

Lab12

API

本節目的：

- 了解 Http Get 與 Post 觀念
- 了解 JSON 觀念
- 使用 GSON 轉換 JSON 與 String。
- 了解透過 OkHttpClient 來進行網路請求。

12.1 觀念說明：

在 Android 中想要透過網路取得資料，或是與他人交換訊息，就需要使用 Http 通訊機制，Http 通訊協定被廣泛應用於網路環境，除了常見的瀏覽器外，Android 應用程式也能使用 Http 協定存取伺服器的資料。

12.1.1 Http 觀念

Http(Hypertext Transfer Protocol)是一種網路通訊協定，用於客戶端發出請求(Request)給伺服器，伺服器將要給予的資料回傳(Response)給客戶端使用，一般我們稱之為 API(Application Programming Interface)，如下圖 1 手機送出 Request 與伺服器建立連線，伺服器回傳 Response 資料，資料通常為 Xml 或 JSON...等格式的字串，需要再透過翻譯器轉換成 Android 看得懂的資料型態。

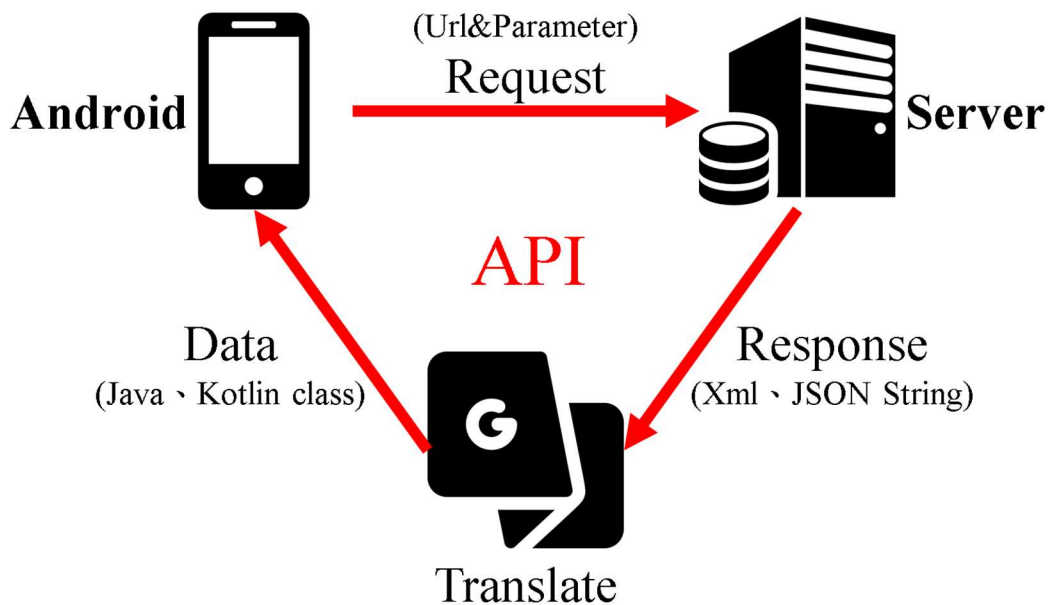


圖 1、API 示意圖

Http 通訊協定中常用的方法(HTTP methods)有 Http Get 與 Http Post 兩種，用信件來比喻的話，信封的格式就是 HTTP，信封外的內容為 http-header，信封內的書信為 message-body，那麼 HTTP Method 就是你要告訴郵差的寄信規則，Get 就像廣告信，參數直接附加於網址後面，人人都看得到，而 Post 就是私人信件，裡面的參數只有寄信者與收信者能看到，相對的安全性也較高。

● Http Get

當我們要從網路取得資料時，就需要從客戶端發出 Http Get Request 跟伺服器建立連線，Get Request 請求的參數會附在 URL 之後(就是直接加在網址後面)，網址以「?」分割 URL 和傳輸參數，參數之間以「&」相連，如下網址所示 name1 和 name2 是參數的名稱，value1 和 value2 是參數的值，傳送的連結如下，黑字的部分為網址，「?」前是 URL，「?」後是要傳輸的參數.多個在 and 之後便可加上我們的參數，並以「&」做區別：

```
http://網址?name1=value1&name2=value2
```

Get 可以直接使用瀏覽器顯示。圖 2 我們以瀏覽器測試以下網址，從伺服器撈出 postId 為 1 的所有文章資料：

```
https://jsonplaceholder.typicode.com/comments?postId=1
```



圖 2、Http Get 示範

執行之後，瀏覽器的顯示區會出現一組字串，此字串即為伺服器回傳的資料，並採用 JSON 字串格式表示，有關 JSON 字串格式會在下一節做說明。

Get 使用非常簡單方便，不過也有其缺點：

- 由於 Get 是直接在網址後加上參數，任何人都能看到。因此不該使用 Get

Request 傳送重要的資料。

- Get Request 有長度限制。

● Http Post

不同於 Get 直接在網址上寫上要傳送的資料，Post 將要傳送給伺服器的資料用 body 另外包起來，隨著 request 一起送給伺服器，如下圖 3 採用 <https://gurujsonrpc.appspot.com> 網頁來測試 Post。

圖 3 例子中，採用 Post 對 <https://gurujsonrpc.appspot.com/guru> 做請求，搜尋資料的方法為 "guru.test"，params 為 "Guru" 字串陣列且 id 為 123 的資料。

1. Enter the service URL and the JSON request string to get started.

Service URL
 要求的 URL

Request JSON String
 Request 的資料

Call Method ⓘ Use the HTTP POST method when submitting data.

圖 3、Http Post Request

執行之後，伺服器可以接收到客戶端發送過去的資料，經由伺服器執行相關處理後會 Response 一組 JSON 字串資料，如下圖 4 所示 id 為 123 且 result 帶有 Guru 的資料。

2. The response is shown here.

Raw Response JSON String
 伺服器 Response 的資料

圖 4、Response

說明

理解了 Get 與 Post 的觀念之後，本教學中會教導在 Android 上採用 JSON 字串格式，透過 OkHttp 來與後台溝通。

12.1.2 JSON 觀念

HTTP 的操作上，傳遞的資料必須是字串形式，因此為了讓資料能透過單一字串送出多組的資料，就必須要設計一套標準化的格式，此教學中會教導使用的是 JSON 字串格式。


JSON 是個以純文字為基底去儲存和傳送簡單結構資料，你可以透過特定的格式去儲存任何資料(字串、數字、陣列、物件)，也可以透過物件或陣列來傳送較複雜的資料。

JSON 字串採用 name : value 的表示方式，name 表示這個變數的名稱，而 value 表示其內容。物件或陣列的 value 值的表示方式如下：

整數或浮點數	直接寫入數值
字串	加上""
布林函數 (boolean)	true 或 false
陣列	加上[]
物件	加上{ }

如下方所示，左邊的類別結構以 JSON 來表示後，會呈現圖 5 的結果，變數名稱以字串的方式表示，並以冒號連接變數內容，陣列物件會以一個大括弧包含所有成員，而布林、整數與字串則是直接顯示數值。

```
int[] myArray = {1, 2, 3};
boolean myBoolean = true;
int myNumber = 123;
String myString = "abc";
String[] myStringArray = {"a",
    "b", "c"};
```



```
1 {
2   "myArray": [
3     1,
4     2,
5     3
6   ],
7   "myBoolean": true,
8   "myNumber": 123,
9   "myString": "abc",
10  "myStringArray": [
11    "a",
12    "b",
13    "c"
14  ]
15 }
```

圖 5、JSON 資料結構

12.1.3 GSON

圖 1 手機透過 API 與伺服器溝通,由於 Http 連線存在資料格式的問題,客戶端看不懂伺服器的資料型態,而伺服器看不懂客戶端的類別,因此在送出 Http Request 前,我們需要將程式的物件轉成 JSON 字串,這動作叫做序列化(Serialization),而接收 Http Response 後,需要將 JSON 字串轉成程式的物件,這動作叫做反序列化(Deserialization)。Google 提供一個開源 library **GSON** 可以快速的處理物件與 JSON 格式轉換。

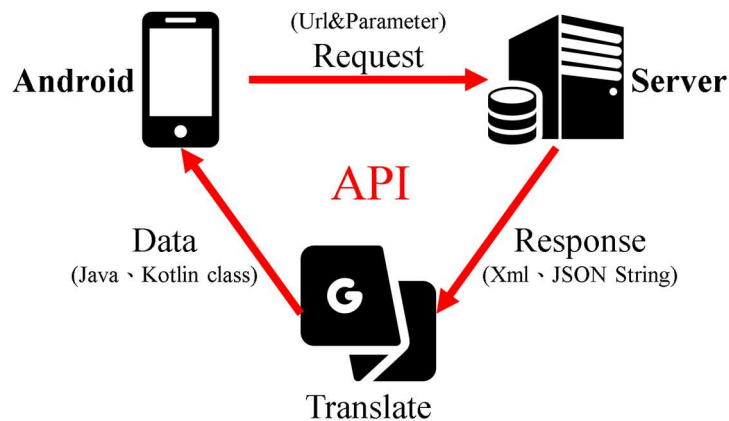


圖 1、API 示意圖

要使用 GSON, 首先我們要引入 GSON 的 library, 下圖 6 在 gradle 的 dependencies 中, 我們加入 'com.google.code.gson:gson:2.8.5' 後讓系統將其匯入。

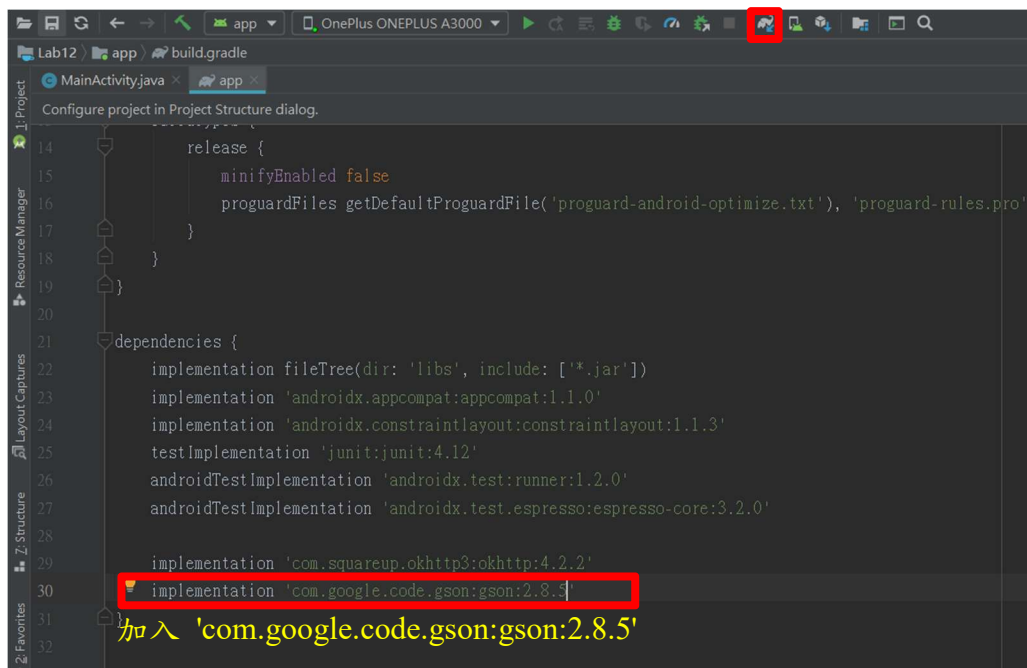


圖 6、加入 GSON library

- 序列化(把物件轉成 JSON 字串)

- 1) 在轉換之前，我們要先設計好一個要被轉換的 Data 類別，來接收來自伺服器的資料：

```
class Data{  
    int myNumber;  
    String myString;  
}
```

- 2) 要透過 GSON 把該物件轉成 JSON 字串有兩個步驟：

```
Step1：建立一個物件，放入資料至物件中  
Data data = new Data();  
data.myNumber = 123;  
data.myString = "abc";  
Step2：使用 Gson().toJson()把物件轉成 JSON 字串  
String json = new Gson().toJson(data);
```

- 3) 輸出字串後可以看到圖 7 的字串結果：

```
json: {"myNumber":123,"myString":"abc"}
```

圖 7、JSON 字串

- 反序列化(把 JSON 字串轉成物件)

- 1) 要透過 GSON 把 JSON 字串轉成物件有兩個步驟：

```
Step1：準備一個 JSON 字串  
String json = "{\"myNumber\":456,\"myString\":\"efg\"}";  
Step2：使用 Gson().fromJson()把 json 字串以 Data 格式做轉換並輸出  
Data data = new Gson().fromJson(json, Data.class);
```

- 2) 把 Data 物件中 myNumber 與 myString 輸出後的結果如下圖 8：

```
myNumber: 456  
myString: efg
```

圖 8、JSON 資料

12.1.4 OkHttp

OkHttp 是 Square 出產的一個 Open source project，是一個 Http 連線的第三方 library，使用它快速實作 Http Get/Post 資料交換的工作，讓 HTTP 連線的過程更加有效率，能避免人為的錯誤設計，加快程式執行的速度。

圖 7 要使用 OkHttp，首先我們要引入 OkHttp 的 library，在 gradle 的 dependencies 中，我們加入 implementation 'com.squareup.okhttp3:okhttp:4.2.2'，然後讓系統將其匯入。

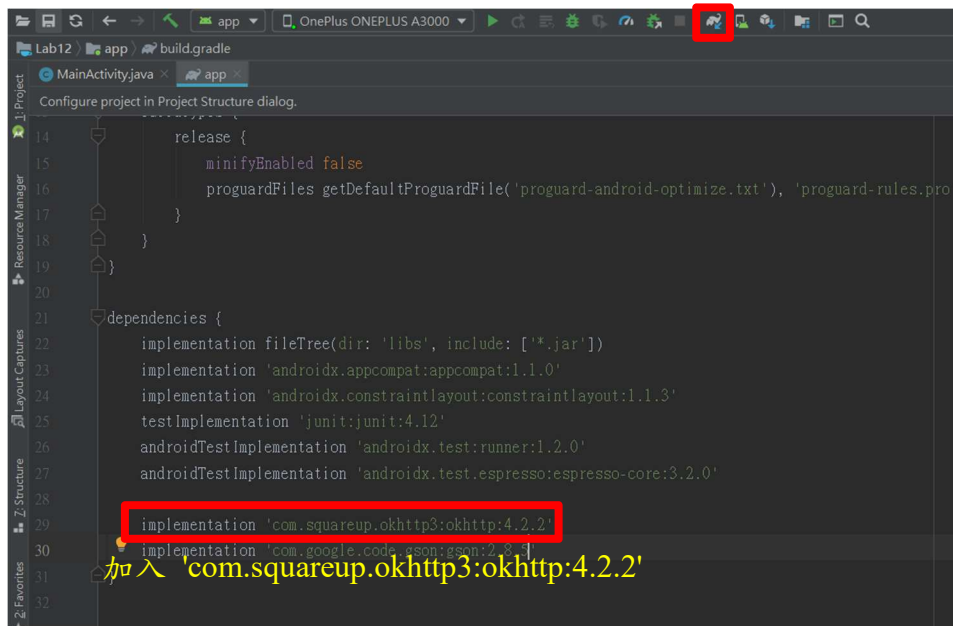
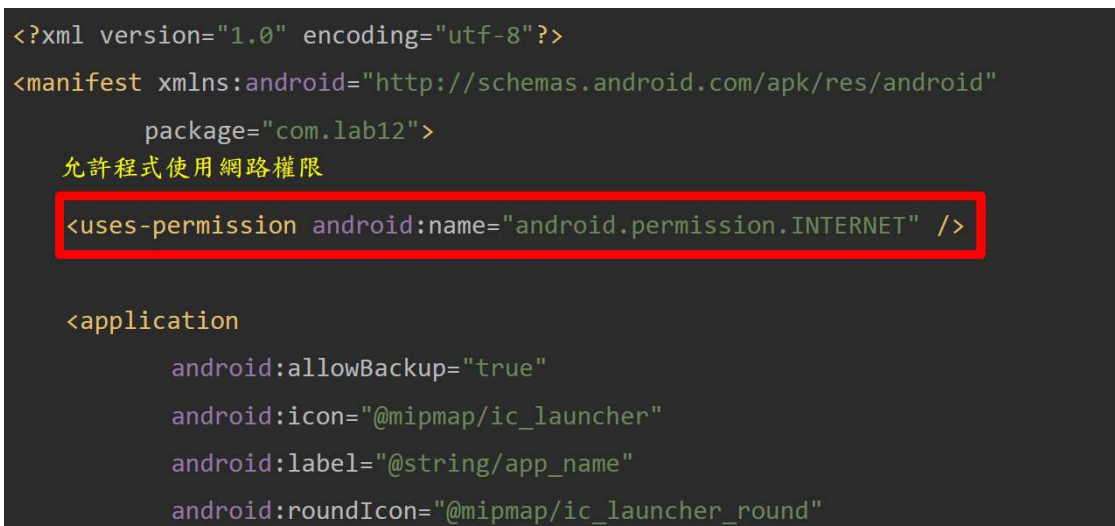


圖 9、加入 OKHttp library

接著，由於我們要讓 Android 進行網路服務，因此需要在 AndroidManifest.xml 加入網路連線的權限。



● Http Get

OkHttp 使用 Get Request 有以下步驟：

Step1：建立一個 Request 物件，並使用 url()方法加入 URL

```
Request req = new Request.Builder()
    .url("https://jsonplaceholder.typicode.com/comments?postId=1").build();
```

Step2：建立 OkHttpClient 物件，newCall()送出請求，enqueue()接收回傳

```
new OkHttpClient().newCall(req).enqueue(new Callback() {
    發送失敗執行此方法
    @Override
    public void onFailure(Call call, IOException e) {
    }

    發送成功執行此方法
    @Override
    public void onResponse(Call call, Response response) throws
IOException {
        Step3：用 response.body().string()取得 Json 字串
        String json = response.body().string();
    }
});
```

以上就是發送一個 get 請求的步驟，首先建立一個 Request 物件，參數要有 url 來設定網址或是 Get Request。

然後通過 request 的對象去產生得到一個 Call 物件，類似於將 Request 封裝成了任務，既然是任務，就會有 execute()和 cancel()等方法。

最後採用非同步執行方式去執行，這裡使用 newCall().enqueue 然後等待任務執行完成，我們便可在 Callback 中即可得到結果。

● Http Post

OkHttp 使用 Post Request 有以下步驟：

```
Step1：建立一個 RequestBody 物件，設定 Request 的參數
RequestBody body = RequestBody.create(
    "{\"method\":\"guru.test\",\"params\":[\"Guru\"],\"id\":123}",
    MediaType.parse("application/json; charset=utf-8"));

Step2：建立一個 Request 物件，加入 URL 與 RequestBody
Request request = new Request.Builder()
    .url("https://gurujsonrpc.appspot.com/guru")
    .post(body).build();

Step3：建立 OkHttpClient 物件，使用 newCall()送出 request，enqueue()接收結果
new OkHttpClient().newCall(req).enqueue(new Callback() {
    發送失敗執行此方法
    @Override
    public void onFailure(Call call, IOException e) {
    }
    發送成功執行此方法
    @Override
    public void onResponse(Call call, Response response) throws
IOException {
        Step4：用 response.body().string()取得 Json 字串
        String json = response.body().string();
    }
});
```

說明

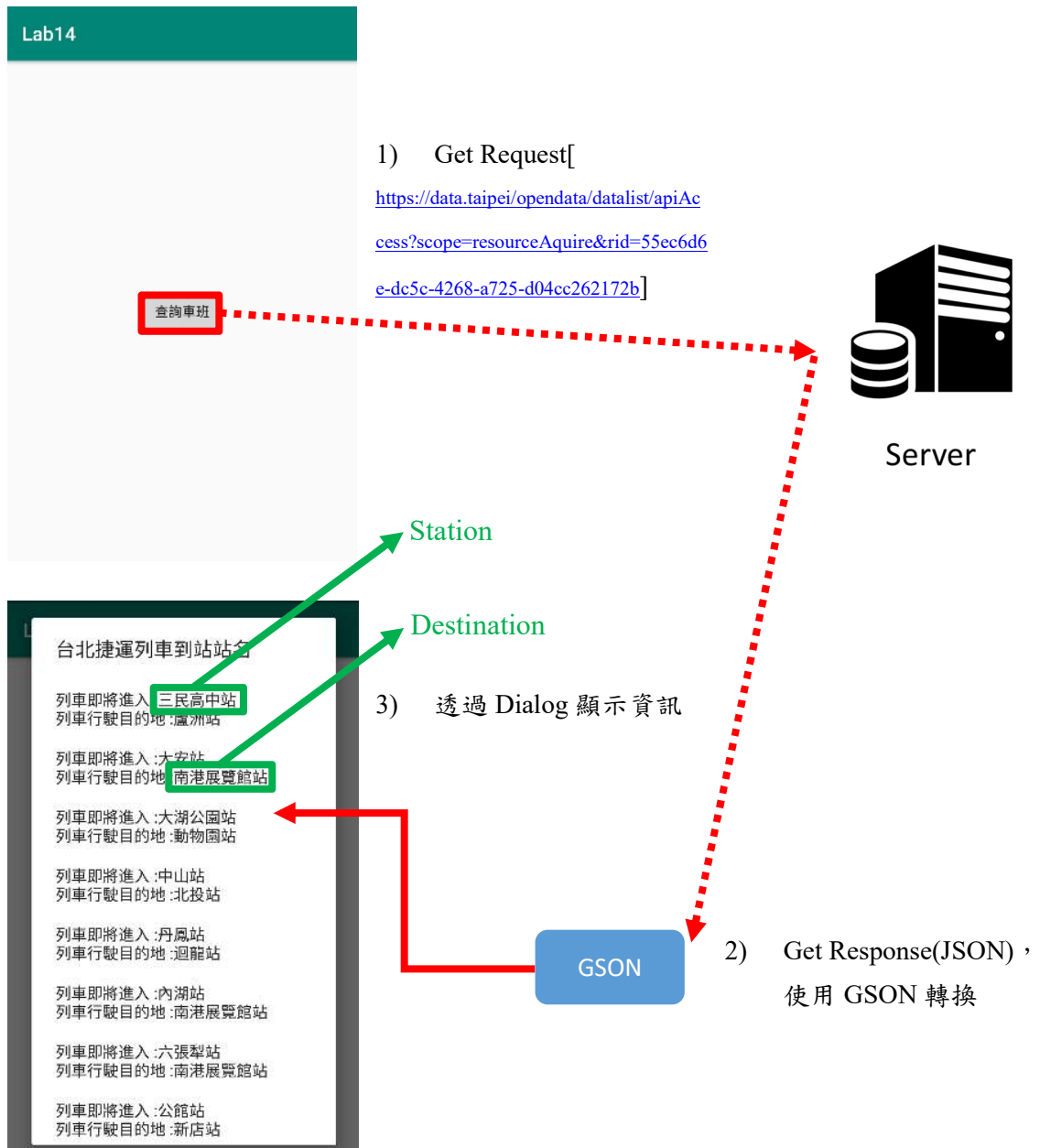
Http Post 與 Get 唯一的差別是 Request 物件需要多一個 post()參數，post()中我們要加入 RequestBody 物件，RequestBody 用來封裝資料(第一個參數)，以及 Body 的格式(第二個參數)。由於我們要採用 JSON 的格式傳送，因此格式部分要加入"application/json; charset=utf-8"，資料部分則加入要傳送過去的 JSON 字串。

12.2 設計重點：

- 練習使用 Http Get 取得“臺北捷運列車到站站名” API 資料，流程如下：

API 連結：<https://data.taipei/opendata/datalist/apiAccess?scope=resourceAquire&rid=55ec6d6e-dc5c-4268-a725-d04cc262172b>

- 1) 按下按鈕後，送出 Http Get 取得 API response(JSON 字串)
- 2) 使用 GSON 將 Response 的 JSON 字串轉換成物件
- 3) 從物件中取得 Station(列車進入月台車站站名)與 Destination(行駛目的地)資訊，並使用 AlertDialog 顯示



12.3 設計步驟：

Step1 建立新專案，以及下圖 10 對應的 class 和 xml 檔：

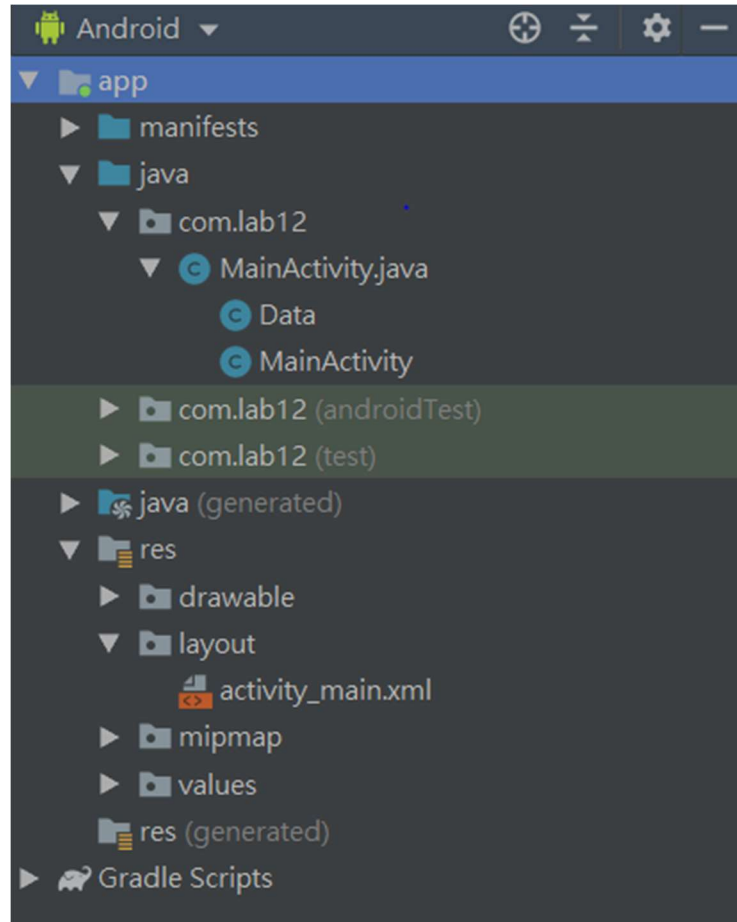


圖 10、專案架構

Step2 繪製 activity_main.xml 如圖 11 所示：



圖 11-1、預覽畫面



圖 11-2、元件樹

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_query"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="查詢車班"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Step3 圖 12 在 gradle 的 dependencies 中引入 Gson 與 OkHttp 的 library，加入 'com.google.code.gson:gson:2.8.5' 與 'com.squareup.okhttp3:okhttp:4.2.2'，然後讓系統將其匯入。

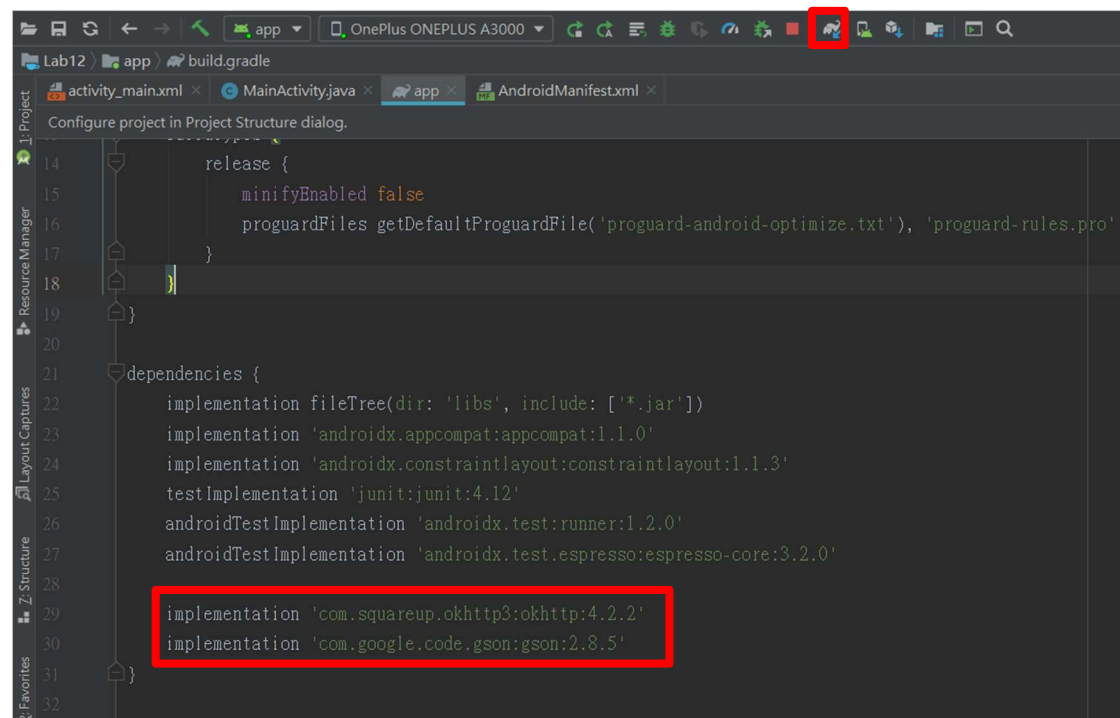


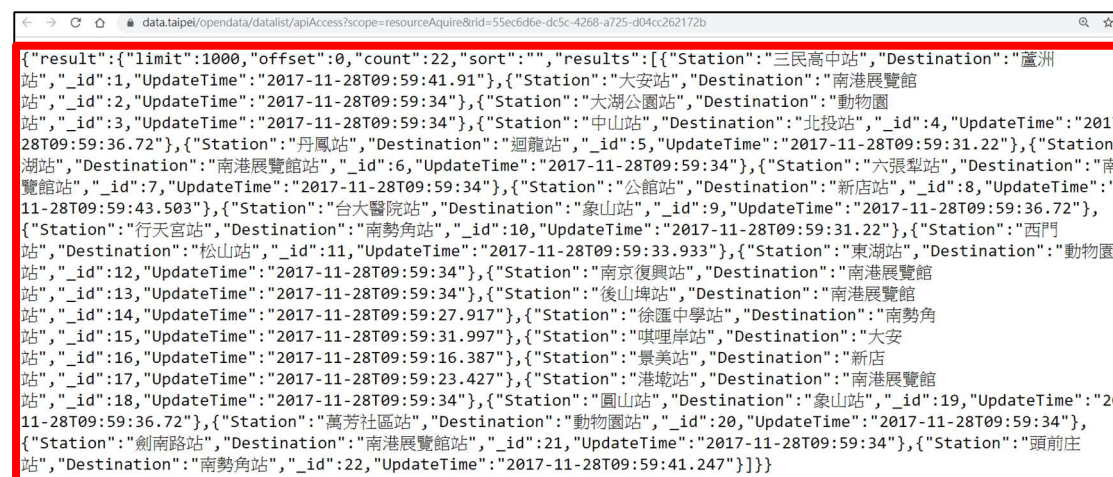
圖 12、加入 GSON 與 OKHttp library

Step4 在 AndroidManifest.xml 加入網路連線的權限。



Step5 圖 13 先以瀏覽器測試 Get

[<https://data.taipei/opendata/datalist/apiAccess?scope=resourceAquire&rid=55ec6d6e-dc5c-4268-a725-d04cc262172b>]是否能成功，同時得到 Response。



```
{
  "result": {
    "limit": 1000,
    "offset": 0,
    "count": 22,
    "sort": "",
    "results": [
      {
        "Station": "三民高中站",
        "Destination": "蘆洲站",
        "_id": 1,
        "UpdateTime": "2017-11-28T09:59:41.91"
      },
      {
        "Station": "大安站",
        "Destination": "南港展覽館站",
        "_id": 2,
        "UpdateTime": "2017-11-28T09:59:34"
      },
      {
        "Station": "大湖公園站",
        "Destination": "動物園站",
        "_id": 3,
        "UpdateTime": "2017-11-28T09:59:34"
      },
      {
        "Station": "中山站",
        "Destination": "北投站",
        "_id": 4,
        "UpdateTime": "2017-11-28T09:59:36.72"
      },
      {
        "Station": "丹鳳站",
        "Destination": "迴龍站",
        "_id": 5,
        "UpdateTime": "2017-11-28T09:59:31.22"
      },
      {
        "Station": "內湖站",
        "Destination": "南港展覽館站",
        "_id": 6,
        "UpdateTime": "2017-11-28T09:59:34"
      },
      {
        "Station": "六張犁站",
        "Destination": "南港展覽館站",
        "_id": 7,
        "UpdateTime": "2017-11-28T09:59:34"
      },
      {
        "Station": "公館站",
        "Destination": "新店站",
        "_id": 8,
        "UpdateTime": "2017-11-28T09:59:43.503"
      },
      {
        "Station": "台大醫院站",
        "Destination": "象山站",
        "_id": 9,
        "UpdateTime": "2017-11-28T09:59:36.72"
      },
      {
        "Station": "行天宮站",
        "Destination": "南勢角站",
        "_id": 10,
        "UpdateTime": "2017-11-28T09:59:31.22"
      },
      {
        "Station": "西門站",
        "Destination": "東湖站",
        "_id": 11,
        "UpdateTime": "2017-11-28T09:59:33.933"
      },
      {
        "Station": "松山站",
        "Destination": "動物園站",
        "_id": 12,
        "UpdateTime": "2017-11-28T09:59:34"
      },
      {
        "Station": "南京復興站",
        "Destination": "南港展覽館站",
        "_id": 13,
        "UpdateTime": "2017-11-28T09:59:34"
      },
      {
        "Station": "後山埤站",
        "Destination": "南港展覽館站",
        "_id": 14,
        "UpdateTime": "2017-11-28T09:59:27.917"
      },
      {
        "Station": "徐匯中學站",
        "Destination": "南勢角站",
        "_id": 15,
        "UpdateTime": "2017-11-28T09:59:31.997"
      },
      {
        "Station": "噶哩岸站",
        "Destination": "大安站",
        "_id": 16,
        "UpdateTime": "2017-11-28T09:59:16.387"
      },
      {
        "Station": "景美站",
        "Destination": "新店站",
        "_id": 17,
        "UpdateTime": "2017-11-28T09:59:23.427"
      },
      {
        "Station": "港墘站",
        "Destination": "南港展覽館站",
        "_id": 18,
        "UpdateTime": "2017-11-28T09:59:34"
      },
      {
        "Station": "圓山站",
        "Destination": "象山站",
        "_id": 19,
        "UpdateTime": "2017-11-28T09:59:36.72"
      },
      {
        "Station": "萬芳社區站",
        "Destination": "動物園站",
        "_id": 20,
        "UpdateTime": "2017-11-28T09:59:34"
      },
      {
        "Station": "劍南路站",
        "Destination": "南港展覽館站",
        "_id": 21,
        "UpdateTime": "2017-11-28T09:59:34"
      },
      {
        "Station": "頭前庄站",
        "Destination": "南勢角站",
        "_id": 22,
        "UpdateTime": "2017-11-28T09:59:41.247"
      }
    ]
  }
}
```

圖 13、測試 Http GET

Response

Step6 圖 14 使用這網頁：<https://jsoneditoronline.org/>，將 Response 做排版，然後可以得到右邊的資料結構。

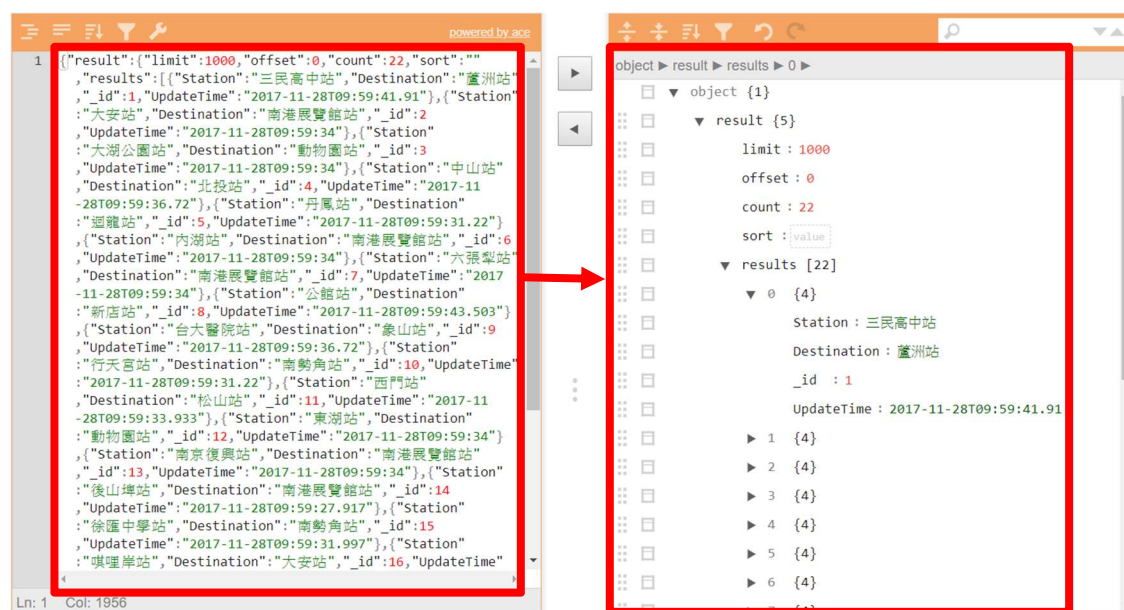
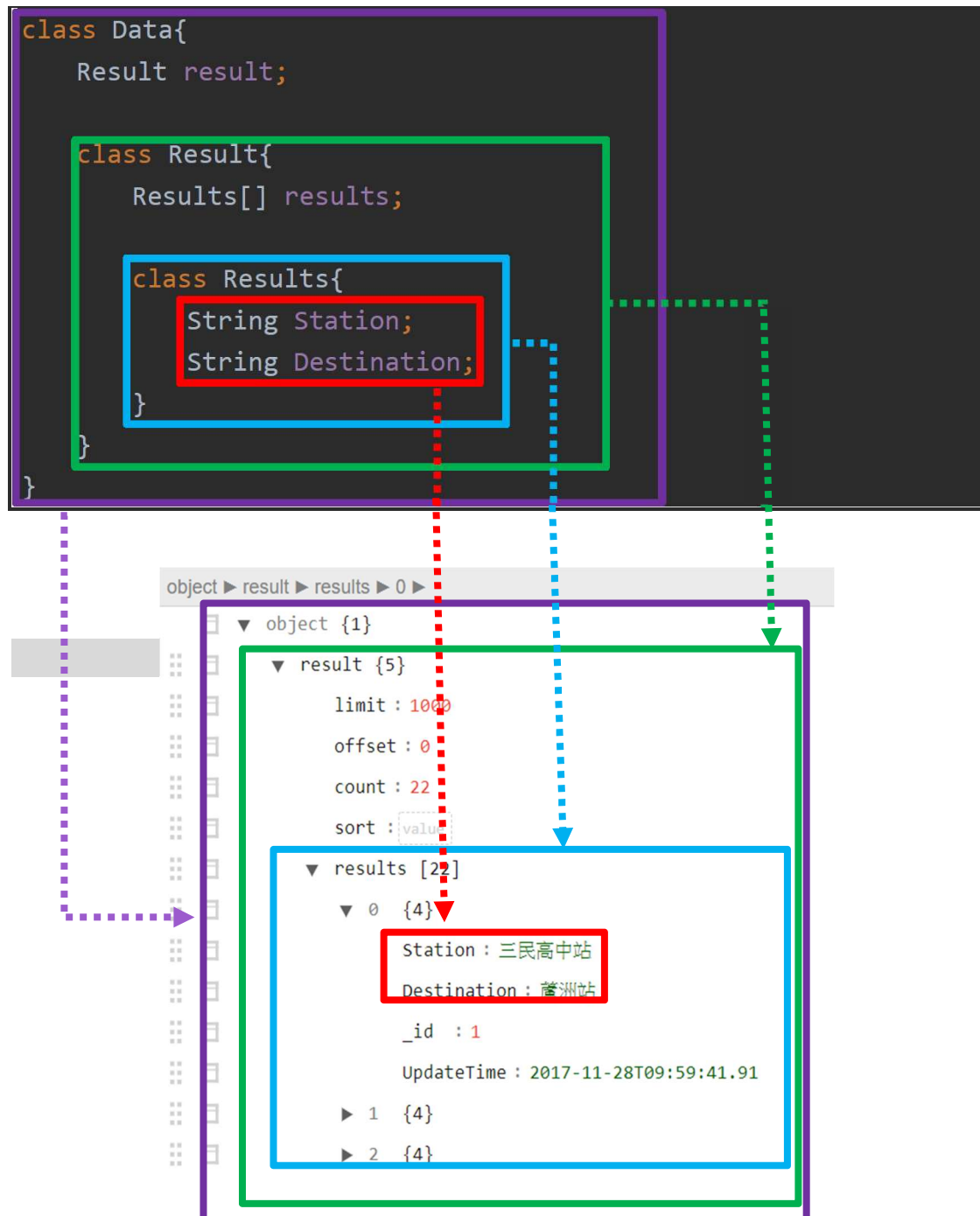


圖 14、排版後的資料

Step7 根據 Step6 資料結構的 JSON，在 MainActivity 中客製化 Data 類別，該資料結構必須依照 Response 來做設計，因此要比對網頁的 Response 結構去規劃類別內容。由於我們只需要 Station 與 Destination 的資訊，因此只需要提取兩者變數即可。



Step8 編寫查詢按鈕按下後，使用 OkHttpClient 來發出 Http Get Request 至 <https://data.taipei/opendata/datalist/apiAccess?scope=resourceAquire&rid=55ec6d6e-dc5c-4268-a725-d04cc262172b>。

```
findViewById(R.id.btn_query).setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View v) {
        建立一個 Request 物件，並使用 url()方法加入 URL
        Request req = new Request.Builder()
            .url("https://data.taipei/opendata/datalist/apiA
ccess?scope=resourceAquire&rid=55ec6d6e-dc5c-4268-a725-
d04cc262172b").build();
        建立 okHttpClient 物件，newCall()送出請求，enqueue()接收回傳
        new OkHttpClient().newCall(req).enqueue(new Callback()
{
    發送失敗執行此方法
    @Override
    public void onFailure(Call call, IOException e) {
        Log.e("查詢失敗",e.toString());
    }
}
```

Step9 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

```
發送成功則執行此方法
@Override
public void onResponse(@Nullable Call call, @Nullable Response response) throws IOException {
    判斷是否為空值
    if (response == null) return;
    判斷回傳code狀態
    if (response.code() == 200) {
        判斷回傳的body是否有值
        if (response.body() == null) return;
        將回傳的JSON物件以Data格式做轉換
        Data data = new Gson().fromJson(response.body().string(), Data.class);
        建立一個字串陣列，用於題與Station 與 Destination 資訊
        final String[] items = new String[data.result.results.length];
        提取 Data 中的 Station 與 Destination 資訊
        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;
        建立 AlertDialog 物件
        MainActivity.this.runOnUiThread() -> new AlertDialog.Builder( context MainActivity.this)
            .setTitle("台北捷運列車到站站名")
            .setItems(items, listener, null)
            .show();
        判斷錯誤情況
    } 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());
}
});
});
```