

OkHttp

Disadvantages of HttpURLConnection

- Poor readability and less expressive
- Lots of boilerplate code
 - Byte arrays, stream readers
- No built-in support for parsing JSON response
- Manage background threads manually (Doesn't support native asynchronous calls)

OkHttp

- OkHttp is an HTTP and HTTP/2 client for Android and java application for making restful API calls that's **efficient** by default.
- OkHttp is a third party library that was introduced by Square in 2013 to send and receive HTTP-based network requests.
- The client is designed for both blocking synchronous and non blocking asynchronous calls.
- OkHttp android provides an implementation of **HttpURLConnection** and **Apache Client** interfaces by working directly on a top of java Socket without using any extra dependencies.
- OkHttp perseveres when the network is troublesome: it will silently recover from common connection problems.
- If your service has multiple IP addresses, OkHttp will attempt alternate addresses if the first connect fails.
- Using OkHttp is easy.
- Its request/response API is designed with fluent builders and immutability.
- It supports both synchronous blocking calls and async calls with callbacks.
- OkHttp supports modern TLS(Transport Layer Security) features (TLS 1.3, ALPN, certificate pinning).
- It can be configured to fall back for broad connectivity.

a. Setup the http request

`okhttp3.OkHttpClient`

Factory class used for making API calls

Configure caching strategy, read and connection timeouts

Think of this as component which actually does the network call

b. Make the request

`okhttp3.Request`

Actually contains HTTP request information

Url

method (GET, POST, DELETE, PUT/PATCH)

header (Content-type token, authorization

body (JSON payload meant for HTTP request)

c. Read the response

`okhttp3.Callback`

i. Success

`onResponse`

ii. Failure

`onFailure`

Call back interface for handling success and failure cases of API call

`onResponse` has Response containing status code and response payload

`onFailure` has Exception information

OkHttp

Synchronous vs Asynchronous calls

Synchronous means that you call a web service (or function or whatever) and wait until it returns - all other code execution and user interaction is stopped until the call returns.

Asynchronous means that you do not halt all other operations while waiting for the web service call to return. Other code executes and/or the user can continue to interact with the page (or program UI).

OkHttp Synchronous vs Asynchronous calls

- **Synchronous calls**

The **advantages** of synchronous requests are : simple understanding; simple implementation; the response result can be returned for subsequent processing; The **disadvantage** of synchronous request is that it cannot support high concurrency, and it will affect performance in scenarios where real-time request response is not high.

- **Asynchronous Call using AsyncTask**

require an AsyncTask(Android driven) wrapper around it. That means **it doesn't support cancelling a request**. Also, AsyncTasks generally **leak the Activity's context**, which is not preferred.

- **Asynchronous Calling(API driven)** is the recommended way since it supports native cancelling, tagging multiple requests and canceling them all with a single method call (by invoking the cancel on the Activity instance inside the onPause or onDestroy method).

enqueue

Schedules the request to be executed at some point in the future.

The dispatcher defines when the request will run: usually immediately unless there are several other requests currently being executed.

This client will later call back `responseCallback` with either an HTTP response or a failure exception.

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

```
build.gradle(:app)
```

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

```
android {  
    namespace = "com.example.myapplication"  
    compileSdk = 34
```

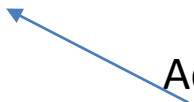
```
    defaultConfig {  
        applicationId = "com.example.myapplication"  
        minSdk = 24  
        targetSdk = 33  
        versionCode = 1  
        versionName = "1.0"  
  
        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"  
    }
```

```
    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.okhttp3:okhttp:5.0.0-alpha.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 this dependency in
build.gradle(:app)file



AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.jasonparsingokhttp">
    <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.Jasonparsingokhttp">
        <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>
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    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">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
    <Button
        android:id="@+id/btnFetch"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="30dp"
        android:layout_gravity="center_horizontal"
        android:text="Fetch data"/>
    <TextView
        android:id="@+id/result_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingHorizontal="16dp" />
    </LinearLayout>

</ScrollView>
```

MainActivity.java

```
package com.example.jasonparsingokhttp;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.telecom.Call;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import java.io.IOException;
import okhttp3.Callback;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;

public class MainActivity extends AppCompatActivity {

    TextView resultView;
    Button fetch;
    OkHttpClient client;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        resultView = findViewById(R.id.result_view);
        fetch=findViewById(R.id.btnFetch);
        client = new OkHttpClient();
        fetch.setOnClickListener(view -> {
            getWebService();
        });
    }
}
```

```

private void getWebService() {
    //String url = "https://api.github.com/users";
    String url = "https://reqres.in/api/users/2";
    Request request = new Request.Builder()
        .url(url).build();
    client.newCall(request).enqueue(new Callback() {
        @Override
        public void onResponse(@NonNull okhttp3.Call call, @NonNull Response
response) throws IOException {
            if(response.isSuccessful()) {
                final String result = response.body().string();
                MainActivity.this.runOnUiThread(() ->
resultView.setText(result));
            }
        }

        @Override
        public void onFailure(@NonNull okhttp3.Call call, @NonNull IOException e)
{
            Toast.makeText(getApplicationContext(), e.getMessage(),
Toast.LENGTH_SHORT).show();
        }
    });
}

```