CST2335 – Graphical Interface Programming

Lab 6

Introduction:

The goal of this lab is become familiar with writing files, asynchronous tasks, and network connectivity. You will learn how to connect to a web server and download some images and save them to your application’s file directory.

References:

1. <https://developer.android.com/training/basics/network-ops/xml.html>
2. <https://developer.android.com/training/basics/data-storage/files.html>

Steps:

1. From the command line, create a branch of your software from Lab 5. If you have a windows computer, open the “Git Bash” program. If you have an Apple or Linux computer, just open a terminal. On the command line, navigate to your AndroidStudioProjects folder, and then to your Labs work folder. Once there, type “**git branch**” to list the branches on your computer. Next type “**git branch Lab6**” to create a new branch. Then type “**git checkout Lab6**” to move to that branch. Now type “**git branch**” to list all the branches, and there should be a star (\*) next to Lab6 to show it is your current branch.
2. Create a new Activity called WeatherForecast. Right-click on the app directory and select “New” -> “Activity” -> “Empty Activity”. Add a button to the ProfileActivity page, which should launch the WeatherForecast activity when the user clicks on the button.
3. In the AndroidManifest.xml, you will need use the following permissions:

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

1. In the WeatherForecast activity, create a layout that shows:
   * An ImageView for the current weather. Set the width and height to be 200dp.
   * A TextVew for the current temperature.
   * A TextView for the min temperature.
   * A TextView for the max temperature.
   * A TextView for the UV rating
   * A normal progress bar. Set the initial visibility to “invisible”. To make the progress bar a horizontal bar instead of a spinning progress, add this parameter to the ProgressBar tag: ***style="?android:attr/progressBarStyleHorizontal"***

In the onCreate() function, write code to set the progressBar’s visibility to View.Visible so that the progress bar will show.

1. Create an inner class in WeatherForecast, called ForecastQuery, which extends AsyncTask<String, Integer, String>. The class should have 4 string variables for the UV, min, max, and current temperature. There should also be a Bitmap variable to store the picture for the current weather.
2. We will be using a web server to tell us what the weather is in Ottawa. The URL to use is:

<http://api.openweathermap.org/data/2.5/weather?q=ottawa,ca&APPID=7e943c97096a9784391a981c4d878b22&mode=xml&units=metric>

The parameters are: **q=** … the city you want the weather for.

**APPID=** … an API key for measuring how many queries per hour. If we go over the limit for the free service level, then it will stop working.

**Mode=** … Can be JSON, XML, HTTP that gets returned

**Units=metric** … we want units in Celcius, not Kelvin or Imperial

Try pasting the URL into a web browser and you can see the XML data that is returned.

For getting the UV rating, use this URL:

<http://api.openweathermap.org/data/2.5/uvi?appid=7e943c97096a9784391a981c4d878b22&lat=45.348945&lon=-75.759389>

The parameters are lat, and lon for latitude and longitude.

***doInBackground( )***

1. Write the protected String doInBackground(String …args) function for the ForecastQuery class. Look at the example code from branch “week6\_xml” on the InClassExamples code on GitHub. Copy the code in that function to the doInBackground() function. For this lab, the URL will be the one mentioned above. You should also change the starting tags that you are looking for. Each time, you should check if the current tag’s name is “temperature”. If it is, then you must the get the value, min, and max parameters and save the text so that you can write the strings in a later step. XMLPullParser has a getAttributeValue(String namespace, String name) function which will give you those values. The namespace should be null, but the attributes you are looking for are “value”, “min”, “max”.
2. For each of the attributes, call publishProgress() with 25, 50, 75 as the parameters to show the progress of retrieving the data. You also need to look for the “weather” tag. The attribute you want to store is “icon”, representing the iconName to show. Run your application to verify that your web connection works properly, and your XMLPullParser is finding the values you need for later.
3. The OpenWeatherMap website also has icons for showing “Cloudy”, or “Sunny”, or “Raining”, etc. The URL you must build is "http://openweathermap.org/img/w/" + iconName + ".png". Here is some code to download an image:

*Bitmap image = null;*

*URL url = new URL(urlString);*

*connection = (HttpURLConnection) url.openConnection();*

*connection.connect();*

*int responseCode = connection.getResponseCode();*

*if (responseCode == 200) {*

*image = BitmapFactory.decodeStream(connection.getInputStream());*

*}*

You must build a Bitmap object, and then save it to the local storage. Once you have downloaded the image, call publishProgress() with 100 as the parameter to show that the progress is completed. Save the Bitmap object to the local application storage with the following code:

*FileOutputStream outputStream = openFileOutput( iconName + ".png", Context.MODE\_PRIVATE);*

*image.compress(Bitmap.CompressFormat.PNG, 80, outputStream);*

*outputStream.flush();*

*outputStream.close();*

Now, add code to check if your cloudy, sunny, raining images are already present in the local storage directory:

*public boolean fileExistance(String fname){*

*File file = getBaseContext().getFileStreamPath(fname);*

*return file.exists(); }*

If the Image file exists, then you don’t need to re-download it, just read it from your disk:

*FileInputStream fis = null;*

*try { fis = openFileInput(imagefile); }*

*catch (FileNotFoundException e) { e.printStackTrace(); }*

*Bitmap bm = BitmapFactory.decodeStream(fis);*

Write Log.i() messages showing which filename you are looking for, and a message saying if you found the image locally, or if you need to download it.

For the UV rating, you will need to use the given URL, but use a JSON object. Look at slides #26 and #27:

JSONObject jObject = new JSONObject(result);

From jObject, you want to retrieve the ***value*** parameters as floats:

float value = jObject.getDouble(“value”);

***onProgressUpdate( )***

1. Write the onProgressUpdate(Integer …value) function so that it sets the visibility of your progressBar to visible. Also, it should set the progress of the progressBar to the variable value[0] being passed in.

***onPostExecute( )***

1. In the onPostExecute function, update the GUI components with the min, max, and current temperature that you have read from the XML. Also update the ImageView with the Bitmap that you downloaded. Also, set the visibility of the progress bar to invisible, using the setVisibility(View.INVISIBLE ) function.
2. Demonstrate your work to the lab professor showing the following parts of your lab work: **(9 marks)**
3. Clicking login starts the Profile activity. Clicking on “Weather Forecast” button launches the WeatherForecast activity. **+1**
4. Show that the Debug information window has displayed your Debug Information that about finding the weather image locally, or downloading it.. **+1**
5. The progress bar shows the progress of loading information from the internet. **+1**
6. The weather information is correctly displayed on your activity. **+4**
7. The progress bar is hidden. **+1**
8. Paste the weather query into a web browser to show that the results match **+1**

In order to see the progress bar properly, you might have to set your Android virtual device network speed to GPRS:



1. Make sure to push all 3 branches to your github account to save your work!