# Web Services Lab – Android Portion

## Using a Web Service from Android

|  |  |
| --- | --- |
|  |  |

1. Create a new project called WebServiceLab.
2. Make sure the manifest file has the uses permission for reading the Internet. It’s a child element of the manifest root element hence should be at the same level as application element.

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

1. Create a new Activity using the Basic(Blank) template with the name, CourseActivity. It will be the home for our fragments. Change the layout of its xml file to use a FrameLayout and provide an id. I called mine fragment\_container.
2. Create a new Fragment called CourseListFragment that will display the course information from the web service. Use the list Fragment template to create it as we did in the previous lab.
3. Delete the dummy code package. We will add our own model class that retrieves data from the service. Create a new model package called model. Create a class called Course inside this course package.
4. Perform the following steps for Course Class.
   1. Modify the Course class to add member variables (String type) for courseId, shortDescription, longDescription, prereqs. Use m as the prefix for member variables.
   2. Create the following constants in the class. These values must match the json names that we use on the web service.

**public static final** String ***ID*** = **"id"**, ***SHORT\_DESC*** = **"shortDesc"** , ***LONG\_DESC*** = **"longDesc"**, ***PRE\_REQS*** = **"prereqs"**;

* 1. Create a constructor that takes all the parameters specified in (a) as well as getters and setters.
  2. Make the Course class implement Serializable. This allows us to the pass the object as a parameter.

**public class** Course **implements** Serializable

* 1. Add a method to parse json String as shown below.

*/\*\*  
 \* Parses the json string, returns an error message if unsuccessful.  
 \* Returns course list if success.  
 \** ***@param courseJSON*** *\** ***@return reason or null if successful.*** *\*/***public static** String parseCourseJSON(String courseJSON, List<Course> courseList) {  
 String reason = **null**;  
 **if** (courseJSON != **null**) {  
 **try** {  
 JSONArray arr = **new** JSONArray(courseJSON);  
  
 **for** (**int** i = 0; i < arr.length(); i++) {  
 JSONObject obj = arr.getJSONObject(i);  
 Course course = **new** Course(obj.getString(Course.***ID***), obj.getString(Course.***SHORT\_DESC***)  
 , obj.getString(Course.***LONG\_DESC***), obj.getString(Course.***PRE\_REQS***));  
 courseList.add(course);  
 }  
 } **catch** (JSONException e) {  
 reason = **"Unable to parse data, Reason: "** + e.getMessage();  
 }  
  
 }  
 **return** reason;  
}

1. Make the following changes to the MyCourseRecyclerViewAdapter:
   1. Change the list to type Course.

**private final** List<Course> **mValues**;

* 1. Change the onCreateViewHolder to use fragment\_course.
  2. Change the onBindViewHolder to use the Course methods.

holder.**mIdView**.setText(**mValues**.get(position).getCourseId());  
holder.**mContentView**.setText(**mValues**.get(position).getShortDescription());

* 1. Change the ViewHolder class’s member variable to use Course instead of Item.

**public** Course **mItem**;

1. Download the PHP Code zip file (it’s in the same place as this document) and transfer all the php files into your own public\_html folder. I put all mine under public\_html/Android folder. Before you transfer, make sure to change all the uwnetids to your own and mysqlpassword to your own.
2. There is also a script file in the zip file that you can use to create the tables in your schema. I will show you how to do this in class or you can do it on your own.
3. Test to see if these files work on the browser. Type in the url of the php file. Change uwnetid to yours. [http://cssgate.insttech.washington.edu/~uwnetid/](http://cssgate.insttech.washington.edu/~mmuppa/)
4. To test if test.php page works, you must use the url with ?cmd=courses at the end.

**http://cssgate.insttech.washington.edu/~uwnetid/test.php?cmd=courses"**;

1. Make the following changes to the CourseListFragment class:
   1. Add a String URL link to your web service that retrieves the courses. Change the url to reflect your directory structure.

**private static final** String ***COURSE\_URL*** = **"http://cssgate.insttech.washington.edu/~uwnetid/test.php?cmd=courses"**;

* 1. Create a member variable for the RecyclerView so that we can access it in the thread to load the data.

**private** RecyclerView **mRecyclerView**;

* 1. Create a private class called DownloadCoursesTask to setup of asynchronous loading of the data.

**private class** DownloadCoursesTask **extends** AsyncTask<String, Void, String> {

* 1. Override doInBackground to process the URL as shown below.

@Override  
**protected** String doInBackground(String... urls) {  
 String response = **""**;  
 HttpURLConnection urlConnection = **null**;  
 **for** (String url : urls) {  
 **try** {  
 URL urlObject = **new** URL(url);  
 urlConnection = (HttpURLConnection) urlObject.openConnection();  
  
 InputStream content = urlConnection.getInputStream();  
  
 BufferedReader buffer = **new** BufferedReader(**new** InputStreamReader(content));  
 String s = **""**;  
 **while** ((s = buffer.readLine()) != **null**) {  
 response += s;  
 }  
  
 } **catch** (Exception e) {  
 response = **"Unable to download the list of courses, Reason: "** + e.getMessage();  
 }  
 **finally** {  
 **if** (urlConnection != **null**)  
 urlConnection.disconnect();  
 }  
 }  
 **return** response;  
}

* 1. Implement onPostExecute to populate the RecyclerView or to show a Toast message with the error.

@Override  
**protected void** onPostExecute(String result) {  
 *// Something wrong with the network or the URL.* **if** (result.startsWith(**"Unable to"**)) {  
 Toast.*makeText*(getActivity().getApplicationContext(), result, Toast.***LENGTH\_LONG***)  
 .show();  
 **return**;  
 }  
  
 List<Course> courseList = **new** ArrayList<Course>();  
 result = Course.*parseCourseJSON*(result, courseList);  
 *// Something wrong with the JSON returned.* **if** (result != **null**) {  
 Toast.*makeText*(getActivity().getApplicationContext(), result, Toast.***LENGTH\_LONG***)  
 .show();  
 **return**;  
 }  
  
 *// Everything is good, show the list of courses.* **if** (!courseList.isEmpty()) {  
 **mRecyclerView**.setAdapter(**new** MyCourseRecyclerViewAdapter(courseList, **mListener**));  
 }  
}

* 1. Launch the AsyncTask inside the onCreateView method of this class. Place the code before the “return view;”. Remove the adapter code from it.

@Override  
**public** View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 View view = inflater.inflate(R.layout.***fragment\_course\_list***, container, **false**);  
  
 *// Set the adapter* **if** (view **instanceof** RecyclerView) {  
 Context context = view.getContext();  
 **mRecyclerView** = (RecyclerView) view;  
 **if** (**mColumnCount** <= 1) {  
 **mRecyclerView**.setLayoutManager(**new** LinearLayoutManager(context));  
 } **else** {  
 **mRecyclerView**.setLayoutManager(**new** GridLayoutManager(context, **mColumnCount**));  
 }  
  
 }  
  
 DownloadCoursesTask task = **new** DownloadCoursesTask();  
 task.execute(**new** String[]{***COURSE\_URL***});  
  
 **return** view;  
}

1. Make the following changes to the CourseActivity:
   1. Modify the onCreate method to add the fragment.

**if** (savedInstanceState == **null** || getSupportFragmentManager().findFragmentById(R.id.***list***) == **null**) {  
 CourseListFragment courseListFragment = **new** CourseListFragment();  
 getSupportFragmentManager().beginTransaction()  
 .add(R.id.***fragment\_container***, courseListFragment)  
 .commit();  
}

* 1. Implement the listener for the Fragment.

**public class** CourseActivity **extends** AppCompatActivity  
 **implements** CourseListFragment.OnListFragmentInteractionListener,

* 1. Implement the method for fragment interaction but leave it empty for now.

@Override  
**public void** onListFragmentInteraction(Course item) { }

1. Run the app to test. It should show the list of courses.
2. Add a new Fragment using blank template called CourseDetailFragment. This is the fragment we will go to when we click on an item in CourseListFragment. We will follow the same procedure as Fragments Lab.
3. Make the following changes to CourseDetailFragment:
   1. Modify the layout of the xml to show the course information. I used LinearLayout with vertical orientation as we did in the Fragments lab.

<**LinearLayout 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:orientation="vertical"  
 tools:context=".CourseDetailFragment"**>  
  
 <**TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/course\_item\_id"** />  
  
 <**TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/course\_short\_desc"** />  
  
 <**TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/course\_long\_desc"** />  
  
 <**TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/course\_prereqs"** />  
  
</**LinearLayout**>

* 1. Create member variables to access the text views inside the class.

**private** TextView **mCourseIdTextView**;  
**private** TextView **mCourseShortDescTextView**;  
**private** TextView **mCourseLongDescTextView**;  
**private** TextView **mCoursePrereqsTextView**;

* 1. Modify onCreateView to set the member variable instances.

@Override  
**public** View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 *// Inflate the layout for this fragment* View view = inflater.inflate(R.layout.***fragment\_course\_detail***, container, **false**);  
 **mCourseIdTextView** = (TextView) view.findViewById(R.id.***course\_item\_id***);  
 **mCourseShortDescTextView** = (TextView) view.findViewById(R.id.***course\_short\_desc***);  
 **mCourseLongDescTextView** = (TextView) view.findViewById(R.id.***course\_long\_desc***);  
 **mCoursePrereqsTextView** = (TextView) view.findViewById(R.id.***course\_prereqs***);  
  
 **return** view;  
}

* 1. Add a method updateView that can be called to set the course information.

**public void** updateView(Course course) {  
 **if** (course != **null**) {  
 **mCourseIdTextView**.setText(course.getCourseId());  
 **mCourseShortDescTextView**.setText(course.getShortDescription());  
 **mCourseLongDescTextView**.setText(course.getLongDescription());  
 **mCoursePrereqsTextView**.setText(course.getPrereqs());  
 }  
}

* 1. Override onStart to use the updateView method.

@Override  
**public void** onStart() {  
 **super**.onStart();  
  
 *// During startup, check if there are arguments passed to the fragment.  
 // onStart is a good place to do this because the layout has already been  
 // applied to the fragment at this point so we can safely call the method  
 // below that sets the article text.* Bundle args = getArguments();  
 **if** (args != **null**) {  
 *// Set article based on argument passed in* updateView((Course) args.getSerializable(*COURSE\_ITEM\_SELECTED*));  
 }  
}

1. Add code to the fragment interaction method in CourseActivity to launch the detail fragment.

@Override  
**public void** onListFragmentInteraction(Course item) {  
  
 CourseDetailFragment courseDetailFragment = **new** CourseDetailFragment();  
 Bundle args = **new** Bundle();  
 args.putSerializable(CourseDetailFragment.*COURSE\_ITEM\_SELECTED*, item);  
 courseDetailFragment.setArguments(args);  
  
 getSupportFragmentManager().beginTransaction()  
 .replace(R.id.***fragment\_container***, courseDetailFragment)  
 .addToBackStack(**null**)  
 .commit();  
}

1. Run the app and click on the course list. It should take you to the detail fragment and you should be able to hit back to go back to the list.

Add a new course to our service (Next week) and check for network connection.