Android Application Development, COMP 10073

Mohawk College, Winter 2021

Web API Demonstration

This example will illustrate the core ideas necessary for the final project. Start by reviewing this example, then take a look at the project specifications. At this point in the course you should know enough to satisfy most of the project requirements. We will cover a few more Android features and services in weeks 10, 11, and 12 which you may wish to include as part of your project, although the Web API should form the core.

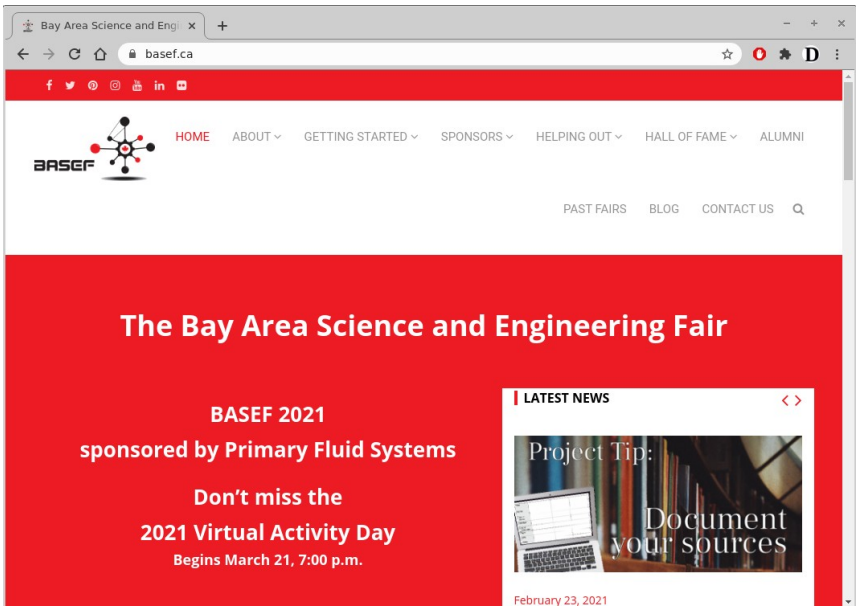
Permissions

Your application will need some normal permissions. These need to be included in the manifest file, but don't require any special dialogs with the user:

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

Example Webservice

http://basef.ca



The Bay area science and engineering fair collects local highschool science fair projects for students living in the City of Hamilton, the Regional Municipality of Halton, Six Nations, Haldimand, Norfolk, and Brant Counties.

The website also provides a basic WebAPI that gives access to their database.

http://www.basef.ca/api/basefws.php

activity_main.xml

The layout for our application is based on the ListView example from the previous lecture. Take a look at the details from that discussion.

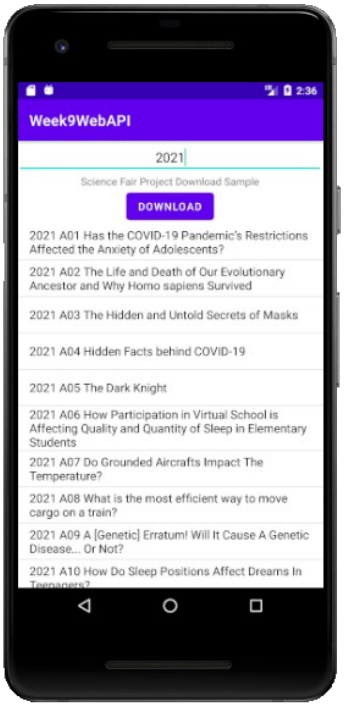
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:ems="10"
        android:hint="Enter Year"
        android:inputType="number"
        android:text=""
        android:textAlignment="center" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Science Fair Project Download Sample"
        android:layout_gravity="center_horizontal"
        android:id="@+id/textView" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Download"
        android:id="@+id/button"
        android:onClick="startDownload"
        android:layout_gravity="center_horizontal"
        android:layout_centerHorizontal="true" />

    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/listView"
        android:layout_gravity="center_horizontal"
        android:layout_centerHorizontal="true" />
</LinearLayout>
```



Downloading Data

To download data from the WebAPI we use an AsynchTask, also discussed previously. Create a new class called “DownloadAsyncTask”. The DownloadAsyncTask class will do the work of fetching data in the background. The onClick handler is similar to the template from the previous lecture.

```
/**
 * onClick handler composes URI and initiates download in background
 * @param view - from button
 */
public void startDownload(View view) {

    DownloadAsyncTask dl = new DownloadAsyncTask();

    // Build call to Webservice
    String uri = "http://www.basef.ca/api/basefws.php";

    EditText editYear = (EditText) findViewById(R.id.editText);
    String sYear = editYear.getText().toString();
    if (!sYear.equals("")) {
        // Filter - numbers (Year) no quotes on value, strings have quotes on value
        uri += "?filter=[{\"type\":\"number\",\"column\":\"Year\",\"value\":" + sYear + "}]";
    }

    Log.d(TAG, "startDownload " + uri);
    dl.execute(uri);
}
```

Communicating between Classes

For this example we will store the current activity in a private static class attribute which will be updated any time we create the current activity.

```
public class MainActivity extends AppCompatActivity {

    public static final String TAG = "==MainActivity==";
    /** single instance of current activity **/
    private static Activity currentActivity = null;

    /**
     * Set the current Activity each time onCreate is called
     * @param savedInstanceState
     */

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        currentActivity = this;
        Log.d(TAG, "onCreate");
    }

    /**
     * Provides access to the current activity
     * @return foreground Activity
     */
    public static Activity getCurrentActivity() {
        return currentActivity;
    }
```

URL

https://developer.android.com/reference/java/net/URL

Class URL represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web. A resource can be something as simple as a file or a directory, or it can be a reference to a more complicated object, such as a query to a database or to a search engine.

Creates a URL object from the String representation.

```
URL url = new URL(params[0]);
```

HttpURLConnection

https://developer.android.com/reference/java/net/HttpURLConnection

Obtain a new HttpURLConnection by calling URL#openConnection() and casting the result to HttpURLConnection. Get is the default, but you can prepare other connection requests using conn.setRequestMethod().

```
// Open the Connection - GET is the default setRequestMethod
HttpURLConnection conn = (HttpURLConnection)url.openConnection();
```

Read the response

Response headers typically include metadata such as the response body's content type and length, modified dates and session cookies. The response body may be read from the stream returned by URLConnection.getInputStream(). If the response has no body, that method returns an empty stream.

```
// Read the response
int statusCode = conn.getResponseCode();
if (statusCode == 200) {
    InputStream inputStream = new BufferedInputStream(
            conn.getInputStream());
    BufferedReader bufferedReader =
            new BufferedReader(new InputStreamReader(inputStream,
                    "UTF-8"));
    while ((line = bufferedReader.readLine()) != null) {
        results.append(line);
    }
}
```

doInBackground

```
public class DownloadAsyncTask extends AsyncTask<String, Void, String> {

    public static final String TAG = "==MainActivity==";

    /**
     * Download data from the supplied URL, catch exceptions
     * @param params - first parameter is the URL
     * @return a string that concatenates all of the output together, null on failure
     */
    @Override
    protected String doInBackground(String... params) {

        Log.d(TAG, "Starting Background Task");
        StringBuilder results = new StringBuilder();

        try {
            URL url = new URL(params[0]);
            String line = null;

            // Open the Connection - GET is the default setRequestMethod
            HttpURLConnection conn = (HttpURLConnection)
                    url.openConnection();

            // Read the response
            int statusCode = conn.getResponseCode();
            if (statusCode == 200) {
                InputStream inputStream = new BufferedInputStream(
                        conn.getInputStream());
                BufferedReader bufferedReader =
                        new BufferedReader(new InputStreamReader(inputStream,
                                "UTF-8"));
                while ((line = bufferedReader.readLine()) != null) {
                    results.append(line);
                }
            }
            Log.d(TAG, "Data received = " + results.length());
            Log.d(TAG, "Response Code: " + statusCode);

        } catch (IOException ex) {
            Log.d(TAG, "Caught Exception: " + ex);
        }

        return results.toString();
    }
```

JSON

https://en.wikipedia.org/wiki/JSON

JSON is a language-independent data format. It was derived from JavaScript, but many modern programming languages include code to generate and parse JSON-format data. The official Internet media type for JSON is application/json. JSON filenames use the extension .json.

```
{
   "name":"John",
   "surname":"Doe",
   "cars":[
      {
         "manufacturer":"Audi",
         "model":"A4",
         "capacity":1.8,
         "accident":false
      },
      {
         "manufacturer":"Škoda",
         "model":"Octavia",
         "capacity":2.0,
         "accident":true
      }
   ],
   "phone":2025550191
}
```

GSON

https://en.wikipedia.org/wiki/Gson

Gson (also known as Google Gson) is an open-source Java library to serialize and deserialize Java objects to (and from) JSON. Example conversion

```
import Person.Person;
import com.google.gson.Gson;

public class Main {
    public static void main(String[] args) {
        Gson gson = new Gson();
        String json = "{\"name\":\"John\",\"surname\":\"Doe\",\"cars\":
[{\"manufacturer\":\"Audi\",\"model\":\"A4\"," +
                "\"capacity\":1.8,\"accident\":false},
{\"manufacturer\":\"Škoda\",\"model\":\"Octavia\",\"capacity\"" +
                ":2.0,\"accident\":true}],\"phone\":2025550191}";
        Person johnDoe = gson.fromJson(json, Person.class);
        System.out.println(johnDoe.toString());
    }
}
```

Before we can import gson we need to add it explicitly to our gradle.build file at the application level in the dependencies section, include:

```
implementation 'com.google.code.gson:gson:2.8.6'
```

We also define a simple list class as:

```
public class FairList extends ArrayList<Projects> {
    private static final long serialVersionUID = 1L;
}
```

In the SimpleList example, we used an ArrayList of items “String”. Define the class Projects for storing results. We will use GSON to translate our data into an ArrayList of Projects, each item will have a toString().

```
public class Projects {
    public String   _id;
    public String   Year;
    public String   ProjectNum;
    public String   Title;
    @Override
    public String toString() {
        return Year + " " + ProjectNum + " " + Title;
    }
}
```

Also define a simple class called FairList that creates an ArrayList of Projects

```
public class FairList extends ArrayList<Projects> {
    private static final long serialVersionUID = 1L;
}
```

GSON can parse our JSON object into

```
Gson gson = new Gson();
FairList fairlist = gson.fromJson(result, FairList.class);
```

Connect the ArrayAdapter to the fairlist, which is an ArrayList of Projects.

```
ListView lv = (ListView) myActivity.findViewById(R.id.listView);

ArrayAdapter<Projects> adapter =
        new ArrayAdapter<Projects>(myActivity,
android.R.layout.simple_list_item_1, fairlist);
lv.setAdapter(adapter);
```

onPostExecute

Some of the steps can fail, so we should check for null values before executing code. Recall that onPostExecute is called after doInBackground completes, and the return value from doInBackground gets passed into onPostExecute. The currentActivity might have changed in this time (phone rotation for example), so we need to do the lookup immediately before we connect the listview to the adapter.

```
/**
 * After download has completed associate, parse results
 * @param result - do nothing if results == null
 */
protected void onPostExecute(String result) {

    FairList fairlist = null;
    if (result == null) {
        Log.d(TAG, "no results");
    } else {
        Gson gson = new Gson();
        fairlist = gson.fromJson(result, FairList.class);
    }
    // fetch the current activity so we can lookup the ListView
    Activity currentActivity = MainActivity.getCurrentActivity();
    ListView lv = currentActivity.findViewById(R.id.listView);

    if (fairlist != null) {
        // if we populated fairlist then connect the adapter
        ArrayAdapter<Projects> adapter =
                new ArrayAdapter<Projects>(currentActivity,
                        android.R.layout.simple_list_item_1, fairlist);
        lv.setAdapter(adapter);
    } else {
        // clear the list
        lv.setAdapter(null);
    }
}
```

What about Rotation?

Our list is populated in onPostExecute(). When we rotate the phone the original layout is destroyed, along with our ListView object and its associated adapter. This very much defeats the idea of rotation.

One answer is to always execute the onClick() handler for the download button in onResume(). This works, but it is rather expensive as it involves re-downloading all the data each time the phone is rotated. Think about a better way to handle this problem for your project.