RESTful Android

An Introduction to developing on Android with a focus on web-enabled applications

Roy van de Water

What is Android

- ”Android is a software stack for mobile devices that includes an operating system, middleware and key applications” - developer.android.com
- The ”Google” phone operating system
- Open Source
- Android takes care of the hardware on mobile devices so the developer can worry about the software

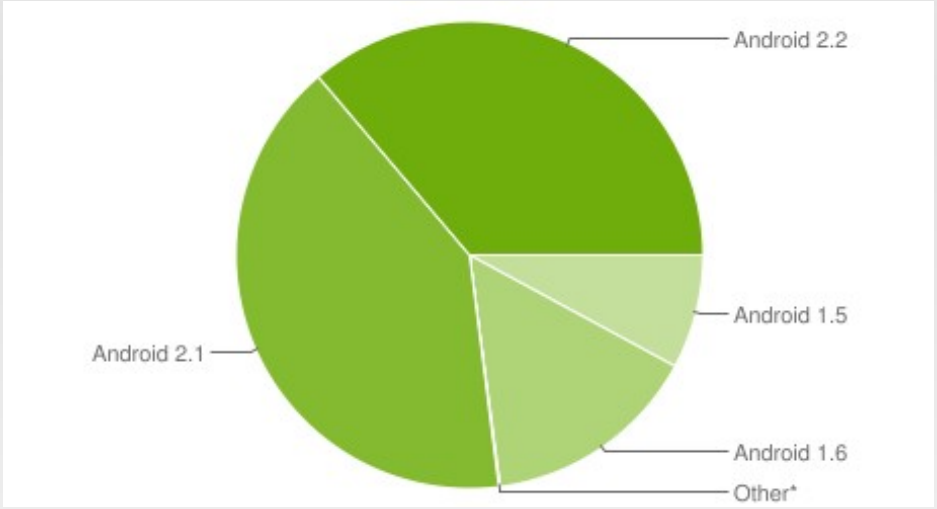
Why Android?

- Developing for Android is free
- Largest share of US Smart Phone Market
  - 43.6% of market in the US
- Development is in Java and XML

Statistic from: www.canalys.com/pr/2010/r2010111.html

Some Drawbacks

- Market Fragmentation
  - Currently split across 4 OS versions
- Manufacturer deviations
  - All major manufacturers (HTC, Motorola and Samsung) have their own Interfaces



As of November 1, 2010
http://developer.android.com/resources/dashboard/platform-versions.html

What do I need to start?

- The Android SDK (http://developer.android.com)
- Eclipse IDE (http://eclipse.org)
- Physical device is optional

Building your first app

- Android applications use a self-imposed MVC pattern
- The framework does not require that an MVC archicture is required, but it is highly recommended.

Hello World

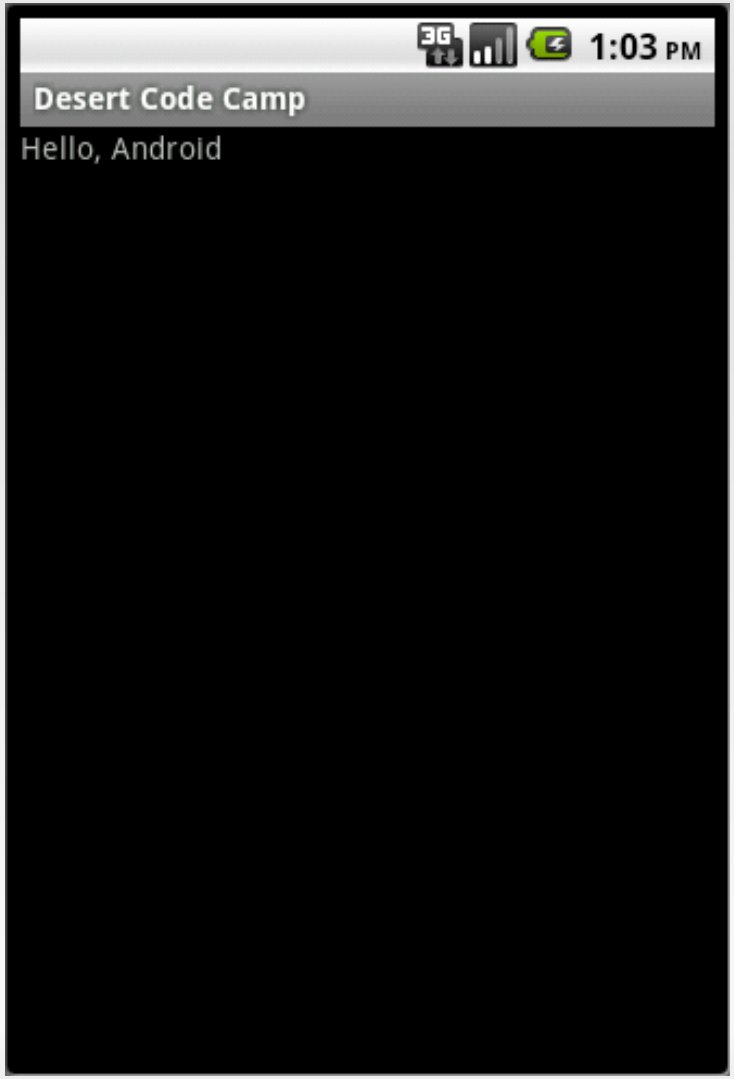
This is all the Java needed for a static app

```
1  package com.desertcodecamp.android.views;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5  import com.desertcodecamp.android.R;
6
7  public class HelloAndroid extends Activity
8  {
9      @Override
10     protected void onCreate(Bundle savedInstanceState)
11     {
12         super.onCreate(savedInstanceState);
13
14         setContentView(R.layout.main);
15     }
16 }
```

Add a touch of XML

```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" >
  <TextView
    android:text="Hello, Android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
</LinearLayout>
```

The result



Adding a Button

```
<?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" >
  <TextView
    android:text="Hello, Android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
  <Button
    android:id="@+id/refresh_button"
    android:text="Refresh"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
</LinearLayout>
```

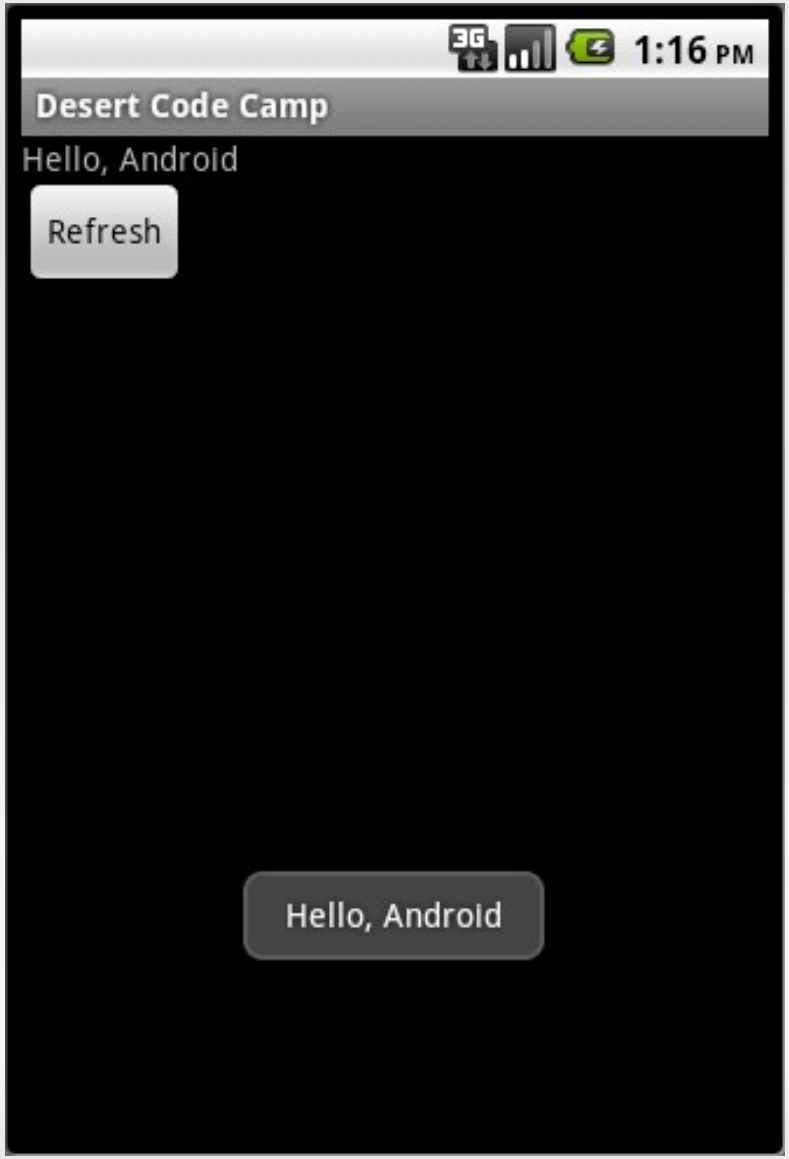
Make it do something

- R is an automatically generated file with references to XML objects
- Toasts are short messages that appear accross the screen.

```
Button refreshButton = (Button)findViewById(R.id.refresh_button);
refreshButton.setOnClickListener(new OnClickListener() {

    public void onClick(View v)
    {
        Toast.makeText(HelloAndroid.this, "Hello", Toast.LENGTH_LONG).show();
    }
});
```

A Toast


1:16 PM
Desert Code Camp
Hello, Android
Refresh
Hello, Android

Accessing the Web

- HttpClient is used to create an Http Request

```
1  private String loadSessions() {
2      HttpClient httpClient = new DefaultHttpClient();
3      HttpGet httpRequest = new HttpGet(URL);
4
5      HttpResponse response;
6      try {
7          response = httpClient.execute(httpRequest);
8      } catch (ClientProtocolException e) {
9          e.printStackTrace();
10     } catch (IOException e) {
11         e.printStackTrace();
12     }
13
14     InputStream responseStream = response.getEntity().getContent()
15     return streamToString(responseStream);
16 }
```

Put the M in MVC

- Seperate your business logic from the code that renders the views
- Model methods will be called from a background thread, so they will not have access to the UI
- Let our model parse and process our data

Parsing JSON

```
1  public class Session
2  {
3      private String name;
4      private String summary;
5
6      public Session(String jsonString) throws JSONException
7      {
8          JSONObject json = new JSONObject(jsonString);
9
10         this.name    = json.getString("Name");
11         this.summary = json.getString("Abstract");
12     }
13 }
```

Doing it in the Background

- While the HttpRequest is occuring, the UI thread is locked
- Instead
  - Spawn of background thread
  - Run HttpRequest and process data
  - Send message back to UI thread to update data

Using ASyncTask

- ASyncTask avoids the need to manage interthread communication.
- Create an anonymous instance of ASyncTask and override two functions:
  - doInBackground()
  - onPostExecute()

AsyncTask in action

```
1  AsyncTask<String,Integer,Bundle> asyncTask = new AsyncTask<String,Integer,Bundle>()
2  {
3
4      @Override
5      protected Bundle doInBackground(String... params)
6      {
7          return null;
8      }
9
10     @Override
11     protected void onPostExecute(Bundle result)
12     {
13         super.onPostExecute(result);
14     }
15
16 };
```

doInBackground()

- Everything in this function is done in the background
- Nothing within this function can touch the UI

```
@Override
protected Bundle doInBackground(String... params) {
    Bundle bundle = new Bundle();

    try {
        ArrayList<Session> sessions = Session.getAll();
        bundle.putSerializable(SESSIONS, sessions);
    } catch (UnsupportedEncodingException e) {
        Log.e(DesertCodeCampApplication.TAG, e.getStackTrace().toString());
    } catch (JSONException e) {
        Log.e(DesertCodeCampApplication.TAG, e.getStackTrace().toString());
    }

    return bundle;
}
```

OnPostExecute(Bundle result)

- Update the UI with the retrieved information

```
ArrayList<Session> sessions =
(ArrayList<Session>)result.getSerializable(SESSIONS);
```

Seperating the V and C

- The doInBackground() method performs the functions of a controller: ”receive input and initiates a response by making calls on model objects”
- The onPostExecute() method performs the function of a view: ”render the model into a form suitable for interaction”

Displaying the results

- Display a list of items with ListView

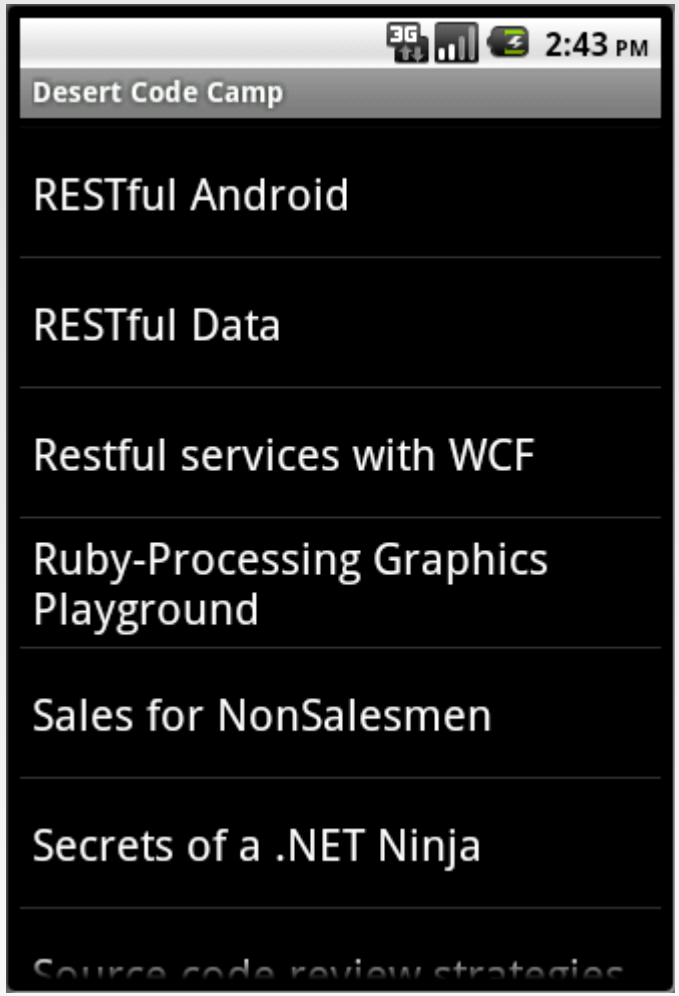
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent" >
  <ListView
    android:id="@+id/sessions_list"
    android:layout_width="fill_parent"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

Building a ListView

- ListViews made easy with the ArrayAdapter

```
1  @Override
2  protected void onPostExecute(Bundle result)
3  {
4      if (result.containsKey(SESSIONS)) {
5          ArrayList<Session> sessions = (ArrayList<Session>)result
6                                                  .getSerializable(SESSIONS);
7
8          ArrayAdapter<Session> adapter = new ArrayAdapter<Session>(Sessions.this,
9                  android.R.layout.simple_list_item_1, android.R.id.text1, sessions);
10
11
12         setListAdapter(adapter);
13
14         if(loadingDialog.isShowing())
15             loadingDialog.dismiss();
16     }
17 }
```

Questions


2:43 PM
Desert Code Camp
RESTful Android
RESTful Data
Restful services with WCF
Ruby-Processing Graphics Playground
Sales for NonSalesmen
Secrets of a .NET Ninja