Practical No. 10

Title: Android program to work with REST API

Aim: Create an application to demonstrate rest api

Introduction

Representational **S**tate **T**ransfer (REST) is an architectural style that defines a set of constraints to be used for creating web services. **REST API** is a way of accessing web services in a simple and flexible way without having any processing.
REST technology is generally preferred to the more robust Simple Object Access Protocol (SOAP) technology because REST uses less bandwidth, simple and flexible making it more suitable for internet usage. It's used to fetch or give some information from a web service. All communication done via REST API uses only HTTP request.

Working: A request is sent from client to server in the form of a web URL as HTTP GET or POST or PUT or DELETE request. After that, a response comes back from the server in the form of a resource which can be anything like HTML, XML, Image, or JSON. But now JSON is the most popular format being used in Web Services.

In **HTTP** there are five methods that are commonly used in a REST-based Architecture i.e., POST, GET, PUT, PATCH, and DELETE. These correspond to create, read, update, and delete (or CRUD) operations respectively. There are other methods which are less frequently used like OPTIONS and HEAD.

- **GET:** The HTTP GET method is used to **read** (or retrieve) a representation of a resource. In the safe path, GET returns a representation in XML or JSON and an HTTP

response code of 200 (OK). In an error case, it most often returns a 404 (NOT FOUND) or 400 (BAD REQUEST).

- **POST:** The POST verb is most often utilized to **create** new resources. In particular, it's used to create subordinate resources. That is, subordinate to some other (e.g. parent) resource. On successful creation, return HTTP status 201, returning a Location header with a link to the newly-created resource with the 201 HTTP status.

In this Practical, we are going to create an Android App that uses the Retrofit Library to download JSON Objects

What is Retrofit?

Retrofit is a REST Client library (Helper Library) used in Android and Java to create an HTTP request and also to process the HTTP response from a REST API. It was created by Square, you can also use retrofit to receive data structures other than JSON, for example SimpleXML and Jackson. Before we continue, let's briefly define REST Client and REST API in our context.

REST Client in our case is the Retrofit library that is used on the client side (Android) to make HTTP request to REST API

A REST API defines a set of functions which developers can perform requests and receive responses via HTTP protocol such as GET and POST.

We can also simply say that a RESTful API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data.

To use Retrofit in your Android Application, you'll need 3 major classes.

1. **An Interface which defines the HTTP operations (Functions or methods)**
   According to Square, creators of Retrofit documentation, Retrofit turns your HTTP API into a Java interface.

2. **A Retrofit class** which generates an implementation of the GitHubService interface. The below sample code would be inside the Retrofit class and this is how it creates an instance of Retrofit and implements the listRepos() method that's in the GitHubService Interface.

3. **The last of the 3 needed class is a simple POJO** that matches each field in the JSON response object gotten from querying an API. It's a simple class with getter and setter methods for each fields.

Retrofit Converters

Retrofit Converters are like an agreement between and Android client and the Server on the format on which data will be represented. Both parties can agree that for our communication, the format for data transfer will be JSON

Gson

Gson is for JSON mapping and can be added with the following dependency:

compile 'com.squareup.retrofit2:converter-gson:2.2.0

SimpleXML

SimpleXML is for XML mapping. You'll need the following line for your build.gradle:

compile 'com.squareup.retrofit2:converter-simplexml:2.2.0'

Exercise - Create android application to demonstrate REST API

Implementation:
Program:

MainActivity.java

```
package com.example.apiproject;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Toast;
import java.util.List;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class MainActivity extends AppCompatActivity {

    Apiinterface apiinterface;

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

        apiinterface= Retrofitinstance.getR().create(Apiinterface.class);

        apiinterface.getposts().enqueue(new Callback<List<postpojo>>() {
            @Override
            public void onResponse(Call<List<postpojo>> call,
Response<List<postpojo>> response) {
            if(response.body().size() > 0) {
                Toast.makeText(MainActivity.this, "List is not empty",
Toast.LENGTH_SHORT).show();
            }else{
                Toast.makeText(MainActivity.this, "List is empty",
Toast.LENGTH_SHORT).show();
            }}
            @Override
```

```
            public void onFailure(Call<List<postpojo>> call, Throwable t)
{

                Toast.makeText(MainActivity.this,"error",
Toast.LENGTH_SHORT).show();

            }
        });
    }}
```

Retrofitinstance.java

```
package com.example.apiproject;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;
public class Retrofitinstance {
    private static Retrofit r;
    private static final String
Baseurl="https://jsonplaceholder.typicode.com/";
    public static Retrofit getR() {
        if(r == null)
        {
            r=new Retrofit.Builder().baseUrl(Baseurl).
                    addConverterFactory(GsonConverterFactory.create()).
                    build();
        }
        return r;
    }}
```

Apiinterface.java

```
package com.example.apiproject;
import java.util.List;
import retrofit2.Call;
import retrofit2.http.GET;
public interface Apiinterface {
    // https://jsonplaceholder.typicode.com/
@GET("posts")
    Call<List<postpojo>> getposts();
}
```

postpojo.java

```
package com.example.apiproject;
public class postpojo {
    private float userId;
    private float id;
    private String title;
    private String body;
    // Getter Methods
    public float getUserId() {
        return userId;
    }
```

```
    public float getId() {
        return id;
    }

    public String getTitle() {
        return title;
    }

    public String getBody() {
        return body;
    }

    // Setter Methods

    public void setUserId(float userId) {
        this.userId = userId;
    }

    public void setId(float id) {
        this.id = id;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public void setBody(String body) {
        this.body = body;
    }}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

Output: