Retrofit

Retrofit

- Retrofit android is a type-safe HTTP client for Android and Java.
- Retrofit is a REST Client for Java, Android, and Kotlin by Square inc under Apache 2.0 license.
- Retrofit will save your development time and also you can keep your code developer-friendly.
- Retrofit has given almost all the APIs to make a server call and to receive a response.
- Internally they also use GSON to do the parsing.
- Retrofit library makes downloading JSON or XML data from a web API fairly straightforward.
- Once the data is downloaded then it is parsed into a Plain Old Java Object (POJO) which must be defined for each "resource" in the response.

Retrofit

- Retrofit automatically serialize the JSON response using a POJO(Plain Old Java Object) which must be defined in advanced for the JSON Structure.

- To serialise JSON we need a converter to convert it into Gson first.

- We need to add the following dependencies in our build.grade file.

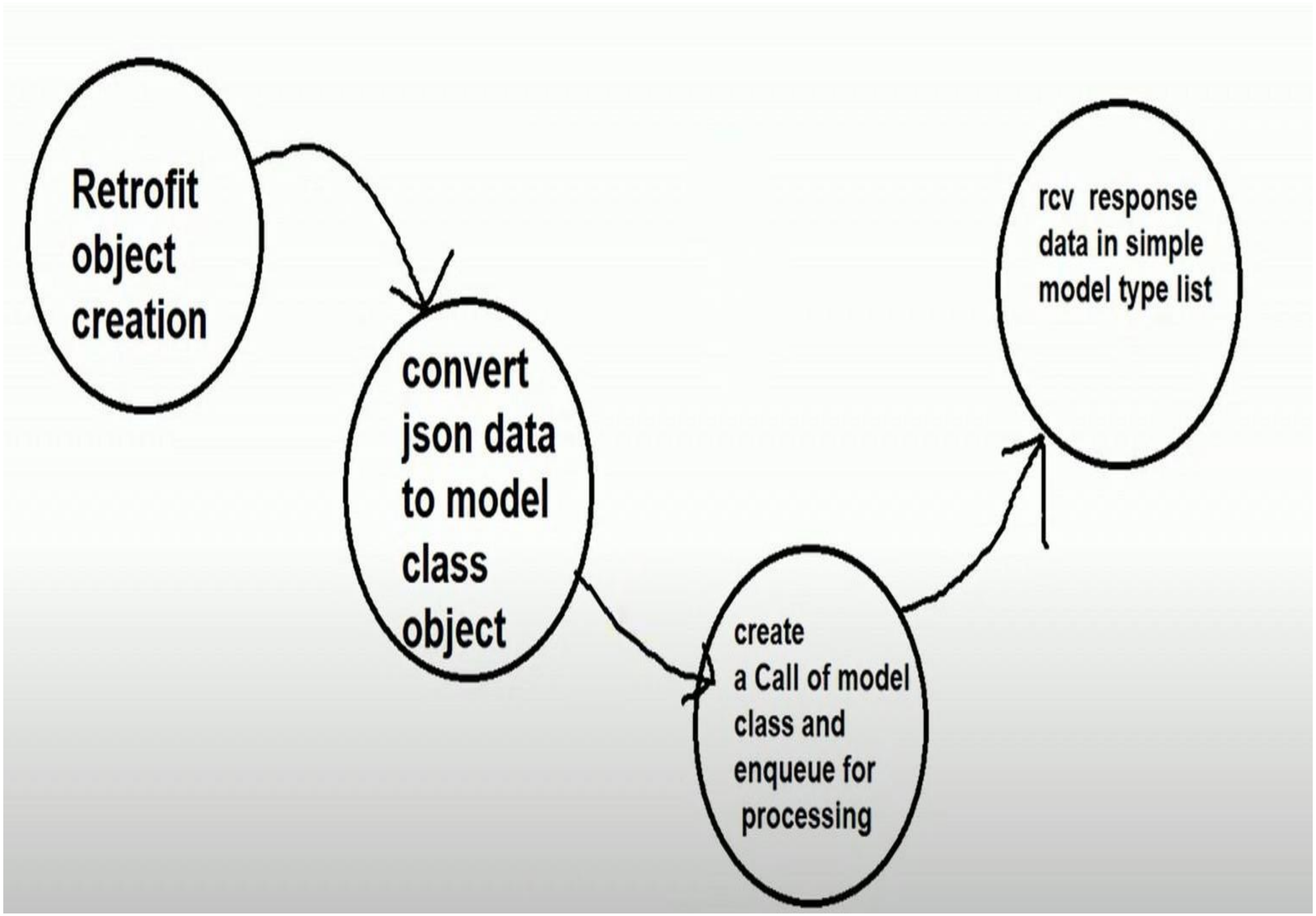
```
compile 'com.squareup.retrofit2:retrofit:2.1.0'

compile  'com.google.code.gson:gson:2.6.2'

compile 'com.squareup.retrofit2:converter gson:2.1.0'
```

- Make sure to add required Internet permission in your AndroidManifest.xml file

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

- In the past, Retrofit relied on the Gson library to serialize and deserialize JSON data.
Retrofit 2 now supports many different parsers for processing network response data, including Moshi, a library built by Square for efficient JSON parsing.

- Gson : com.squareup.retrofit2:converter-gson:2.9.0
- Jackson : com.squareup.retrofit2:converter-jackson:2.9.0
- Moshi : com.squareup.retrofit2:converter-moshi:2.9.0
- Protobuf : com.squareup.retrofit2:converter-protobuf:2.9.0
- Wire : com.squareup.retrofit2:converter-wire:2.9.0
- Simple XML : com.squareup.retrofit2:converter-simplexml:2.9.0


Retrofit object creation
convert json data to model class object
create a Call of model class and enqueue for processing
rcv response data in simple model type list

OkHttp vs Retrofit: What are the differences?

OkHttp

An open source HTTP client.

HTTP is the way modern applications network.

It’s how we exchange data & media. Doing HTTP efficiently makes your stuff load faster and saves bandwidth

OkHttp is a pure HTTP/SPDY client responsible for any low-level network operation, caching, request and response manipulation, and many more.

Retrofit

Retrofit is a high-level REST abstraction build on top of OkHttp.

A type-safe HTTP client for Android and Java.

Retrofit turns your HTTP API into a Java interface.

OkHttp and Retrofit can be primarily classified as "API" tools.

01.
Model Class

which is used as a JSON model (POJO)

02.
Interface

that define the possible HTTP operations

03.
Retrofit Builder

Instance which uses the interface and the Builder API to

Creating the Retrofit instance

To send out network requests to an API, we need to use the Retrofit builder class and specify the base URL for the service.

/ Trailing slash is needed

```
public static final String BASE_URL = "http://api.myservice.com/";
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl(BASE_URL)
    .addConverterFactory(GsonConverterFactory.create())
    .build();
```

Define the Endpoints

With Retrofit 2, endpoints are defined inside of an interface using special retrofit annotations to encode details about the parameters and request method.
In addition, the return value is always a parameterized Call<T> object such as Call<User>.
If you do not need any type-specific response, you can specify return value as simply Call<ResponseBody>.

For instance, the interface defines each endpoint in the following way
```java
public interface MyApiEndpointInterface {
    // Request method and URL specified in the annotation


    @GET("users/{username}")
    Call<User> getUser(@Path("username") String username);

    @GET("group/{id}/users")
    Call<List<User>> groupList(@Path("id") int groupId, @Query("sort") String sort);

    @POST("users/new")
    Call<User> createUser(@Body User user);
}
```

POSTing JSON data(model class)

```
public class User {

  int mId;

  String mName;

  public User(int id, String name ) {
    this.mId = id;
    this.mName = name;
  }
}
```

Invoke API call

```
User user = new User(123, "John Doe");
Call<User> call = apiService.createuser(user);
call.enqueue(new Callback<User>() {
  @Override
  public void onResponse(Call<User> call, Response<User> response) {
  }
  @Override
  public void onFailure(Call<User> call, Throwable t) {
  }
```

Some of the features offered by OkHttp

- HTTP/2 support allows all requests to the same host to share a socket.
- Connection pooling reduces request latency (if HTTP/2 isn't available).
- Transparent GZIP shrinks download sizes.

Key features provided by Retrofit

- Easier troubleshooting
- Expressive code
- Async calls and queues
- Automatic JSON parsing
- Automatic error handling callbacks
- Built-in User authentication support
- URL parameter replacement and query parameter support
- Object conversion to request body (e.g., JSON, protocol buffers)
- Multipart request body and file upload

- Create an android application to demonstrate JSON data parsing using Retrofit (you can use https://api.github.com/users json data).

build.gradle(:app)

```
plugins {
    id("com.android.application")
}

android {
    namespace = "com.example.retrofit"
    compileSdk = 34

    defaultConfig {
        applicationId = "com.example.retrofit"
        minSdk = 24
        targetSdk = 33
        versionCode = 1
        versionName = "1.0"
        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_1_8
        targetCompatibility = JavaVersion.VERSION_1_8
    }
}

dependencies {
    implementation("com.squareup.retrofit2:retrofit:2.7.2")
    implementation("com.squareup.retrofit2:converter-gson:2.7.2")
    implementation("androidx.appcompat:appcompat:1.6.1")
    implementation("com.google.android.material:material:1.10.0")
    implementation("androidx.constraintlayout:constraintlayout:2.1.4")
    testImplementation("junit:junit:4.13.2")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
}
```

Add these two dependencies to build.gradle file

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.jasondataparsingretrofit">
<uses-permission android:name="android.permission.INTERNET"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Jasondataparsingretrofit">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Add internet permission to manifest file

API.java

```java
package com.example.jasondataparsingretrofit;

import java.util.List;

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

public interface API {
    String BASE_URL = "https://api.github.com/";
    @GET("users")
    Call<List<User>> getRecords();

}
```

User.java

```java
package com.example.jasondataparsingretrofit;

public class User {
    String login;
    String node_id;

    public String getLogin() {
        return login;
    }

    public void setLogin(String login) {
        this.login = login;
    }

    public String getNode_id() {
        return node_id;
    }

    public void setNode_id(String node_id) {
        this.node_id = node_id;
    }
}
```

activity_main.xml

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

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

</LinearLayout>
```

MainActivity.java

```java
package com.example.jasondataparsingretrofit;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
import android.widget.Toast;
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 {

    TextView textView;

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

        textView = findViewById(R.id.text_view);

        Retrofit retrofit = new Retrofit.Builder()
                .baseUrl(API.BASE_URL)
                .addConverterFactory(GsonConverterFactory.create()).build();
        API api = retrofit.create(API.class);
        Call<List<User>> call = api.getRecords();
        call.enqueue(new Callback<List<User>>() {
            @Override
            public void onResponse(Call<List<User>> call, Response<List<User>> response) {
                List<User> list = response.body();

                for(int i = 0; i < list.size(); i++) {
                    textView.append(list.get(i).getLogin() + ", " + list.get(i).getNode_id() + "\n");
                }
            }

            @Override
            public void onFailure(Call<List<User>> call, Throwable t) {
                Toast.makeText(getApplicationContext(), "Error | Failed to fetch data!!!",
Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```