Weather app tutorial part 1

## Step 1:

Download “sunshine\_starter.zip” from the Moodle, expand and open in Android Studio.

This gives you a starter project with:

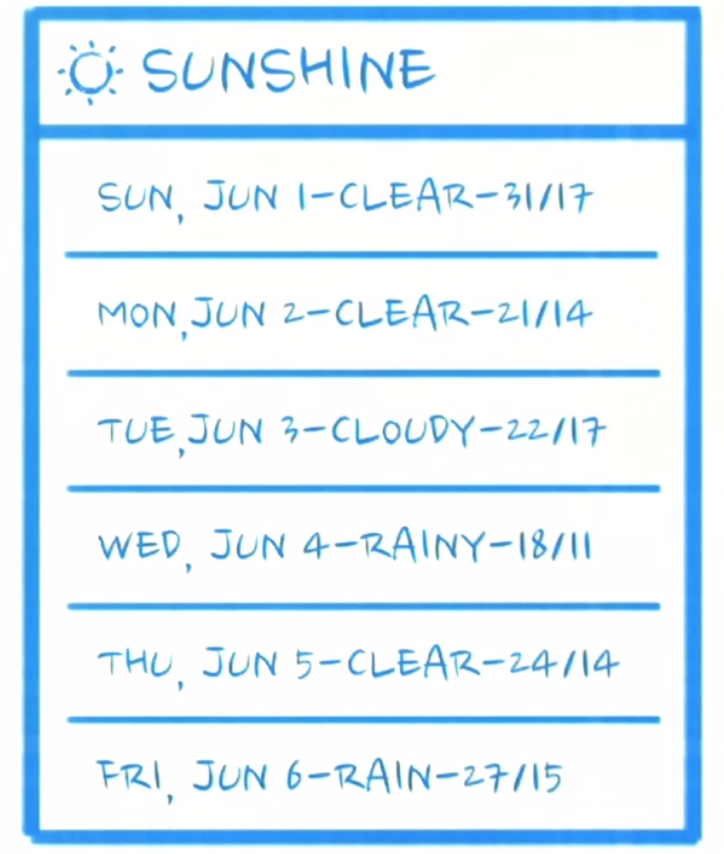
* MainActivity.java, with an associated XML layout “activity\_main.xml”
* ForeCastFragment.java, with an associated XML layout “fragement\_forecast.xml”

Note that this app makes use of Fragments. For details of Fragments see:

<https://developer.android.com/guide/components/fragments.html>

## Step 2

Create the layout elements you will use to display the individual forecast for each day.

* Create a new xml layout resource with the name “list\_item\_forecast.xml”
* Add a TextView with the id "@+id/list\_item\_forecast\_textview"
* Modify the resource “fragment\_forecast.xml” by removing the TextView and adding a ListView with the id "@+id/listview\_forecast"
* Change the dimensions of the ListView so that it takes up the full screen.
* Change the layout of “fragment\_forecast.xml” from a RelativeLayout to a FrameLayout.
* Run your app. At this point you’ll have a blank screen because the ListView is empty.

## Step 3

Create some fake data for the ListView[[1]](#footnote-1).

* Open the ForeCastFragment.java file.
* In the onCreateView() method create an ArrayList of strings for several forecast entries. For example “Today – sunny – 18/9C”.
* Create an ArrayAdapter that will be used to bind the fake data to the ListView.[[2]](#footnote-2)
* Find a reference to the ListView and set the adapter.
* Run your app. It should now show a list of forecasts.

## Step 4

Modify the file ForeCastFragment.java so that a Toast appears when the user click on an item in the list.

For details on Toasts see:

<https://developer.android.com/guide/topics/ui/notifiers/toasts.html>

## Step 5

Add the code shown in step 5 of the solution section to your project. It tries to open a network connection to OpenWeatherMaps and retrieve weather data.

Your app will probably crash at this stage.

1. Why does the app crash?
2. How can you prevent this?

Solutions

## Step 2 code:

**“list\_item\_forecast.xml”**

<?xml version="1.0" encoding="utf-8"?>  
  
<TextView xmlns:android="http://schemas.android.com/apk/res/android"  
  
 android:id="@+id/list\_item\_forecast\_textview" android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:minHeight="?android:attr/listPreferredItemHeight"  
 android:gravity="center\_vertical"  
/>

**“forecast\_fragment.xml”**

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:paddingLeft="@dimen/activity\_horizontal\_margin"  
 android:paddingRight="@dimen/activity\_horizontal\_margin"  
 android:paddingTop="@dimen/activity\_vertical\_margin"  
 android:paddingBottom="@dimen/activity\_vertical\_margin"  
 tools:context=".MainActivityFragment"  
 >  
  
 <ListView  
 android:id="@+id/listview\_forecast"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent" />  
  
</FrameLayout>

## Step 3 code

**Create some fake data in the onCreateView method.**

@Override  
public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {

View rootView = inflater.inflate(R.layout.*fragement\_forecast*, container, false);  
  
 String[] data = {  
 "Mon 6/23 - Sunny - 31/17",  
 "Tue 6/24 - Foggy - 21/8",  
 "Wed 6/25 - Cloudy - 22/17",  
 "Thurs 6/26 - Rainy - 18/11",  
 "Fri 6/27 - Foggy - 21/10",  
 "Sat 6/28 - TRAPPED IN WEATHERSTATION - 23/18",  
 "Sun 6/29 - Sunny - 20/7"  
 };  
 List<String> weekForecast = new ArrayList<String>(Arrays.*asList*(data));  
   
 return rootView;  
  
}

**Add the following code before the return statement to create an ArrayAdapter that will be used to bind the data to the ListView:**

// Now that we have some dummy forecast data, create an ArrayAdapter.  
// The ArrayAdapter will take data from a source (like our dummy forecast) and  
// use it to populate the ListView it's attached to.  
ArrayAdapter<String> mForecastAdapter =  
 **new** ArrayAdapter<String>(  
 getActivity(), // The current context (this activity)  
 R.layout.**list\_item\_forecast**, // The name of the layout ID.  
 R.id.**list\_item\_forecast\_textview**, // The ID of the textview to populate.  
 weekForecast);

**Add the following code before the return statement to find the ListView and bind the adapter:**

ListView listView = (ListView) rootView.findViewById(R.id.***listview\_forecast***);  
listView.setAdapter(**mForecastAdapter**);

## Step 4 code

Add the following code before the return statement in onCreateView(). It sets up a Clicklistener on the listView and makes a Toast based on the list item created.

listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
 @Override  
 public void onItemClick(AdapterView<?> adapterView, View view, int position, long l) {  
  
 Context context = getActivity();  
 CharSequence forecast = mForecastAdapter.getItem(position);  
  
 Toast.*makeText*(context, forecast, Toast.*LENGTH\_SHORT*).show();  
  
  
 }  
});

## Step 5

Add the following code before the return statement in onCreateView(). It attempts to connect to the OpenWeatherMap webservice and download location specific weather data.

// These two need to be declared outside the try/catch  
// so that they can be closed in the finally block.  
HttpURLConnection urlConnection = **null**;  
BufferedReader reader = **null**;  
  
// Will contain the raw JSON response as a string.  
String forecastJsonStr = **null**;  
  
**try** {  
 // Construct the URL for the OpenWeatherMap query  
 // Possible parameters are avaiable at OWM's forecast API page, at  
 // http://openweathermap.org/API#forecast  
 URL url = **new** URL(**"http://api.openweathermap.org/data/2.5/forecast/daily?q=94043&mode=json&units=metric&cnt=7"**);  
  
 // Create the request to OpenWeatherMap, and open the connection  
 urlConnection = (HttpURLConnection) url.openConnection();  
 urlConnection.setRequestMethod(**"GET"**);  
 urlConnection.connect();  
  
 // Read the input stream into a String  
 InputStream inputStream = urlConnection.getInputStream();  
 StringBuffer buffer = **new** StringBuffer();  
 **if** (inputStream == **null**) {  
 // Nothing to do.  
 **return null**;  
 }  
 reader = **new** BufferedReader(**new** InputStreamReader(inputStream));  
  
 String line;  
 **while** ((line = reader.readLine()) != **null**) {  
 // Since it's JSON, adding a newline isn't necessary (it won't affect parsing)  
 // But it does make debugging a \*lot\* easier if you print out the completed  
 // buffer for debugging.  
 buffer.append(line + **"\n"**);  
 }  
  
 **if** (buffer.length() == 0) {  
 // Stream was empty. No point in parsing.  
 **return null**;  
 }  
 forecastJsonStr = buffer.toString();  
} **catch** (IOException e) {  
 Log.e(**"PlaceholderFragment"**, **"Error "**, e);  
 // If the code didn't successfully get the weather data, there's no point in attemping  
 // to parse it.  
 **return null**;  
} **finally**{  
 **if** (urlConnection != **null**) {  
 urlConnection.disconnect();  
 }  
 **if** (reader != **null**) {  
 **try** {  
 reader.close();  
 } **catch** (**final** IOException e) {  
 Log.e(**"PlaceholderFragment"**, **"Error closing stream"**, e);  
 }  
 }  
}

1. <https://developer.android.com/guide/topics/ui/layout/listview.html> [↑](#footnote-ref-1)
2. For details of ArrayAdapters see: <https://developer.android.com/guide/topics/ui/declaring-layout.html#FillingTheLayout> [↑](#footnote-ref-2)