

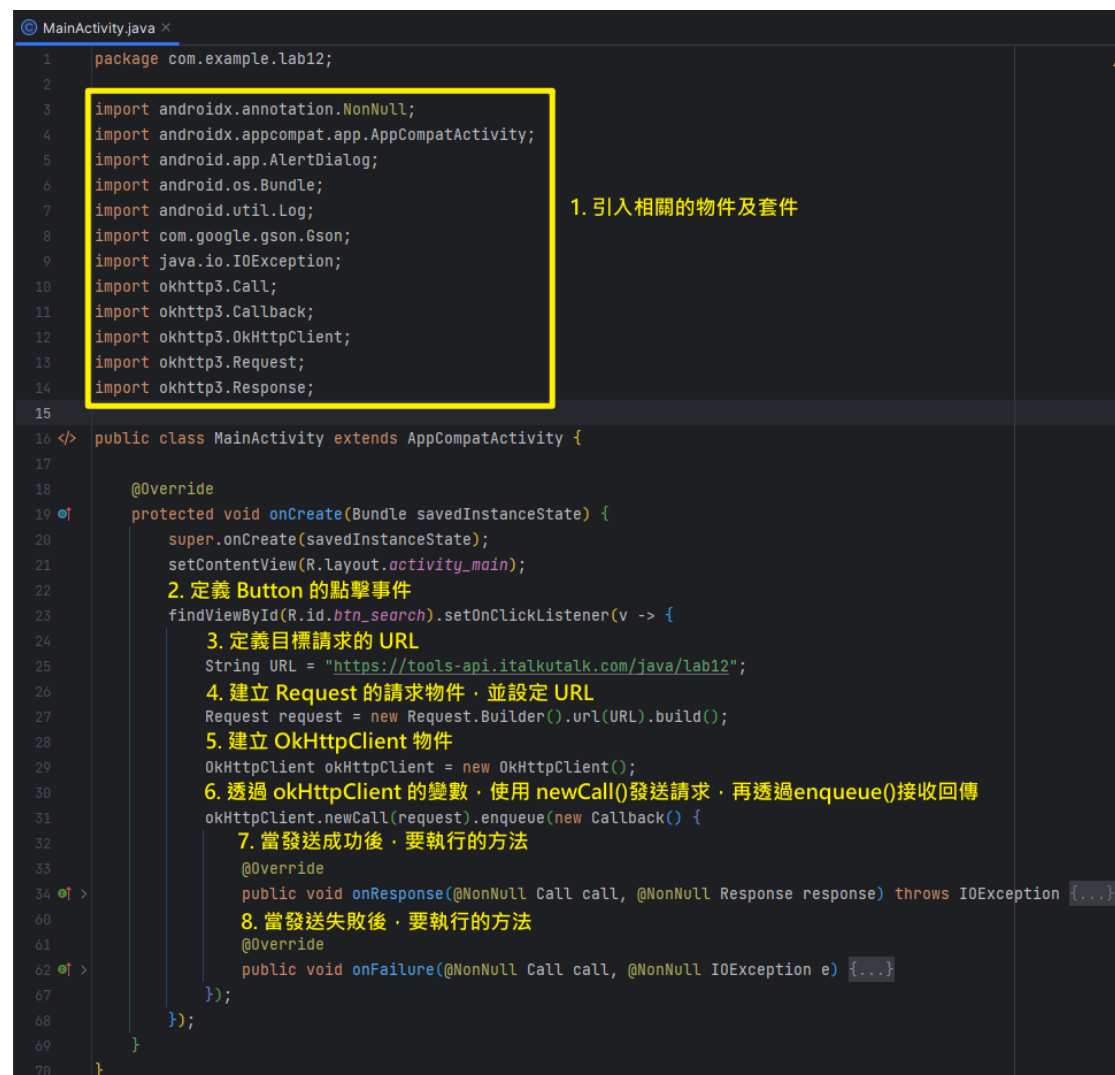
對應的 xml 如下：

```
<?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">

    <Button
        android:id="@+id/btn_search"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="查詢車班"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

**Step8** 編寫查詢按鈕按下後，使用 OkHttpClient 來發出 Http Get Request 至 <https://tools-api.italkutalk.com/java/lab12>，接收到 Response 之後，將 JSON 字串經由 GSON 轉換至 Data 物件之中，再將 Data 物件 轉換放入 AlertDialog 之中。



```
1 package com.example.lab12;
2
3 import androidx.annotation.NonNull;
4 import androidx.appcompat.app.AppCompatActivity;
5 import android.app.AlertDialog;
6 import android.os.Bundle;
7 import android.util.Log;
8 import com.google.gson.Gson;
9 import java.io.IOException;
10 import okhttp3.Call;
11 import okhttp3.Callback;
12 import okhttp3.OkHttpClient;
13 import okhttp3.Request;
14 import okhttp3.Response;
```

1. 引入相關的物件及套件

```
15
16 </> public class MainActivity extends AppCompatActivity {
17
18     @Override
19     protected void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.activity_main);
22         2. 定義 Button 的點擊事件
23         findViewById(R.id.btn_search).setOnClickListener(v -> {
24             3. 定義目標請求的 URL
25             String URL = "https://tools-api.italkutalk.com/java/lab12";
26             4. 建立 Request 的請求物件，並設定 URL
27             Request request = new Request.Builder().url(URL).build();
28             5. 建立 OkHttpClient 物件
29             OkHttpClient okHttpClient = new OkHttpClient();
30             6. 透過 okHttpClient 的變數，使用 newCall() 發送請求，再透過 enqueue() 接收回傳
31             okHttpClient.newCall(request).enqueue(new Callback() {
32                 7. 當發送成功後，要執行的方法
33                 @Override
34                 public void onResponse(@NonNull Call call, @NonNull Response response) throws IOException {...}
35             });
36             8. 當發送失敗後，要執行的方法
37             @Override
38             public void onFailure(@NonNull Call call, @NonNull IOException e) {...}
39         });
40     }
41 }
```

```

okHttpClient.newCall(request).enqueue(new Callback() {
    7. 當發送成功後・要執行的方法
    @Override
    public void onResponse(@NonNull Call call, @NonNull Response response) throws IOException {
        if (response.code() == 200) {判斷回傳的HTTP狀態碼是否為 200・以及回傳是否有值?
            if (response.body() == null) return;

            使用 Gson 轉換伺服器回傳的 Response 資料
            Data data = new Gson().fromJson(response.body().string(), Data.class);
            建立 String Array 來儲存伺服器響應的資料
            final String[] items = new String[data.result.results.length];
            使用迴圈將資料填入String Array中
            for (int i = 0; i < items.length; i++) {
                items[i] = "\n列車即將進入：" + data.result.results[i].Station +
                    "\n列車行駛目的地：" + data.result.results[i].Destination;
            }

            由於API請求屬於複雜任務・所以OkHttp請求會發生在背景執行緒
            所以需要透過runOnUiThread切換執行緒回【主執行緒】
            runOnUiThread() -> {
                new AlertDialog.Builder(context: MainActivity.this)
                    .setTitle("台北捷運列車到站站名")
                    .setItems(items, listener: null)
                    .show(); 使用 Dialog 顯示伺服器回傳的資料
            });
        } else if (!response.isSuccessful()) {
            Log.e(tag: "伺服器錯誤", msg: response.code() + " " + response.message());
        }
        else {
            Log.e(tag: "其他錯誤", msg: response.code() + " " + response.message());
        }
    }
}

8. 當發送失敗後・要執行的方法
@Override
public void onFailure(@NonNull Call call, @NonNull IOException e) {
    if (e.getMessage() != null) {
        Log.e(tag: "查詢失敗", e.getMessage());
    }
}
});

```

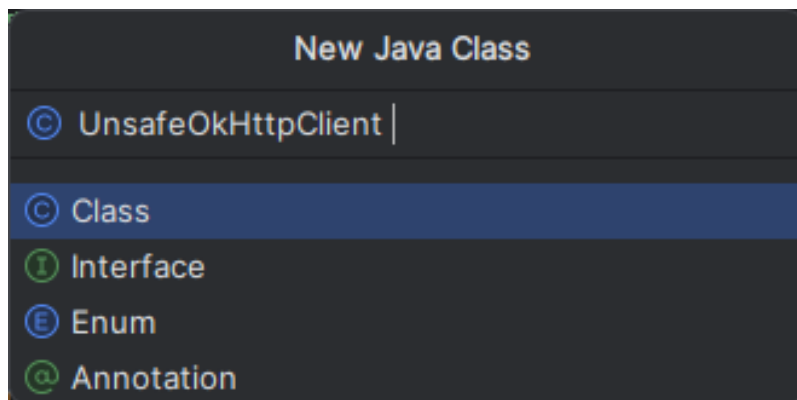
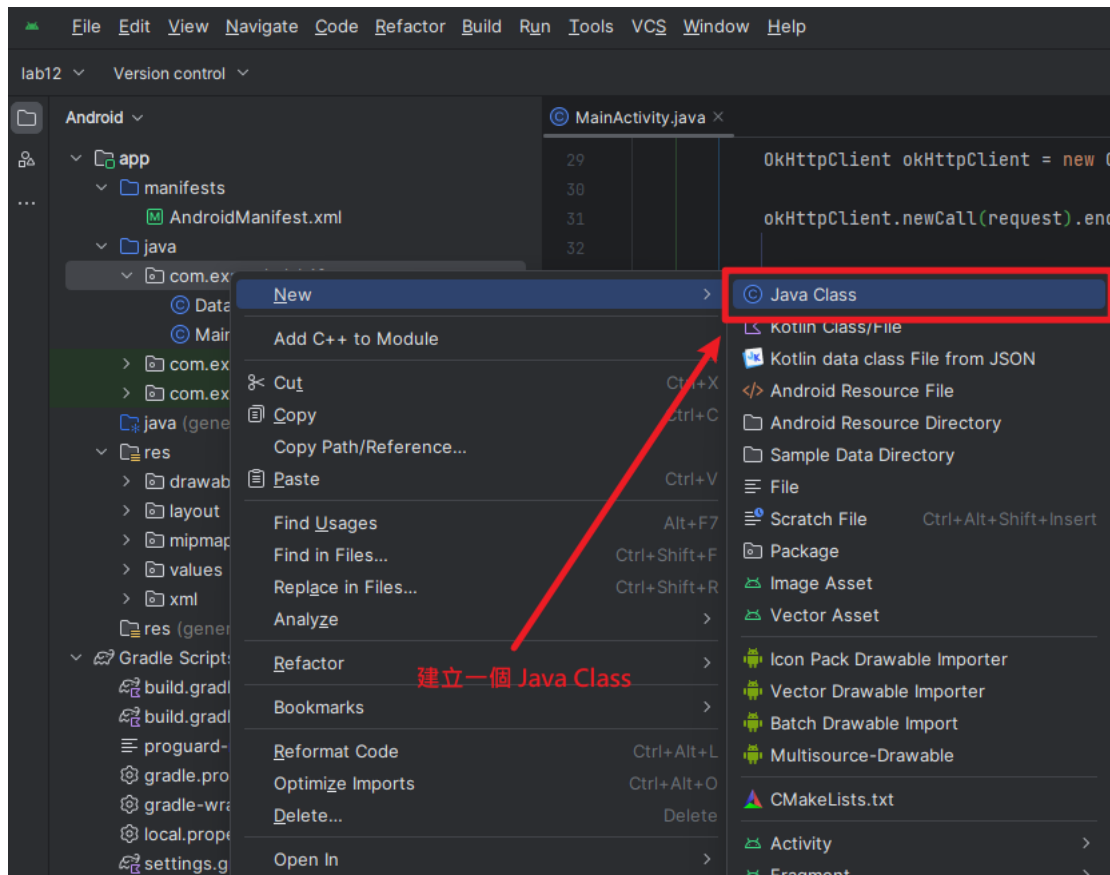
## 助教補充說明：

由於通過 Android 模擬器發送 API 請求，無法通過學校的 SSL 驗證。所以我們需要通過一些特殊方式，使 **OkHttpClient** 物件可以繞過學校的 SSL 驗證。

### 說明

如果使用的是自己的網路環境，就不會出現 SSL 驗證失敗的問題。所以同學們在回家練習的時候，可以不用繞過 SSL 驗證的方式來呼叫 API。

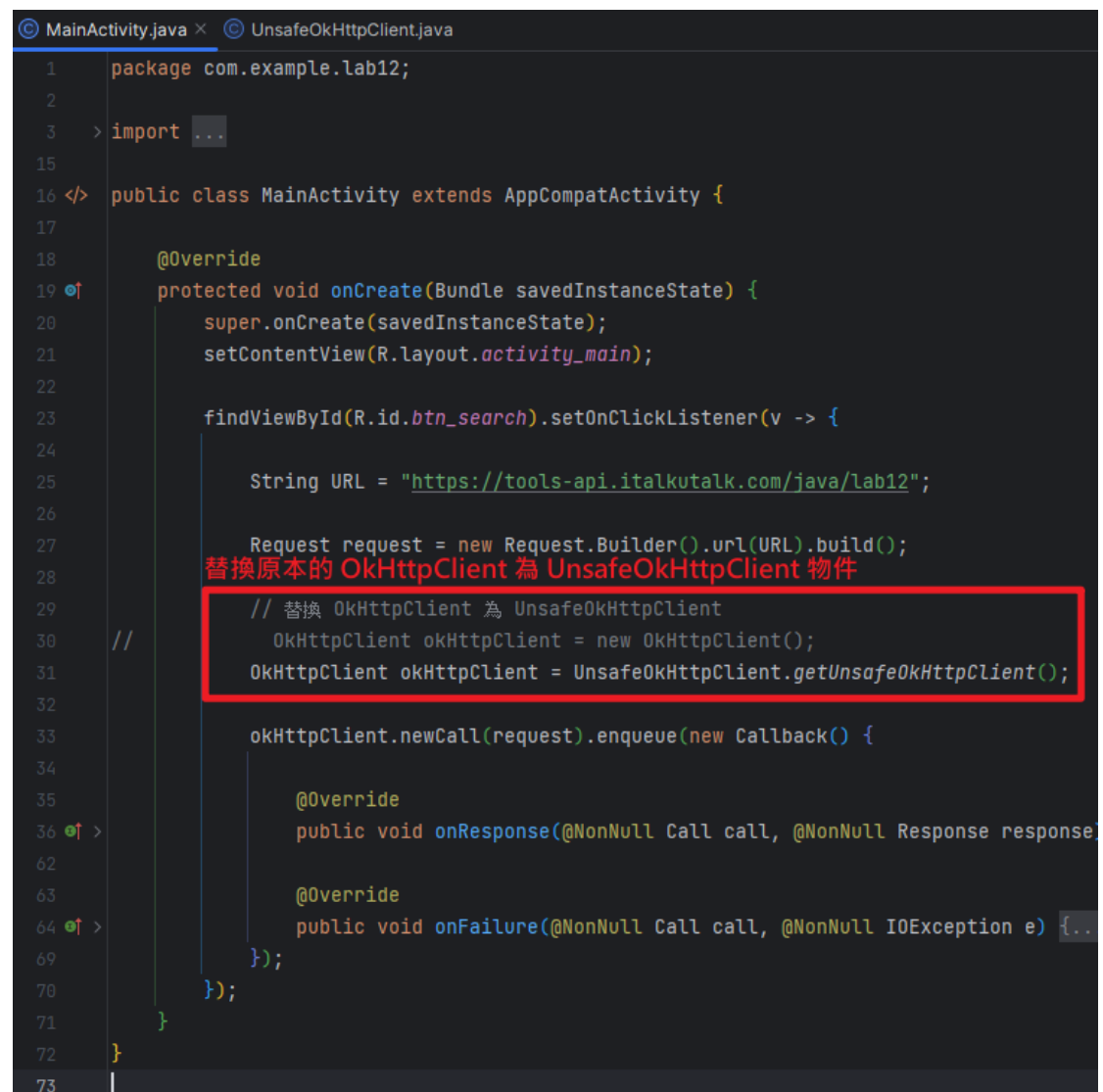
1. 建立一個名為 **UnsafeOkHttpClient** 的 Java 物件。



2. 編寫 `UnsafeOkHttpClient` 的程式碼，裡面包含 `getUnsafeOkHttpClient()` 的靜態方法，用來回傳可以信任所有憑證的 `OkHttpClient` 物件。

```
1 package com.example.lab12;
2
3 import java.security.SecureRandom;
4 import java.security.cert.X509Certificate;
5 import javax.net.ssl.SSLContext;
6 import javax.net.ssl.TrustManager;
7 import javax.net.ssl.X509TrustManager;
8
9 import okhttp3.OkHttpClient;
10
11 public class UnsafeOkHttpClient {
12     取得一個信任所有憑證的 OkHttpClient
13     Returns: OkHttpClient
14
15     @
16     public static OkHttpClient getUnsafeOkHttpClient() {
17         try {
18             // 建立一個信任所有憑證的 X509TrustManager
19             final TrustManager[] trustAllCerts = new TrustManager[]{
20                 new X509TrustManager() {
21                     @Override
22                     public void checkClientTrusted(X509Certificate[] x509Certificates, String s) {
23                         // 不執行任何動作，接受所有用戶端的憑證。
24                     }
25                     @Override
26                     public void checkServerTrusted(X509Certificate[] x509Certificates, String s) {
27                         // 不執行任何動作，接受所有伺服器的憑證。
28                     }
29                     @Override
30                     public X509Certificate[] getAcceptedIssuers() {
31                         // 返回空陣列表示接受所有發行商的憑證。
32                         return new X509Certificate[]{};
33                     }
34                 }
35             };
36
37             // 初始化 SSLContext 物件，並使用剛剛建立的信任所有憑證的 X509TrustManager 初始化。
38             final SSLContext sslContext = SSLContext.getInstance("SSL");
39             sslContext.init(null, trustAllCerts, new SecureRandom());
40             // 建立一個新的 OkHttpClient 並設定 sslSocketFactory 和 hostnameVerifier，以此忽略憑證。
41             // 並將 OkHttpClient 返回。
42             return new OkHttpClient
43                 .Builder()
44                 .sslSocketFactory(sslContext.getSocketFactory(), (X509TrustManager) trustAllCerts[0])
45                 .hostnameVerifier((hostname, session) -> true)
46                 .build();
47         } catch (Exception e) {
48             throw new RuntimeException();
49         }
50     }
51 }
52
```

### 3. 使用 `UnsafeOkHttpClient` 建立一個新的 `OkHttpClient` 物件。



```
1 package com.example.lab12;
2
3 > import ...
15
16 </> public class MainActivity extends AppCompatActivity {
17
18     @Override
19     protected void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.activity_main);
22
23         findViewById(R.id.btn_search).setOnClickListener(v -> {
24
25             String URL = "https://tools-api.italkutalk.com/java/lab12";
26
27             Request request = new Request.Builder().url(URL).build();
28             替換原本的 OkHttpClient 為 UnsafeOkHttpClient 物件
29             // 替換 OkHttpClient 為 UnsafeOkHttpClient
30             OkHttpClient okHttpClient = new OkHttpClient();
31             OkHttpClient okHttpClient = UnsafeOkHttpClient.getUnsafeOkHttpClient();
32
33             okHttpClient.newCall(request).enqueue(new Callback() {
34
35                 @Override
36                 public void onResponse(@NonNull Call call, @NonNull Response response) {
37
38                 }
39
40                 @Override
41                 public void onFailure(@NonNull Call call, @NonNull IOException e) {
42
43                 }
44             });
45         });
46     }
47 }
```