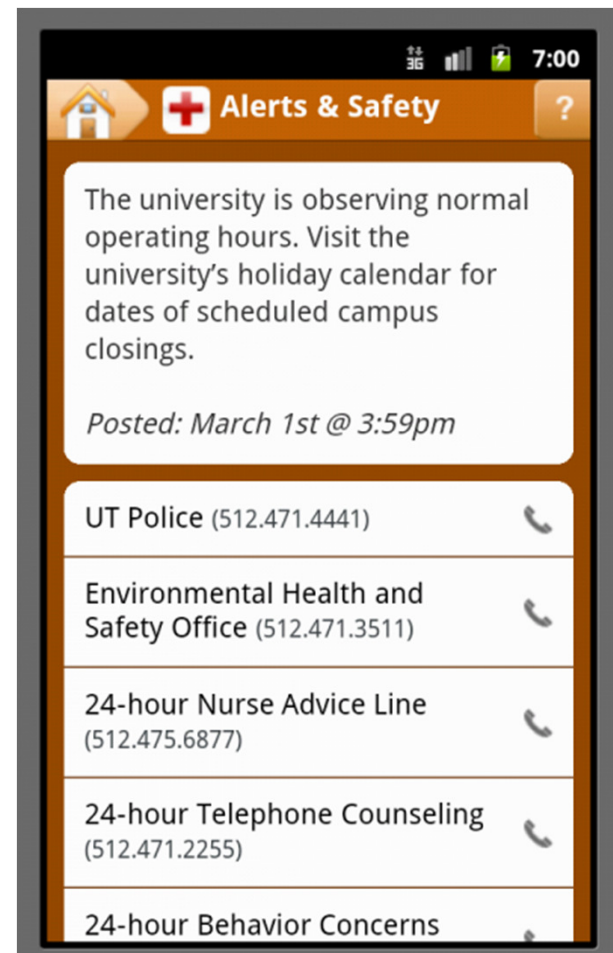
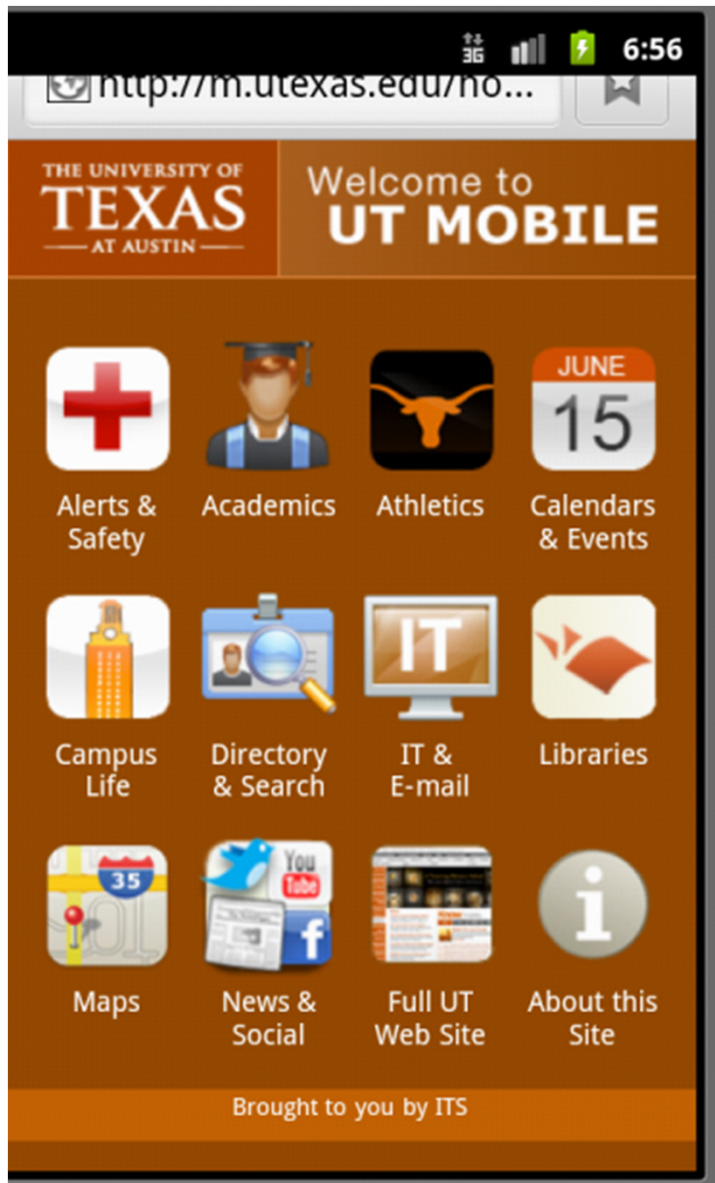
    <uses-sdk android:minSdkVersion="10" />

    <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" >
```

# 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

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

- set client for mWebView

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



# Navigating

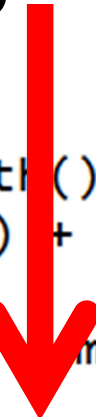
- Making previous changes disables the back button
- Must override onKeyDown method
- Use WebView object to see if possible to go back

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

# Using Built In Browser

- To simply use the built in browser create an Intent and start the Activity
- Like the Top Ten List App

```
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);  
}
```



# 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
- <http://developer.android.com/guide/webapps/webview.html>

# 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 3<sup>rd</sup> edition

# Web Services Sources

- <http://www.programmableweb.com/apis/directory/1?sort=mashups>

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

# WeatherBug Example

- From Deitel Android Programmers: An App-Driven Approach
- Example for Tablets
  - Fragments
  - tabbed navigation in Action Bar
  - Widget for home screen
- Are focus is on the use of Web Services

# WeatherBug API

A promotional graphic for the WeatherBug API. It features a blue header with the text 'WeatherBug API'. Below this is a light blue bar with 'USA'. The main body has a dark blue background with a white grid pattern and a bright blue sky with a sun. The text 'Build your own weather application using WeatherBug API tools' is prominently displayed. At the bottom, a light blue bar contains the text 'WeatherBug API lets novice to advanced developers create amazing weather applications using WeatherBug data.'



**WeatherBug API**

USA

**Build your own weather application  
using WeatherBug API tools**

WeatherBug API lets novice to advanced developers create  
amazing weather applications using WeatherBug data.

# WeatherView App - Current

 Weather Viewer    Current Conditions    Five Day Forecast     Add New City


Boston

Chicago

Dallas

Denver

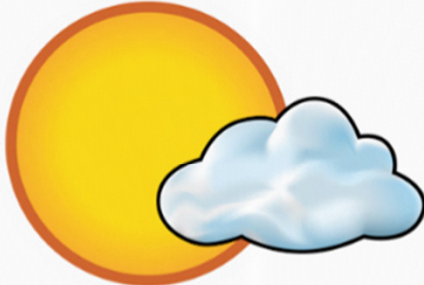
New York

Round Rock 

San Diego

San Francisco

Seattle



Round Rock TX, 78681 United States

Temperature:  
79°F

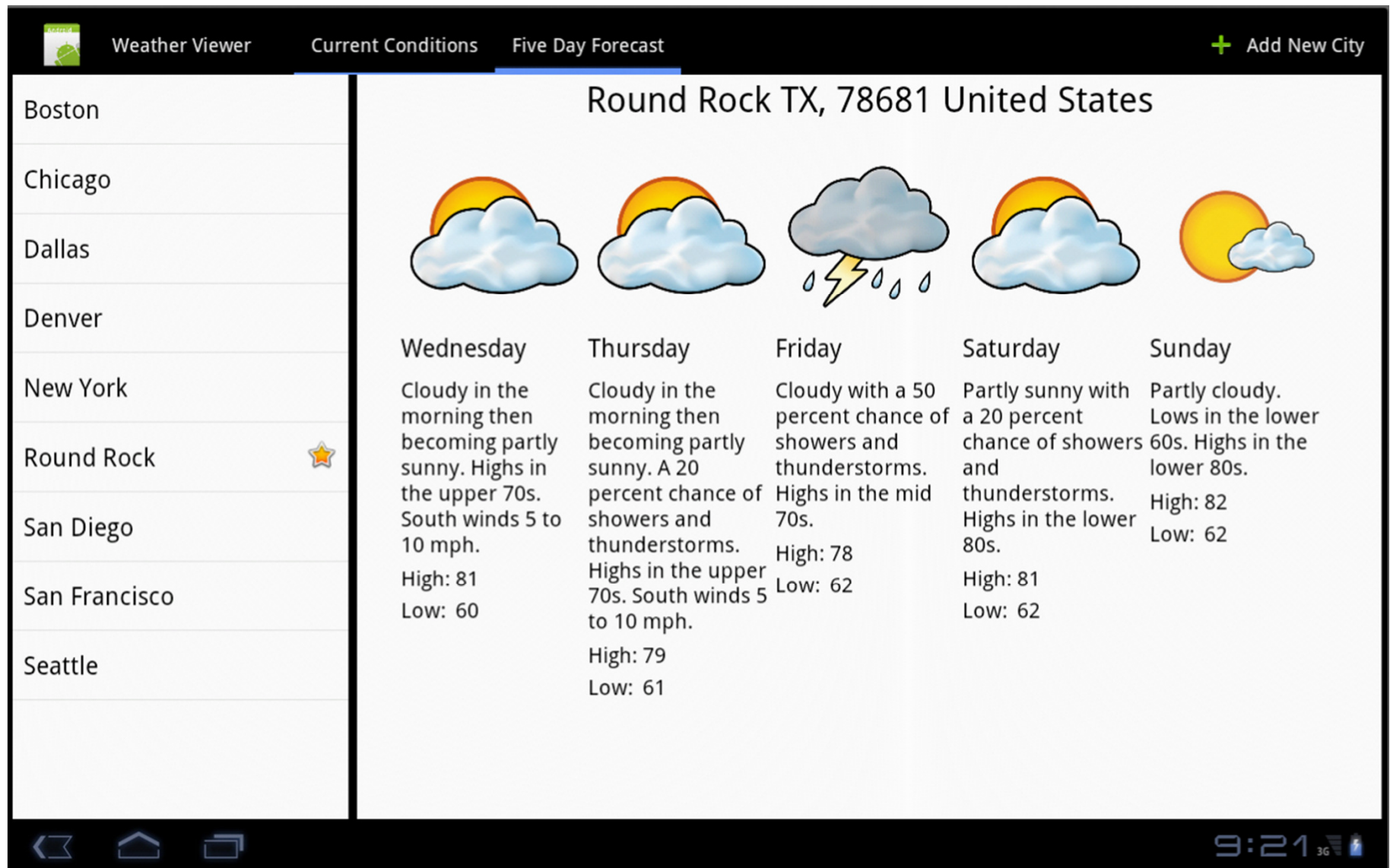
Feels like:  
79°F

Humidity:  
52%

Chance of Precipitation:  
0%



# WeatherView App - Five Day



# Use of API

- many API's require registration and a key value
- key used in requests

```
// 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
  - based on zip 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 extend AsyncTask
- 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

- "[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
- <http://developer.android.com/reference/android/os/AsyncTask.html>

# ReadLocationTask

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

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

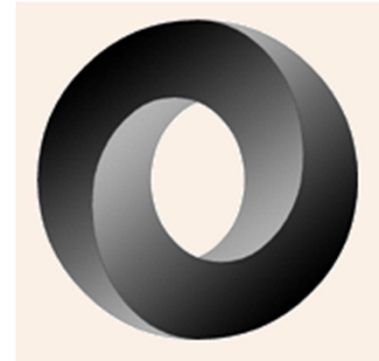
# ReadLocationTask - doInBackground

- Creates URL using zip and key fro API

```
// 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

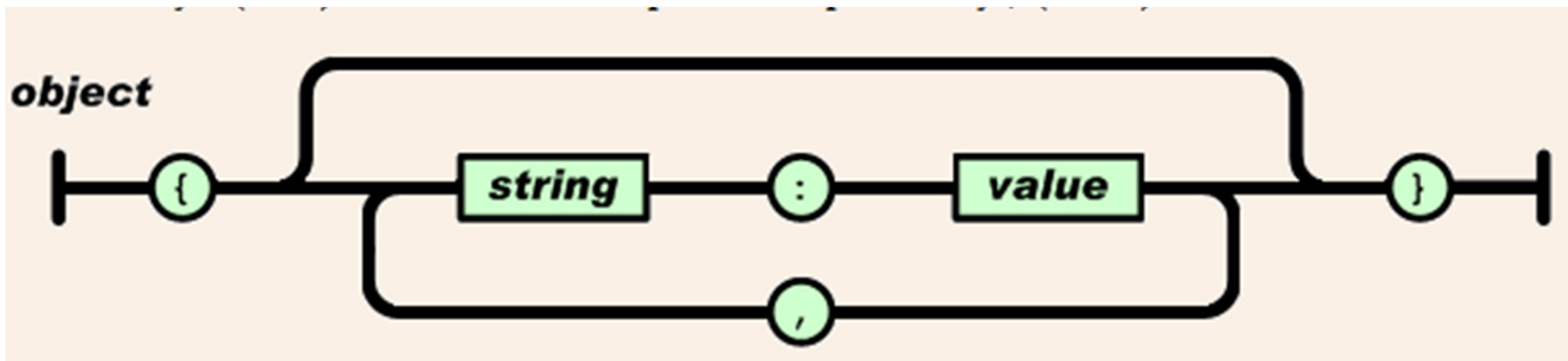


- JavaScript Object Notation
- a way to represent JavaScript objects as Strings
- alternative to XML for passing data between servers and clients
- design for data interchange format that humans can also read and write



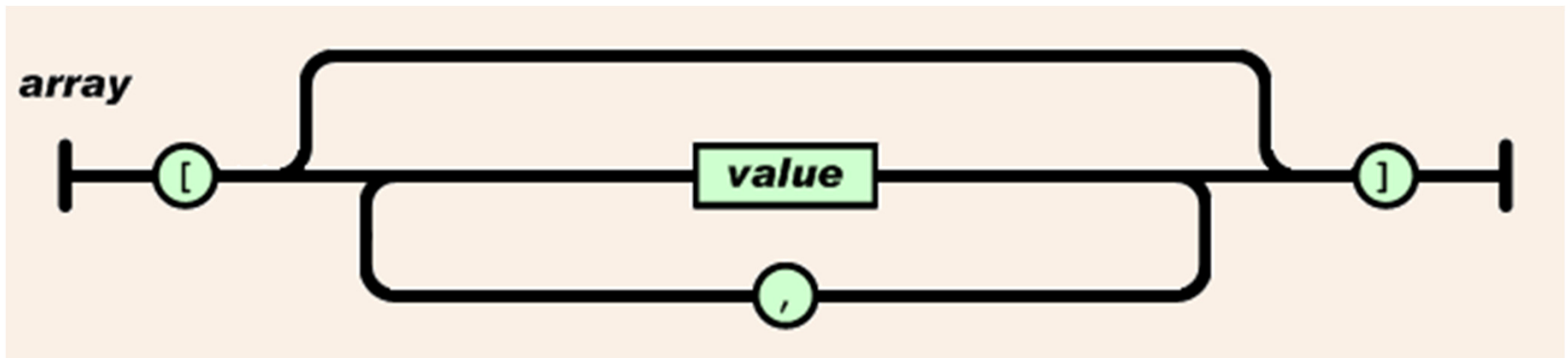
# JSON Format

- 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 Examples

- value:
  - "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=xxxxx](http://i.wxbug.net/REST/Direct/GetLocation.ashx?zip=78681&api_key=xxxxx)
  - where xxxxx is your API key
- Result:
  - {"location":{"city":"Round Rock","cityCode":null,"country":"United States","dma":"635","isUs":true,"lat":30.5123,"lon":-97.7117,"state":"TX","zipCode":"78681"}}

# Parsing JSON

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

# Creating JsonReader

- and checking it is a location

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

String name = forecastJsonReader洗洗洗Name();

// if the name indicates that the next item describes the
// zipcode's location
if (name.equals(resources.getString(R.string.Location))) {
    forecastJsonReader.beginObject();
    String洗洗洗洗洗String;
```

# Reading Location Data

```
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(resources.
        getString(R.string.state)))
        stateString = forecastJsonReader.nextString();
    else if ((nextNameString).equals(resources.
        getString(R.string.country)))
        countryString = forecastJsonReader.nextString();
    else
        forecastJsonReader.skipValue();
}

forecastJsonReader.close();
```

# onPostExecute

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

```
// 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();
    }
}
```



# ReadForecastTask

- Similar in nature to ReadLocationTask, but different url for different data
- {"forecastHourlyList":  
[{"chancePrecip":"10","dateTime":1332882000000,  
"desc":"PartlyCloudy","dewPoint":64,"feelsLike":73,  
"feelsLikeLabel":"Heat Index","humidity":"74",  
"icon":"cond002","skyCover":null,  
"temperature":73,"windDir":null,"windSpeed":10},  
{"chancePrecip":"10","dateTime":1332885600000,  
"desc":"Partly Cloudy","dewPoint":64,"feelsLike":70,  
"feelsLikeLabel":"Heat Index","humidity":"81",  
"icon":"cond002","skyCover":null,  
"temperature":70,"windDir":null,"windSpeed":11},  
and on for another 158 hours

# ReadForecastTask

- Also downloads image for current condition

```
// 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","ni  
ghtDesc":"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