HW#9 Clues

CSCI 571

Spring, 2015

HW#9 Prototype

- https://youtu.be/JQbVeDTdKfQ

Tutorials

- **1. Building your first App**

Creating a Project with Android Studio

http://developer.android.com/training/basics/firstapp/creating-project.html

- **2. Running your first App**

http://developer.android.com/training/basics/firstapp/running-app.html

(on same page see also “Run on the Emulator”)

- **3. Starting another activity**

http://developer.android.com/training/basics/firstapp/starting-activity.html

- **4. Comprehensive Tutorial / Article on getting started with Android**

http://www.vogella.com/tutorials/Android/article.html

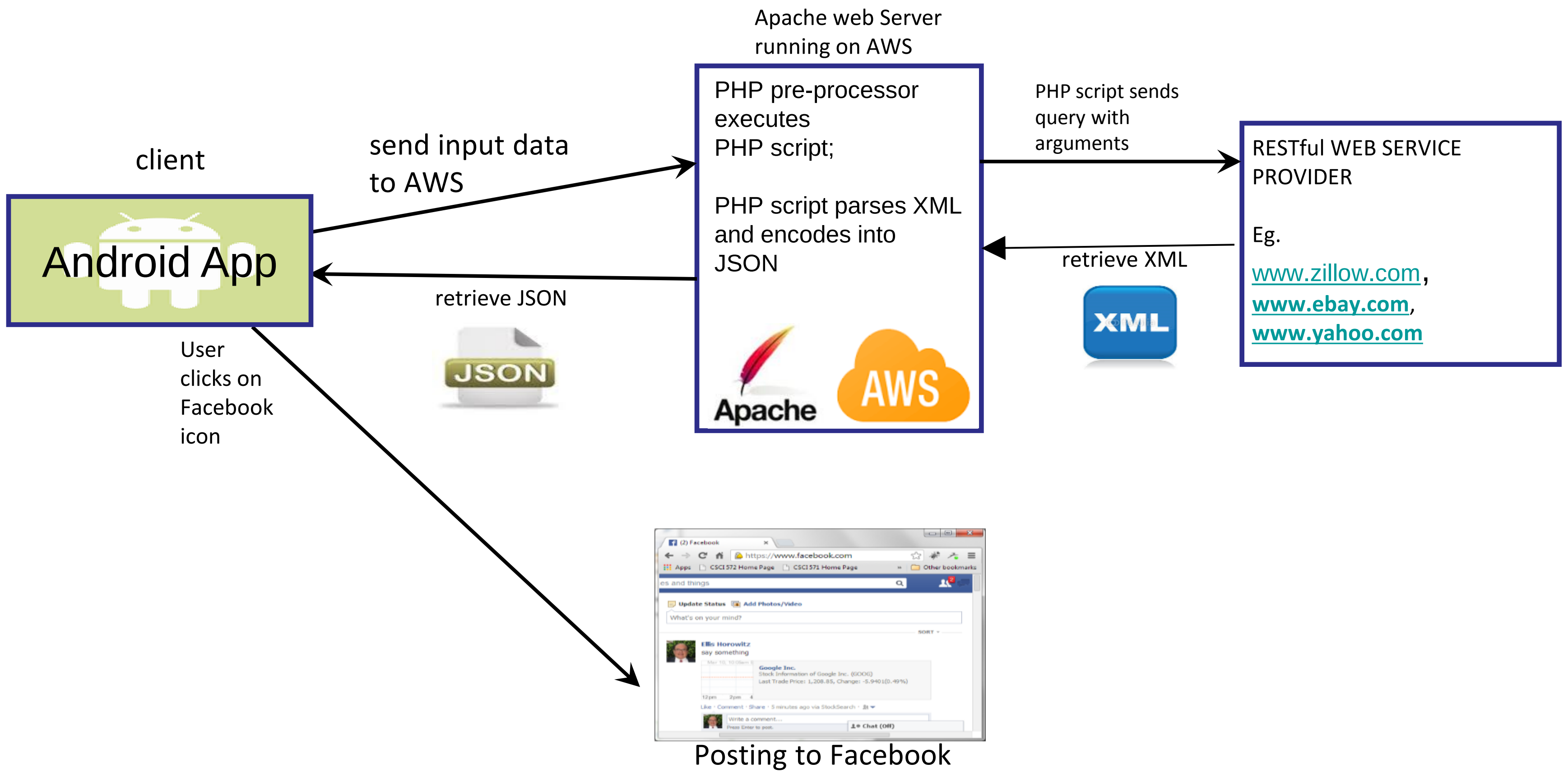
What is needed

- You will need to download and install Android Studio
  https://developer.android.com/sdk/index.html

- Download the Facebook Android SDK, 4.0.1 at
  https://developers.facebook.com/docs/android/getting-started

- Register your new App with Facebook and get an Application ID
  https://developers.facebook.com/apps/

HW#9 Architecture Overview


client
Android App
send input data to AWS
retrieve JSON
JSON
Apache web Server running on AWS
PHP pre-processor executes PHP script;
PHP script parses XML and encodes into JSON
Apache
AWS
PHP script sends query with arguments
retrieve XML
XML
RESTful WEB SERVICE PROVIDER
Eg.
www.zillow.com, www.ebay.com, www.yahoo.com
User clicks on Facebook icon
Posting to Facebook

HW9 Implementation

- You will create 3 activities and a Manifest file
- **AndroidManifest.xml**
- **MainActivity.java** – routine that controls the entire process
  - Puts up initial user interface,
  - sets Listeners to buttons
  - Validates input
  - Calls AWS server
- **ResultActivity.java**
  - implements the table of results using the json result
- **DetailsActivity.java**
  - Displays data for an item in a tabular view
  - Facebook request

AndroidManifest.xml File

Every application must have an AndroidManifest.xml file in its root directory. The manifest presents essential information about the application to the Android system. Among other things, the manifest does the following:

- It names the Java package for the application.
- It describes the components of the application — the activities, services, broadcast receivers, and content providers that the application is composed of.
- It names the classes that implement each of the components and publishes their capabilities.

See http://developer.android.com/guide/topics/manifest/manifest-intro.html.

Please note that the file is created by default on creation of a new Android project using Android Studio IDE.

UI Controls in Android (1 of 2)

For the homework exercise, you can use the following widgets (not limited to):

- **TextView** (i.e., label)
  http://developer.android.com/reference/android/widget/TextView.html
- **EditText** (i.e., text field)
  http://developer.android.com/reference/android/widget/EditText.html
- **Spinner** (i.e., drop-down list)
  http://developer.android.com/reference/android/widget/Spinner.html
- **Button**
  http://developer.android.com/reference/android/widget/Button.html
- **ImageButton**
  http://developer.android.com/reference/android/widget/ImageButton.html
- **ImageView**
  http://developer.android.com/reference/android/widget/ImageView.html

UI Controls in Android (2 of 2)

- **ImageSwitcher** (It is useful to animate an Image on screen)
  http://developer.android.com/reference/android/widget/ImageSwitcher.html
- **TextSwitcher** (It is useful to animate a label on screen)
  http://developer.android.com/reference/android/widget/TextSwitcher.html
- **TableLayout**
  http://developer.android.com/reference/android/widget/TableLayout.html
- **TableRow**
  http://developer.android.com/reference/android/widget/TableRow.html
- **RelativeLayout**
  http://developer.android.com/reference/android/widget/RelativeLayout.html
- **LinearLayout** (It arranges “components” in vertical or horizontal order, via orientation attribute.)
  http://developer.android.com/reference/android/widget/LinearLayout.html
- **ScrollView**
  http://developer.android.com/reference/android/widget/ScrollView.html

RelativeLayout

RelativeLayout lets you position your component base on the nearby (relative or sibling) component's position. You can use "above, below, left and right" to arrange the component position.

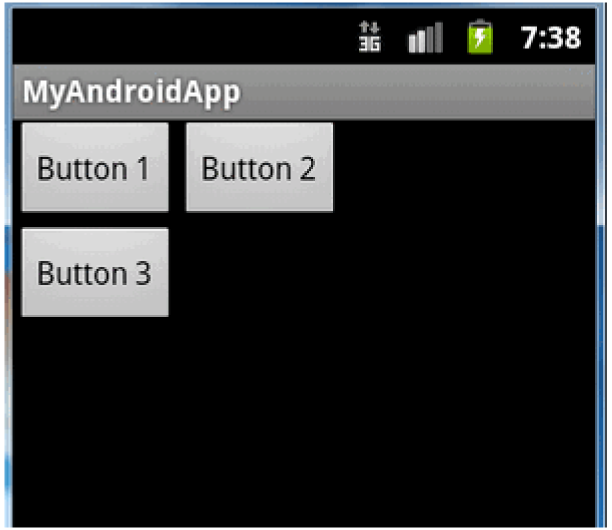
```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <Button
        android:id="@+id/btnButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 1"/>

    <Button
        android:id="@+id/btnButton2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 2"
        android:layout_toRightOf="@+id/btnButton1"/>

     <Button
        android:id="@+id/btnButton3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 3"
        android:layout_below="@+id/btnButton1"/>

</RelativeLayout>
```



Linear Layout

LinearLayout is a common layout that arranges “component” in vertical or horizontal order, via *orientation* attribute

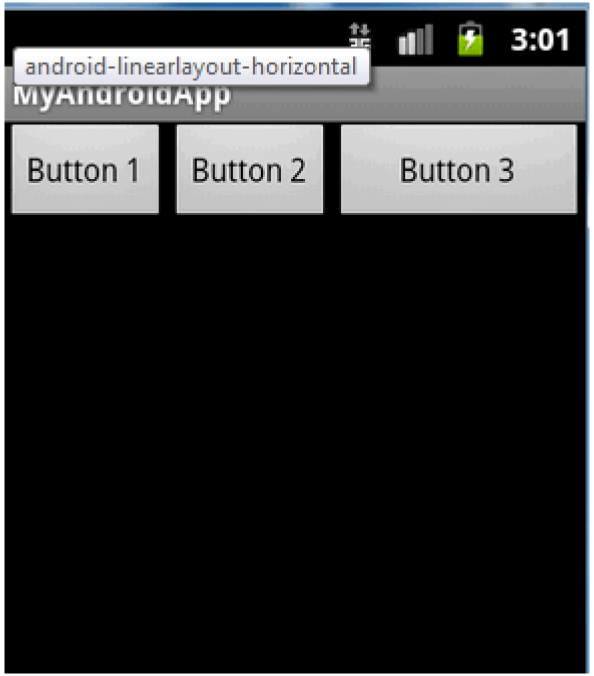
```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="horizontal" >

<Button android:id="@+id/button1"
android:layout_width="wrap_content"
android:layout_height="wrap_content" android:text="Button 1"/>

<Button android:id="@+id/button2"
android:layout_width="wrap_content"
android:layout_height="wrap_content" android:text="Button 2" />

<Button android:id="@+id/button3"
android:layout_width="wrap_content"
android:layout_height="wrap_content" android:text="Button 3"
android:layout_weight="1"/>

</LinearLayout>
```



MainActivity.java (1 of 3)

onCreate does the following

•Display the search UI

•Register a click event with the search button.

–More info about OnClickListener: http://developer.android.com/guide/topics/ui/controls/button.html

onClickListener for the search button does the following

•Build the URL for AWS (xxx.elasticbeanstalk.com)

•Create a new task which is of type AsyncTask to fetch the JSON data. It will initiate an asynchronous call.

–http://developer.android.com/reference/android/os/AsyncTask.html

•Execute the task

MainActivity.java - AsyncTask (2 of 3)

Create a class that extends AsyncTask which overrides two essential methods – *doInBackground*, *onPostExecute*.

1. **doInBackground:** used to perform background computation that can take a long time

For our homework exercise we perform basically the following steps in doInBackground method:

- Set up the HTTP connection stream, see http://developer.android.com/reference/org/apache/http/client/HttpClient.html
- Use HttpGet to GET, see:

  http://developer.android.com/reference/org/apache/http/client/methods/HttpGet.html
- Retrieve the data with HttpResponse
- Return the data using an InputStream object, see: http://developer.android.com/reference/java/io/InputStream.html

MainActivity.java - AsyncTask (3 of 3)

2. **onPostExecute:** invoked on the UI thread after the background computation finishes. The result of the background computation is passed to this step as a parameter

For our homework exercise we perform basically the following steps in onPostExecute method:

- Retrieve JSON data using task’s onPostExecute. Call function to parse JSON, see http://developer.android.com/reference/org/json/JSONObject.html
- After parsing is complete, start a new activity, ResultActivity, passing the extracted data.

ResultActivity.java (1 of 2)

ResultActivity starts with onCreate

- onCreate does the following

- retrieves JSON data which was passed from MainActivity
- stores the data in a JSONObject
  - http://developer.android.com/reference/org/json/JSONObject.html
- parses the result
  - extracts all JSON Objects values
- shows the results
  - fills the table layout
  - Need to fetch images using asynctask (HTTP code using URLconnection, InputStream and BitmapFactory) See:
    - http://developer.android.com/reference/java/net/URLConnection.html
    - http://developer.android.com/reference/java/io/InputStream.html
    - http://developer.android.com/reference/android/graphics/BitmapFactory.html

DetailsActivity.java

- Shows the large image and main text content, Facebook button and tabs, all scrollable.
- Tabs are created with three buttons and three relative layouts, all contained in one relative layout.
- Register event handlers for all tab buttons and Facebook button, etc.
- Toggle visibility of respective tab layouts using event handlers.
- Add Facebook related code as described next.

FACEBOOK POST

For the latest version of Facebook SDK 4.0.1, share functionality may require following:

- **Modifications in AndroidManifest file:**
    - Introducing Fb Application Id

```
<meta-data android:name="com.facebook.sdk.ApplicationId" android:value="@string/facebook_app_id"/>
```

    - Adding FacebookActivity

```
<activity android:name="com.facebook.FacebookActivity"
          android:configChanges=
             "keyboard|keyboardHidden|screenLayout|screenSize|orientation"
          android:theme="@android:style/Theme.Translucent.NoTitleBar"
          android:label="@string/app_name" />
```

    - Adding Facebook Content Provider

```
<provider android:authorities="com.facebook.app.FacebookContentProvider{app_id}"
         android:name="com.facebook.FacebookContentProvider"
         android:exported="true"/>
```

FACEBOOK POST (Cont.)

- To implement the functionality you may use the following approach on click of the fb button: https://developers.facebook.com/docs/sharing/android
    - Initialize facebook sdk
    - Create a ShareDialog
    - Create LinkContent for the post
    - Share the LinkContent through ShareDialog
    - Register Callback for the ShareDialog
    - Bind onActivityResult for maintaining session