



Andhra Pradesh State Skill Development Corporation



ANDROID APPLICATION DEVELOPMENT

RETROFIT

Retrofit

What is Retrofit

Retrofit is a REST Client for Java and Android. It makes it relatively easy to retrieve and upload JSON (or other structured data) via a REST based webservice. In Retrofit you configure which converter is used for the data serialization. Typically for JSON you use GSON, but you can add custom converters to process XML or other protocols. Retrofit uses the OkHttp library for HTTP requests.

Using Retrofit

To work with Retrofit you basically need the following three classes:

- Model class which is used as a JSON model
- Interfaces that define the possible HTTP operations
- Retrofit.Builder class - Instance which uses the interface and the Builder API to allow defining the URL end point for the HTTP operations.

Every method of an interface represents one possible API call. It must have a HTTP annotation (GET, POST, etc.) to specify the request type and the relative URL. The return value wraps the response in a Call object with the type of the expected result.

```
@GET("users")  
Call<List<User>> getUsers();
```

You can use replacement blocks and query parameters to adjust the URL. A replacement block is added to the relative URL with {}. With the help of the **@Path** annotation on the method parameter, the value of that parameter is bound to the specific replacement block.

```
@GET("users/{name}/commits")  
Call<List<Commit>> getCommitsByName(@Path("name") String name);
```

Query parameters are added with the **@Query** annotation on a method parameter. They are automatically added at the end of the URL.

```
@GET("users")  
Call<User> getUserById(@Query("id") Integer id);
```

The **@Body** annotation on a method parameter tells Retrofit to use the object as the request body for the call.

```
@POST("users")  
Call<User> postUser(@Body User user)
```

CONVERTERS

By default, Retrofit can only deserialize HTTP bodies into OkHttp's ResponseBody type and it can only accept its RequestBody type for @Body.

Converters can be added to support other types. Six sibling modules adapt popular serialization libraries for your convenience.

- **Gson:** com.squareup.retrofit2:converter-gson
- **Jackson:** com.squareup.retrofit2:converter-jackson
- **Moshi:** com.squareup.retrofit2:converter-moshi
- **Protobuf:** com.squareup.retrofit2:converter-protobuf
- **Wire:** com.squareup.retrofit2:converter-wire
- **Simple XML:** com.squareup.retrofit2:converter-simplexml
- **JAXB:** com.squareup.retrofit2:converter-jaxb
- **Scalars (primitives, boxed, and String):** com.squareup.retrofit2:converter-scalars

Practical Example for Retrofit

Here we will work on Google books api, In this books api by using book name we can search the books details like authors,title,Book Image

<https://www.googleapis.com/books/v1/volumes?q=two%20states> This is the url to get the book details based on book name.

Basically every url is divided into the 3 parts: - Base Url - path - Query

In the above url also we have a 3 parts: - Base Url => <https://www.googleapis.com/> - path => books/v1/volumes - Query => q=two%20states

Let's create a new androidstudio project with the name of **BookSearch** - Select the EmptyActivity

After creating the project you should have to add the below dependencies:

dependencies

```
{
    implementation 'com.squareup.retrofit2:retrofit:2.8.1'
    implementation 'com.squareup.retrofit2:converter-scalars:2.7.2'
}
```

If your app will crash then only use below compileOption

android

```
{
    compileOptions
    {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}
```

In the **activity_main.xml** file you should have take a 3 views i.e - EditText for to enter the bookname - Button for to search the bookdetails - Textview for to display the bookdetails

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/edittext"
        android:hint="Enter bookname"
        />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Search"
        android:onClick="search"
        />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Result"
        android:textSize="25sp"
        android:id="@+id/result"
        />

</LinearLayout>
```

MainActivity.java

```
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    EditText et_bookname;
    TextView tv_result;

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



```
et_bookname = findViewById(R.id.edittext);  
tv_result = findViewById(R.id.result);  
}
```

```
public void search(View view) {  
    String bookName = et_bookname.getText().toString();  
  
}
```

Now its time to create a one Interface to get the json response.

```
import retrofit2.Call;  
import retrofit2.http.GET;  
import retrofit2.http.Query;
```

```
public interface BookSearchService {  
    @GET("books/v1/volumes")  
    Call<String> getRepos(@Query("q") String name);  
}
```

In the **MainActivity.Java** file you should have to create a object for **Retrofit** class

```
Retrofit retrofit = new Retrofit.Builder().baseUrl("https://www.googleapis.com/")  
    .addConverterFactory(ScalarsConverterFactory.create()).build();
```

with in the **search** method you should have to get the response.

```
public void search(View view) {  
    String bookName = et_bookname.getText().toString();  
    BookSearchService service = retrofit.create(BookSearchService.class);  
    Call<String> response=service.getRepos(bookName);  
    response.enqueue(new Callback<String>() {  
        @Override  
        public void onResponse(Call<String> call, Response<String> response) {  
            Toast.makeText(MainActivity.this, ""+response.body(), Toast.LENGTH_SHORT).show();  
  
            String res = response.body();  
            try {  
                JSONObject object = new JSONObject(res);  
                JSONArray jsonArray = object.getJSONArray("items");  
                JSONObject indexObject=jsonArray.getJSONObject(0);  
                JSONObject volumeInfo = indexObject.getJSONObject("volumeInfo");  
                String name = volumeInfo.getString("title");  
                String authors = volumeInfo.getString("authors");  
                tv_result.setText(name+"\n"+authors);  
  
            } catch (Exception e) {  
                e.printStackTrace();  
            }  
        }  
    });  
}
```

```
}  
  
@Override  
public void onFailure(Call<String> call, Throwable t) {  
  
}  
});
```

The last step you should have to provide the internet permission with in **AndroidManifest.xml** file

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

OutPut

