Practical
Manual
LAB

MCAL34 Mobile
Computing

MCA Sem
III

# Practical No. 10

## Title: Android program to work with REST API

**Aim: Create an application to demonstrate rest api**

### Introduction

**R**epresentational **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.

Practical
Manual
LAB

MCAL34 Mobile
Computing

MCA Sem
III

- **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.

Practical
Manual
LAB

MCAL34 Mobile
Computing

MCA Sem
III

**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:

Practical
Manual
LAB

MCAL34 Mobile
Computing

MCA Sem
III

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**

```java
package com.example.api;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.os.Bundle;
import android.widget.TextView;
import android.widget.Toast;

import com.google.gson.Gson;

import java.util.List;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class MainActivity extends AppCompatActivity {
    private RecyclerView recyclerView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

Practical
Manual
LAB

MCAL34 Mobile
Computing

MCA Sem
III

```java
        recyclerView=findViewById(R.id.recycleriew);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));

        Retrofit retrofit=new Retrofit.Builder()
                .baseUrl("https://jsonplaceholder.typicode.com/")
                .addConverterFactory(GsonConverterFactory.create())
                .build();

        ApiInterface apiInterface=retrofit.create(ApiInterface.class);
        Call<List<postpojo>> call=apiInterface.getposts();
        call.enqueue(new Callback<List<postpojo>>() {
            @Override
            public void onResponse(Call<List<postpojo>> call,
Response<List<postpojo>> response) {
                if(response.isSuccessful()){
                    Toast.makeText(MainActivity.this, "sucess",
Toast.LENGTH_SHORT).show();
                }
                List<postpojo> postpojoList=response. body();
                PostAdapter postAdapter=new
PostAdapter(MainActivity.this,postpojoList);
                recyclerView.setAdapter(postAdapter);
            }

            @Override
            public void onFailure(Call<List<postpojo>> call, Throwable t) {
                Toast.makeText(MainActivity.this, t.getMessage(),
Toast.LENGTH_SHORT).show();
            }
        });

    }
}
```

**postpojo.java**

```java
package com.example.api;

public class postpojo {
    private String userId;
    private String id;
    private String title;
    private String body;

    public String getUserId() {
        return userId;
    }

    public String getTitle() {
```

Practical
Manual
LAB

MCAL34 Mobile
Computing

MCA Sem
III

```java
        return title;
    }

    public String getId() {
        return id;
    }

    public String getBody() {
        return body;
    }
}
```

## PostAdapter.java

```java
package com.example.api;

import android.annotation.SuppressLint;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import java.util.List;


public class PostAdapter extends
RecyclerView.Adapter<PostAdapter.PostViewHolder> {
    List<postpojo> postList;
    Context context;
    public PostAdapter(Context context,List<postpojo> posts){
        this.context=context;
        postList=posts;
    }
    @NonNull
    @Override
    public PostViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View view=
LayoutInflater.from(context).inflate(R.layout.item,parent,false);
        return new PostViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull PostViewHolder holder, int
position) {
        postpojo post=postList.get(position);
```

Practical
Manual
LAB

MCAL34 Mobile
Computing

MCA Sem
III

```java
        holder.id.setText(post.getId());
        holder.userId.setText(post.getUserId());
        holder.title.setText(post.getTitle());
        holder.body.setText(post.getBody());

    }

    @Override
    public int getItemCount() {
        return postList.size();
    }

    public class PostViewHolder extends RecyclerView.ViewHolder{
        TextView title,id,userId,body;
        @SuppressLint("CutPasteId")
        public PostViewHolder(@NonNull View itemView){
            super(itemView);
            title=itemView.findViewById(R.id.textView5);
            id=itemView.findViewById(R.id.textView6);
            userId=itemView.findViewById(R.id.textView7);
            body =itemView.findViewById(R.id.textView8);
        }
    }
}
```

## ApiInterface.java

```java
package com.example.api;

import java.util.List;

import retrofit2.Call;
import retrofit2.http.GET;

public interface ApiInterface {
    @GET("posts")
    Call<List<postpojo>> getposts();
}
```

## ActivityMain.xml

```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">
```
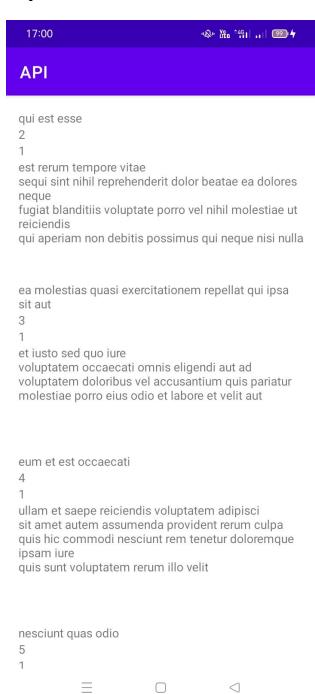
Practical
Manual
LAB

MCAL34 Mobile
Computing

MCA Sem
III

```xml
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycleriew"
        android:layout_width="409dp"
        android:layout_height="729dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

**item.xml**

```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="200dp">

    <LinearLayout
        android:layout_width="332dp"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <TextView
            android:id="@+id/textView5"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="TextView" />

        <TextView
            android:id="@+id/textView6"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="TextView" />

        <TextView
            android:id="@+id/textView7"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="TextView" />

        <TextView
            android:id="@+id/textView8"
            android:layout_width="match_parent"
            android:layout_height="200dp"
```

Practical
Manual
LAB

MCAL34 Mobile
Computing

MCA Sem
III

```
            android:text="TextView" />
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

**Output:**

API

qui est esse
2
1
est rerum tempore vitae
sequi sint nihil reprehenderit dolor beatae ea dolores
neque
fugiat blanditiis voluptate porro vel nihil molestiae ut
reiciendis
qui aperiam non debitis possimus qui neque nisi nulla


ea molestias quasi exercitationem repellat qui ipsa
sit aut
3
1
et iusto sed quo iure
voluptatem occaecati omnis eligendi aut ad
voluptatem doloribus vel accusantium quis pariatur
molestiae porro eius odio et labore et velit aut



eum et est occaecati
4
1
ullam et saepe reiciendis voluptatem adipisci
sit amet autem assumenda provident rerum culpa
quis hic commodi nesciunt rem tenetur doloremque
ipsam iure
quis sunt voluptatem rerum illo velit



nesciunt quas odio
5
1

Practical
Manual
LAB

MCAL34 Mobile
Computing

MCA Sem
III

Practical
Manual
LAB

MCAL34 Mobile
Computing

MCA Sem
III