Birzeit University
Faculty of Engineering and Technology
Electrical and Computer Engineering Department
Advance Computer Systems Engineering Lab ENCS515

EXP. No. 8. Integrating REST API into Android Application

1. Objectives

- ❖ Integrate Android apps with RESTful web services.
- ❖ How to access data from RESTful web services using simple GET and POST requests.
- ❖ How to parse JSON object to java object.

2. Introduction

REST describes a set of architectural principles by which data can be transmitted over a standardized interface (such as HTTP (Hyper-Text Transfer Protocol)). The acronym REST (Representational State Transfer), this basically means that each unique URL is a representation of some object.

Exposing a system's resources through a RESTful API is a flexible way to provide different kinds of applications with data formatted in a standard way.

All the user interface actions are managed by the main thread, so you should not block this thread by a process needs a few of seconds to be executed. If you do that, the UI components will freeze, and you will receive ANR (application not responding) error. One of the ways to solve this error is using AsyncTask class.

AsyncTask is an abstract class which allows you to perform long/background operations and show its result on the main thread.

To integrate an Android app with a RESTful web service, you'll need to make calls over the network. You can choose from a few different HTTP clients, some that are included with the Android SDK, and some open-source libraries that are available from various organizations.

HttpURLConnection: It is an abstract class used to send and receive data over the web .It is included in Android SDK.

To find any resource you need on the Internet, you can use the URL (Uniform Resource Locator) class. The constructor of this class takes string parameter of the following structure:

protocol://host:port/path (E.g.: (http://www.mocky.io/v2/5b4e6b4e3200002c009c2a44))

In this experiment, we will request the API from the server using HttpURLConnection class. So we will build a simple android application to get a JSON object from the server .We will parse the JSON to java object using JSONObject class. Finally, we will print the data in a textview component. In this experiment we will get a JSON of the following structure:

```
[
    {
        "name": "",
        "age": "",
        "id": ""
    }
]
```

3. Procedure

- Designing the Main Activity Layout

Create a new Project and call it “REST Application”, create the basic UI and application structure. This application will consist of one activity (Main Activity) to get the data from the server, and to display the results. To implement the UI of the application, you will use the layout_main so that we can arrange our components. You will use the constraint layout for the design as shown in Figure 8.1, the code below shows xml file of the main activity.

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Main Activty"
android:textAppearance="@style/Base.TextAppearance.AppCompat.Display3"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.1" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="160dp"
        android:layout_marginEnd="8dp"
        android:text="Get Data"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <LinearLayout
        android:id="@+id/layout"
        android:layout_width="395dp"
        android:layout_height="507dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="32dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:background="@android:drawable/gallery_thumb"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/button"></LinearLayout>
    <ProgressBar
        android:id="@+id/progressBar"
        style="?android:attr/progressBarStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.62" />

</android.support.constraint.ConstraintLayout>
```

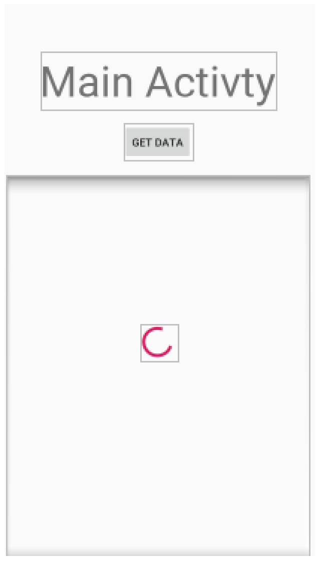


Figure 8.1 REST Application Layout

- Creating Model class and JSON to Array List class:

Getting the data from the URI will return a JSON array which we will paras this array into Array list using a special class. So, we will need a class to present the JSON Object and a class to paras the JSON Array into Array List

- Model Class

  You will create a class which will have the same attributes as the JSON Object has. Name this class “Student”. This class will have 3 attributes:

  private int ID

  private String name

  private Double age

  Create two constructors (empty and with attributes) for this class. Then create setters and getters for all attributes.

  Override toString method as we did in EXP. No. 2 sec 3.2 the following code shows the Student class.

```java
public class Student {
    private int ID;
    private String name;
    private Double age;

    public int getID() {
        return ID;
    }

    public void setID(int ID) {
        this.ID = ID;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Double getAge() {
        return age;
    }

    public void setAge(Double age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "Student{" +
                "\nID= " + ID +
                "\nname= " + name +
                "\nage= " + age +
                +'\n'+'}'+'\n';
    }
}
```

- JSON To Model Class

  Create a new class and call it "StudentJsonParser" this class will convert the JSON object we got from the REST API to Array List of the type Student. The following code shows how to convert the JSON List int Array List.

```java
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.List;

public class StudentJsonParser {

    public static List<Student> getObjectFromJson(String json) {
        List<Student> students;
        try {
            JSONArray jsonArray = new JSONArray(json);
            students = new ArrayList<>();
            for (int i = 0; i < jsonArray.length(); i++) {
                JSONObject jsonObject = new JSONObject();
                jsonObject = (JSONObject) jsonArray.get(i);
                Student student = new Student();
                student.setID(jsonObject.getInt("id"));
                student.setName(jsonObject.getString("name"));
                student.setAge(jsonObject.getDouble("age"));

                students.add(student);
            }
        } catch (JSONException e) {
            e.printStackTrace();
            return null;
        }
        return students;
    }
}
```

➢ HTTP manager Class

Add a new Java Class and call it “HttpManager”, this class is responsible for requesting a Http request to get the JSON Array from the URL we will pass to this class. you are now going to implement the REST API call. Add the getData method which is called in doInBackground method. This will first create a URL based on the string that is passed into the method. It will next open the connection and create a BufferedInputStream to receive the results from the call. The code below is for the HttpManager class.

```
import android.util.Log;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;


public class HttpManager {

    public static String getData(String URL) {
        BufferedReader bufferedReader = null;
        try {
            URL url = new URL(URL);
            HttpURLConnection httpURLConnection =
(HttpURLConnection) url.openConnection();
            bufferedReader = new BufferedReader(new
InputStreamReader(httpURLConnection.getInputStream()));
            StringBuilder stringBuilder = new StringBuilder();
            String line = bufferedReader.readLine();
            while (line != null) {
                stringBuilder.append(line + '\n');
                line = bufferedReader.readLine();
            }
            return stringBuilder.toString();
        } catch (Exception ex) {
            Log.d("HttpURLConnection", ex.toString());
        }
        return null;


    }
}
```

➢ Invoking the API call

You will be making the API call in a separate thread, which is always a good practice since the user interface will not be blocked while the call is being made. This is especially important in mobile devices that may drop network connections or experience high network latency. To accomplish this, you will implement an AsyncTask class. We should add a new class called ConnectionAsyncTask. We will override three methods, onPreExecute() doInBackground(String... params) and onPostExecute(String s). onPreExecute is exexuted at first before starting excecuting the doInBackground in the context of the UI thread. On the other hand the doInBackground executes in a separate thread so we will call the REST API and parse the results in this thread. onPostExecute executes in the context of the UI thread so we will use this method to display the results. We can access the UI components in methods that work in UI thread. So, we should show/hide the ProgressBar in these two methods. The code below shows the ConnectionAsyncTask class code.

```
import android.app.Activity;
import android.os.AsyncTask;

import java.util.List;


public class ConnectionAsyncTask extends AsyncTask<String, String,
String> {

    Activity activity;

    public ConnectionAsyncTask(Activity activity) {

        this.activity = activity;
    }

    @Override
    protected void onPreExecute() {

        ((MainActivity) activity).setButtonText("connecting");
        super.onPreExecute();
        ((MainActivity) activity).setProgress(true);
    }

    @Override
    protected String doInBackground(String... params) {

        String data = HttpManager.getData(params[0]);

        return data;
    }

    @Override
    protected void onPostExecute(String s) {
        super.onPostExecute(s);
        ((MainActivity) activity).setProgress(false);
        ((MainActivity) activity).setButtonText("connected");
        List<Student> students =
StudentJsonParser.getObjectFromJson(s);
        ((MainActivity) activity).fillStudents(students);

    }
}
```

- Implementing the Main Activity

In main activity we will get reference to the button and the root layout. The event handler of this button creates a new object of Connection AsyncTask class so that we connect to the server asynchronously, also we will make a method which will displays the users from array list. which will be called in the onPostExcecute method in the ConnectionAsyncTask class. The code below is for the MainActivity.

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.ProgressBar;
import android.widget.TextView;

import java.util.List;

public class MainActivity extends AppCompatActivity {
    Button button;
    LinearLayout linearLayout;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        setProgress(false);

        button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                ConnectionAsyncTask connectionAsyncTask = new
ConnectionAsyncTask(MainActivity.this);

connectionAsyncTask.execute("http://www.mocky.io/v2/5b4e6b4e3200002c
009c2a44");
            }
        });

        linearLayout = (LinearLayout) findViewById(R.id.layout);

    }

    public void setButtonText(String text) {
        button.setText(text);
    }

    public void fillStudents(List<Student> students) {
        LinearLayout linearLayout = (LinearLayout)
findViewById(R.id.layout);
```

```
            linearLayout.removeAllViews();
            for (int i = 0; i < students.size(); i++) {
                TextView textView = new TextView(this);
                textView.setText(students.get(i).toString());
                linearLayout.addView(textView);
            }
        }

        public void setProgress(boolean progress) {
            ProgressBar progressBar = (ProgressBar)
    findViewById(R.id.progressBar);
            if (progress) {
                progressBar.setVisibility(View.VISIBLE);
            } else {
                progressBar.setVisibility(View.GONE);
            }
        }
    }
```

➢ Adding the internet Permission

In android, to give your application the ability to use internet, special permission should be added in the manifest.xml file, open that file and add the following code before the application tag

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

➢ For API more than 28

- By default, the Http requests are disabled due to low security. And since we will make a Http request, you should enable this.
- To Enable Http requests add a new resource directory by right clicking on the res director and add new Android Resource Directory as shown in Figure 8.2. Then change the Resource Type in the new Resource Directory Screen to xml as shown in Figure 8.3.
- In the new xml Directory add a new Android Resource File and call it network_security_config and add the code below in this file.

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config cleartextTrafficPermitted="true">
        <trust-anchors>
            <certificates src="system" />
        </trust-anchors>
    </base-config>
</network-security-config>
```

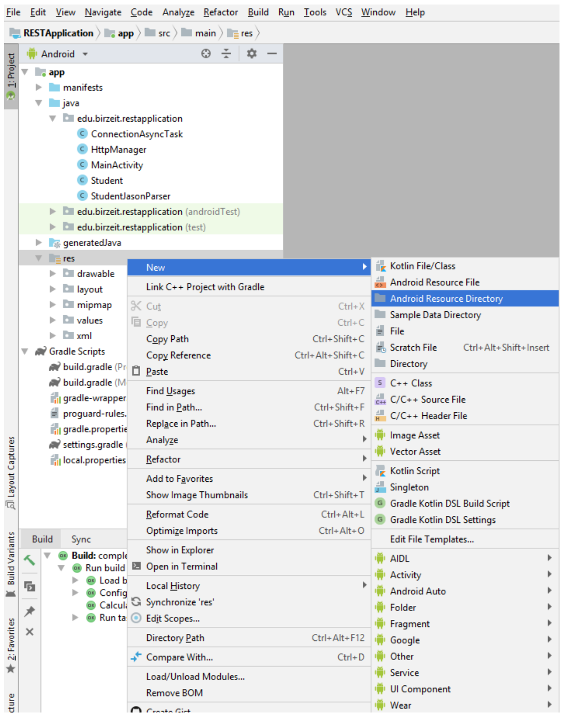
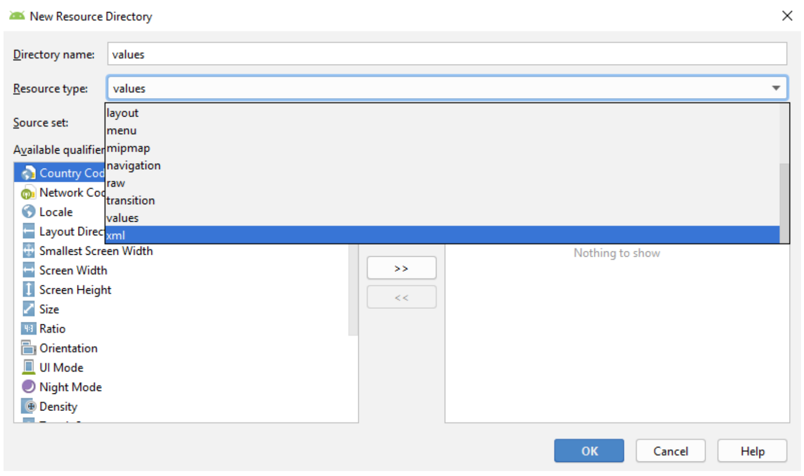


Figure 8.2 Adding new Resource Directory


New Resource Directory

Directory name: values

Resource type: values

Figure 8.3 New Resource Directory Screen

- In the Manifest File and in application header set the networkSecurityConfig to the file we added.

```
android:networkSecurityConfig="@xml/network_security_config"
```

the output of the Application is shown in Figure 8.4

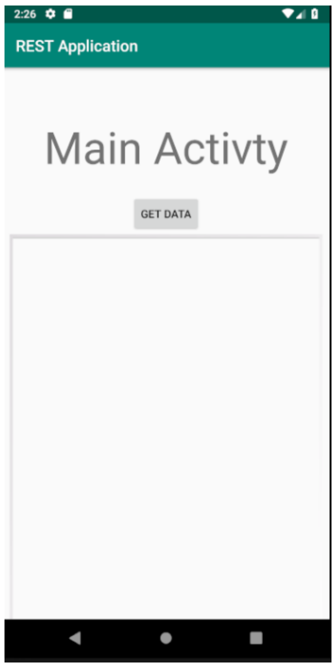


Figure 8.4.a Before Clicking the Button

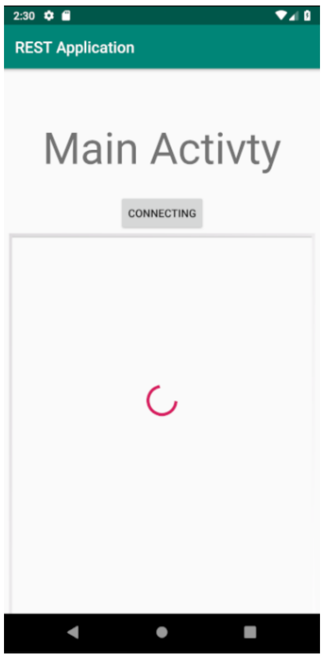
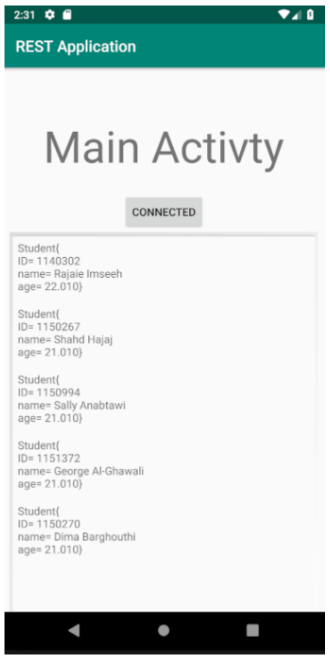


Figure 8.4. While Fetching the Data form the URL



```
Student{
ID= 1140302
name= Rajaie Imseeh
age= 22.010}

Student{
ID= 1150267
name= Shahd Hajaj
age= 21.010}

Student{
ID= 1150994
name= Sally Anabtawi
age= 21.010}

Student{
ID= 1151372
name= George Al-Ghawali
age= 21.010}

Student{
ID= 1150270
name= Dima Barghouthi
age= 21.010}
```

Figure 8.4.c After Fetching the Data and Displaying it

Figure 8.4 REST Application Output

4. Todo

This part will be given to you by the teacher assistant in the lab time.