

14

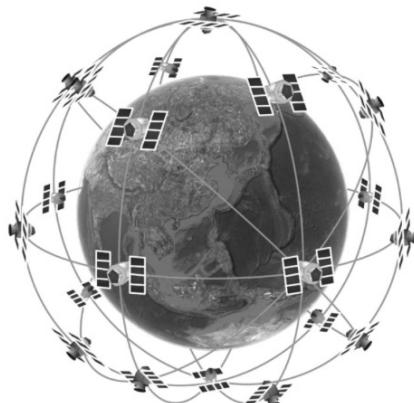
Android GPS、 Google 地圖與導航

- 14-1 取得 GPS 資訊
- 14-2 Google 地圖使用許可權申請與 Hello Map
- 14-3 Hello Map 範例結構與基本使用
- 14-4 ProximityAlert 接近偵測提醒
- 14-5 地圖與街景
- 14-6 地圖與導航

G巨匠電腦



全球定位系統（GPS¹）是一個中距離圓型軌道衛星導航系統。它可以為地球表面絕大部分地區（98%）提供準確的定位、測速和高精度的時間標準。系統由美國國防部研製和維護，可滿足位於全球任何地方或近地空間的軍事用戶連續精確的確定三維位置、三維運動和時間的需要。該系統包括太空中的 24 顆 GPS 衛星；地面上的 1 個主控站、3 個數據注入站和 5 個監測站及做為用戶端的 GPS 接收機。最少只需其中 4 顆衛星，就能迅速確定用戶端在地球上所處的位置及海拔高度；所能收連接到的衛星數越多，解碼出來的位置就越精確。



該系統是由美國政府於 20 世紀 70 年代開始進行研製於 1994 年全面建成。使用者只需擁有 GPS 接收機，無需另外付費。GPS 信號分為民用的標準定位服務（SPS，Standard Positioning Service）和軍規的精確定位服務（PPS，Precise Positioning Service）兩類。由於 SPS 無須任何授權即可任意使用，原本美國因為擔心敵對國家或組織會利用 SPS 對美國發動攻擊，故在民用訊號中人為地加入誤差以降低其精確度，使其最終定位精確度大概在 100 米左右；軍規的精度在十米以下。2000 年以後，柯林頓政府決定取消對民用訊號的干擾。因此，現在民用 GPS 也可以達到十米左右的定位精度。

GPS 系統擁有如下多種優點：全天候，不受任何天氣的影響；全球覆蓋（高達 98%）；三維定速定時高精度；快速、省時、高效率；應用廣泛、多功能；可移動定位；不同於雙星定位系統，使用過程中接收機不需要發出任何信號，增加了隱蔽性並提高了其軍事應用效能。

¹ Global Positioning System 全球定位系統（美系）

- 座標系統 WGS84
- World Geodetic System 1984
- 六軌道 24 顆衛星
- 高度：20,200km
- 演算法：空間後方交會法

GPS 的功能

1. 精確定時：廣泛應用在天文臺、通信系統基站、電視臺中。
2. 工程施工：道路、橋梁、隧道的施工中大量採用 GPS 設備進行工程測量。
3. 勘探測繪：野外勘探及城區規劃中都有用到。
4. 導航：
 - 武器導航：精確制導彈、巡航導彈。
 - 車輛導航：車輛調度、監控系統。
 - 船舶導航：遠洋導航、港口/內河引水。
 - 飛機導航：航線導航、進場著陸控制。
 - 星際導航：衛星軌道定位。
 - 個人導航：個人旅遊及野外探險。
5. 定位：
 - 車輛防盜系統。
 - 手機，PDA，PPC 等通信移動設備防盜，電子地圖，定位系統。
 - 兒童及特殊人群的防走失系統。
 - 精準農業：農機具導航、自動駕駛，土地高精度平整。

GPS 的六大特點

1. 全天候，不受任何天氣的影響。
2. 全球覆蓋（高達 98%）。
3. 三維定點定速定時高精度。
4. 快速、省時、高效率。
5. 應用廣泛、多功能。
6. 可移動定位。

目前正在運行的全球衛星定位系統有美國的 GPS 系統和俄羅斯的 GLONASS 系統。

歐盟 1999 年初正式推出「伽利略」計劃，部署新一代定位衛星。該方案由 27 顆運行衛星和 3 顆預備衛星組成，可以覆蓋全球，位置精度達幾米，亦可與美國的 GPS 系統兼容，總投資為 35 億歐元。該計劃預計於 2010 年投入運行。另外中國大陸還獨立研製了一個區域性的衛星定位系統——北斗導航系統。該系統的覆蓋範圍限於中國大陸及周邊地區，不能在全球範圍提供服務，主要用於軍事用途。

※ 資料來源：<http://wiki.mbalib.com/zh-tw/全球定位系统>

14-1 取得 GPS 資訊

App 開發者可以透過 Android Framework (框架) 來取得 GPS 封裝服務的資訊，其分類目錄與說明如下：

frameworks/base/location/java/android/location

這裡主要是用來被 App 調用的，API 套件： android.location 。

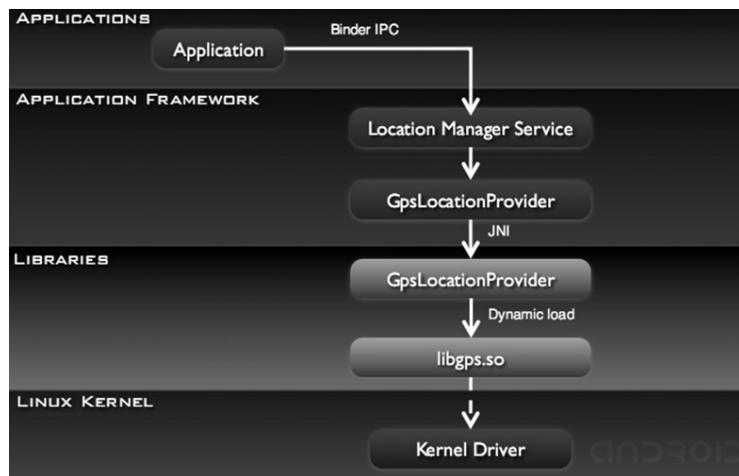
frameworks/base/location/java/com/android/internal/location

這個目錄是 Framework 對 Location 服務的內部實現。

framework\services\java\com\android\server

這個目錄只有一個 java 程式檔 LocationManagerService.java 它是 Location 服務對內部實現的一種封裝。

GPS 啟動過程圖：



Google I/O GPS 啓動過程圖

➔ 範例程式（Map/app_locationgps）相關 Java API

可以藉由 `android.location` 套件下取得 GPS 位置資訊的類別或介面：

`public class LocationManager extends Object`

位置管理器（`LocationManager`）可以擷取系統的定位服務，這個服務允許應用程式定期獲得 GPS 地理位置的更新資料，或當設備裝置接近某個特殊地理位置時會觸發應用程式所指定的 `Intent`。

取得 `LocationManager` 實例可以透過 `Context.getSystemService(Context.LOCATION_SERVICE)`

`public void requestLocationUpdates (String provider, long minTime, float minDistance, PendingIntent intent)`

GPS 位置更新之回呼函式並觸發 `Intent`。

`provider`：定位資訊提供者 `GPS_PROVIDER` 或 `NETWORK_PROVIDER`

`minTime`：更新頻率（微秒）

`minDistance`：更新最短距離（公尺）

```
public void requestLocationUpdates (String provider, long minTime,
float minDistance, LocationListener listener)
```

GPS 位置更新之回呼函式並同時監聽調用 `LocationListener` 接口。在監聽部署時放在建議 `onResume()` 方法，而取消註冊時建議放在 `onPause()` 方法內，實作部署參考如下：

```
// 在 resume 階段設定 mLocationListener 監聽器，以獲得地理位置的更新資料
@Override
protected void onResume() {
```

```
    if (mLocationManager == null) {
        mLocationManager =
            (LocationManager) getSystemService(Context.LOCATION_SERVICE);
        mLocationListener = new MyLocationListener();
    }
    // 獲得地理位置的更新資料
    mLocationManager
        .requestLocationUpdates(LM_GPS, 0, 0, mLocationListener);
    super.onResume();
}
```

|----- LocationManager.GPS_PROVIDER

```
// 在 pause 階段關閉 mLocationListener 監聽器不再獲得地理位置的更新資料
@Override
protected void onPause() {
```

```
    if (mLocationManager != null) {
        // 移除 mLocationListener 監聽器
        mLocationManager.removeUpdates(mLocationListener);
        mLocationManager = null;
    }
    super.onPause();
}
```

```
public interface LocationListener
```

當 GPS 位置更新時可經由此接口來監聽資訊，並且可以由位置管理器調用 `requestLocationUpdates(String, long, float, LocationListener)` 進行註冊。

LocationListener 界面的實作方法有：

public abstract void onLocationChanged (Location location)

當 GPS 位置資訊被更新時會回呼此方法並將相關 GPS 資訊封裝到 location 物件中，以便可以讀出 GPS 位置相關的詳細資訊。

public abstract void onProviderDisabled (String provider)

當 Provider 被用戶關閉時回呼此方法。

public abstract void onProviderEnabled (String provider)

當 Provider 被用戶啟動時回呼此方法。

**public abstract void onStatusChanged (String provider, int status,
Bundle extras)**

當位置資訊的狀態被改變時回呼此方法。其中 status 參數記錄了狀態值：

`LocationProvider.AVAILABLE`：資訊服務中。

`LocationProvider.OUT_OF_SERVICE`：停止服務。

`LocationProvider.PEMLORARILY.UNAVAILABLE`：暫時停止服務。

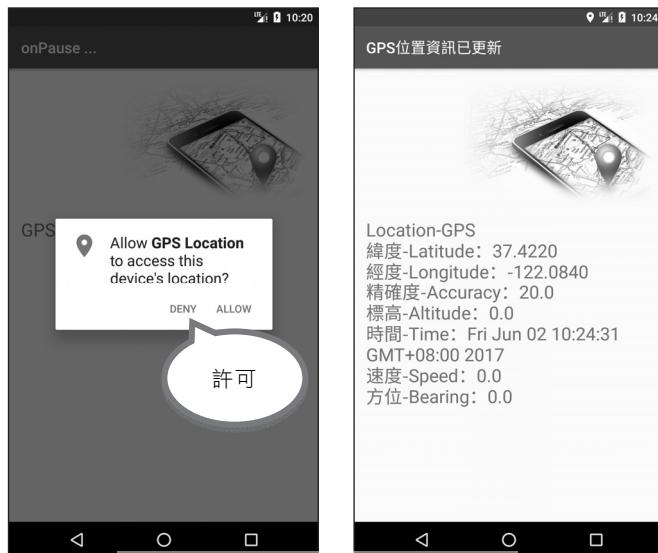
public Location getLastKnownLocation (String provider)

取得最後定位資訊提供者所提供的 Location 資訊。注意使用此方法請特別小心，因為取到的 Location 資料可能是當前資訊提供者所緩存的資料（因為當下可能因為環境的關係暫時無法取得最新定位資料，所以最後資訊提供者就以上一次於緩存中的資訊來提供），另外若無找到，當前資訊提供者則會回傳 null。實作上 GPS 在初始時不是立即定位，若透過此方法來取得 Location 資訊大多會傳回 null（因 Provider 當時可能仍是關閉狀態）。

**public String getBestProvider (Criteria criteria, boolean
enabledOnly)**

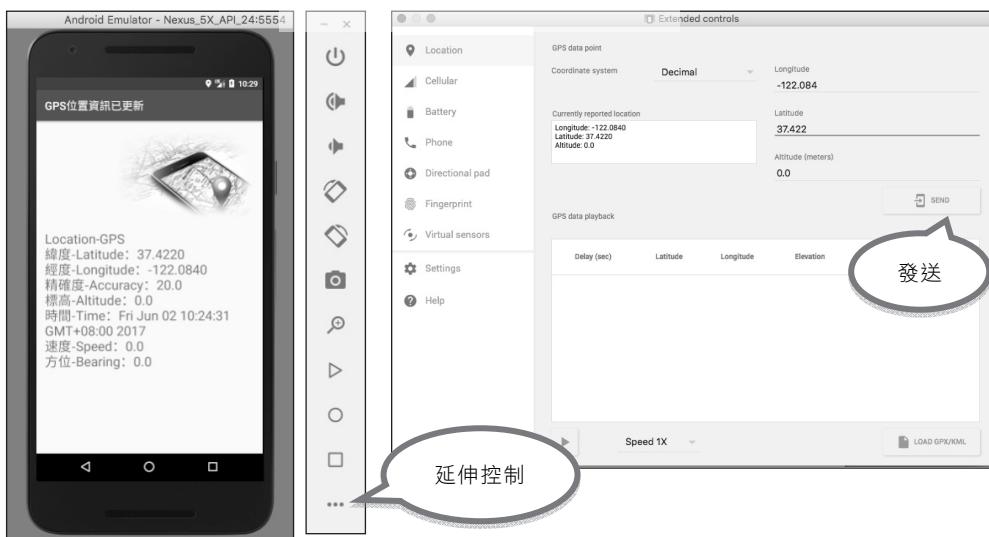
根據自設的 criteria 物件（內含一個定位資訊提供者應該要提供的條件）該方法會根據 criteria 資訊取得最好的位置資訊提供者。enabledOnly 若為 true 是用來表示若位置資訊提供者只有一個，則就以此提供者為主。

↓ 執行結果



範例程式 (Map/app_locationongps) 執行結果

↓ 模擬器延伸控制視窗與 GPS 發送



模擬器延伸控制視窗與 GPS 發送

↓ AndroidManifest.xml 設定

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

    <uses-permission
        android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission
        android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission
        android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission
        android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Map"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

↓ 權限的設定

使用 GPS Provider 設定位置：

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

使用 Network Provider 設定位置：

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

➔ 重點程式碼

MainActivity.java

```
..    ...
21 // 權限請求碼(回呼權限請求時用來區別之，可自行定義不重複的數值資料)
22 private static final int GPS_REQUEST_CODE = 101;
23
24 private Context context;
25 private TextView textView;
26 // 定位管理器
27 private LocationManager mLocationManager;
28 // 定位監聽器
29 private LocationListener mLocationListener;
30
31 @Override
32 public void onCreate(Bundle savedInstanceState) {
33     super.onCreate(savedInstanceState);
34     setContentView(R.layout.activity_main);
35     context = this;
36     mLocationManager =
37         (LocationManager) getSystemService(Context.LOCATION_SERVICE);
38     mLocationListener = new MyLocationListener();
39     textView = (TextView) findViewById(R.id.textView);
40 }
41
42 // 在 resume 階段設定 mLocationListener 監聽器，以獲得地理位置的更新資料
43 @Override
44 protected void onResume() {
45     super.onResume();
46     if (mLocationManager == null) {
47         mLocationManager =
48             (LocationManager) getSystemService(Context.LOCATION_SERVICE);
49         mLocationListener = new MyLocationListener();
50     }
51
52     // 授權驗證 SDK >= 23 (Android 6.0)
53     // 確認是否之前已經授權過此認證
54     if (ContextCompat.checkSelfPermission(this, ACCESS_FINE_LOCATION) ==
55         PackageManager.PERMISSION_GRANTED ) {
56         // 若先前已授權過，則獲得地理位置的更新資料 (GPS 與 NETWORK 都註冊)
57         mLocationManager.requestLocationUpdates(
58             LocationManager.GPS_PROVIDER, 0, 0, mLocationListener);
59         mLocationManager.requestLocationUpdates(
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14

```
61         LocationManager.NETWORK_PROVIDER, 0, 0, mLocationListener);
62     openGPS(); // 檢查 GPS 是否已開啟 ?
63 } else { // 若之前沒有授權過
64     // 彈跳出授權小視窗，讓使用者當場授權
65     ActivityCompat.requestPermissions(this,
66         new String[] {ACCESS_FINE_LOCATION}, // 需授權的權限
67         GPS_REQUEST_CODE // 權限請求碼
68     );
69 }
70 }

72 // 授權回呼結果 (授權驗證 SDK >= 23 (Android 6.0))
73 @Override
74 public void onRequestPermissionsResult(int requestCode,
75     String[] permissions, int[] grantResults) {
76     switch (requestCode) {
77         case GPS_REQUEST_CODE: // 權限請求碼
78             if (grantResults.length > 0
79                 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
80                 setTitle("授權驗證成功 ...");
81             } else {
82                 setTitle("授權驗證失敗 ...");
83             }
84     }
85 }

87 // 在 pause 階段關閉 mLocationListener 監聽器不再獲得地理位置的更新資料
88 @Override
89 protected void onPause() {
90     if (mLocationManager != null) {
91         // 移除 mLocationListener 監聽器
92         mLocationManager.removeUpdates(mLocationListener);
93         mLocationManager = null;
94     }
95     setTitle("onPause ...");
96     super.onPause();
97 }

99 // 定位監聽器實作
100 private class MyLocationListener implements LocationListener {
101     // GPS 位置資訊已更新
102     public void onLocationChanged(Location location) {
103         textView.setText("Location-GPS" + "\n" +
104     }
```



```
105             "緯度-Latitude : " +
106             String.format("%.4f", location.getLatitude()) + "\n" +
107             "經度-Longitude : " +
108             String.format("%.4f", location.getLongitude()) + "\n" +
109             "精確度-Accuracy : " + location.getAccuracy() + "\n" +
110             "標高-Altitude : " + location.getAltitude() + "\n" +
111             "時間-Time : " + new Date(location.getTime()) + "\n" +
112             "速度-Speed : " + location.getSpeed() + "\n" +
113             "方位-Bearing : " + location.getBearing());
114         setTitle("GPS 位置資訊已更新");
115     }
116     public void onProviderDisabled(String provider) {
117     }
118
119     public void onProviderEnabled(String provider) {
120     }
121     // GPS 位置資訊的狀態被更新
122     public void onStatusChanged(String provider,int status,
123                                Bundle extras) {
123
124         switch (status) {
125             case LocationProvider.AVAILABLE:
126                 setTitle("服務中");
127                 break;
128             case LocationProvider.OUT_OF_SERVICE:
129                 setTitle("沒有服務");
130                 break;
131         }
132     }
133 }
134 // 檢查 GPS 是否已開啟 ?
135 public void openGPS() {
136     boolean gps = mLocationManager
137         .isProviderEnabled(LocationManager.GPS_PROVIDER);
138     boolean network = mLocationManager
139         .isProviderEnabled(LocationManager.NETWORK_PROVIDER);
140     Toast.makeText(context, "GPS : " + gps + ", Network : " + network,
141                   Toast.LENGTH_SHORT).show();
142     if (gps || network) {
143         return;
144     } else {
145         // 自動導向 GPS 設定畫面
146         Intent gpsOptionsIntent = new Intent(
147             android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS);
148         startActivity(gpsOptionsIntent);
```

```
149 }  
150 }  
.. ...
```

➔ 重點說明

程式第 36、37 行取得 `LocationManager` 服務實例。

程式第 38 行建立 GPS 監聽器。

程式第 45~70 行在 `onResume()` 階段設定 `mLocationListener` 監聽器，以獲得地理位置的更新資料。

程式第 54~66 行 GPS 驗證授權

```
// 確認是否之前已經授權過此認證  
if (ContextCompat.checkSelfPermission(this, ACCESS_FINE_LOCATION) ==  
    PackageManager.PERMISSION_GRANTED) {  
    // 若先前已授權過，則獲得地理位置的更新資料 (GPS 與 NETWORK 都註冊)  
    mLocationManager.requestLocationUpdates(  
        LocationManager.GPS_PROVIDER, 0, 0, mLocationListener);  
    mLocationManager.requestLocationUpdates(  
        LocationManager.NETWORK_PROVIDER, 0, 0, mLocationListener);  
} else { // 若之前沒有授權過  
    // 彈跳出「授權小視窗」(如下圖)，讓使用者當場授權  
    ActivityCompat.requestPermissions(this,  
        new String[] {ACCESS_FINE_LOCATION}, // 需授權的權限  
        GPS_REQUEST_CODE // 權限請求碼  
    );  
}
```



Allow GPS Location
to access this
device's location?

DENY ALLOW

授權小視窗

程式第 62 行。檢查 GPS 是否已開啟？

程式第 74~85 行。使用者授權回呼結果

並透過(第 78、79 行)

```
if (grantResults.length > 0  
    && grantResults[0] == PackageManager.PERMISSION_GRANTED)
```

來判斷是否通過使用者的授權？

程式第 89~97 行在 onPause() 階段關閉 mLocationListener 監聽器不再獲得地理位置的更新資料。

程式第 100~133 行實作定位監聽器。程式第 105~113 行顯示緯度-Latitude、經度-Latitude、精確度-Accuracy、標高-Altitude、時間-Time、速度-Speed 與方位-Bearing 資訊。



關於 GPS_PROVIDER 與 NETWORK_PROVIDER 的比較與選擇以下就以一個比較表來說明：

表：GPS_PROVIDER 與 NETWORK_PROVIDER

	GPS_PROVIDER	NETWORK_PROVIDER
精度	高	低
耗電	多	少
定位	會因為天氣或障礙物無法取得資料	獲取資訊速度較快

為了程式的通用性以及動態選擇定位資訊提供者（**Location Provider**）可以利用 **Criteria** 來自設一個位置資訊提供者應該要提供的條件，系統會自動根據此條件匹配要選擇哪一種資訊提供者？參考範例如下：

```
LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);

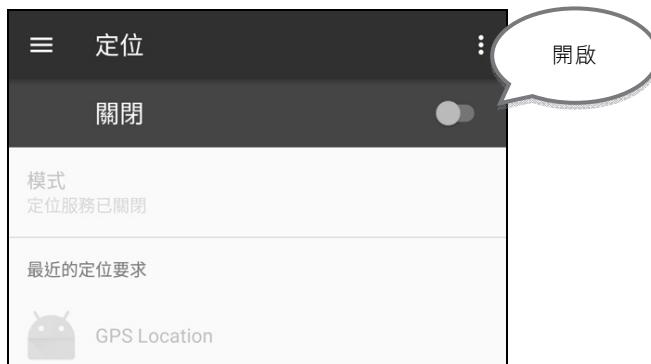
Criteria criteria = new Criteria();
// 設定定位資訊提供者應該要提供的條件 ?
criteria.setAccuracy(Criteria.ACCURACY_FINE); // 精度要求的高低 ?
criteria.setAltitudeRequired(false); // 是否要求海拔資訊 ?
...
criteria.setPowerRequirement(Criteria.POWER_LOW); // 對電量的要求

// 利用 getBestProvider()方法來取得最佳定位資訊提供者。
// 該方法會根據 criteria 資訊取得最好的 Provider。
location = locationManager
.getLastKnownLocation(locationManager.getBestProvider(criteria, true));
```

程式 135~150 行，檢查使用者是否有開啟 GPS？若無則系統會自動導向 GPS 設定畫面，讓使用者手動開啟。

自動導向 GPS 設定畫面 intent：

```
Intent gpsOptionsIntent = new Intent(
    android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS);
startActivity(gpsOptionsIntent);
```



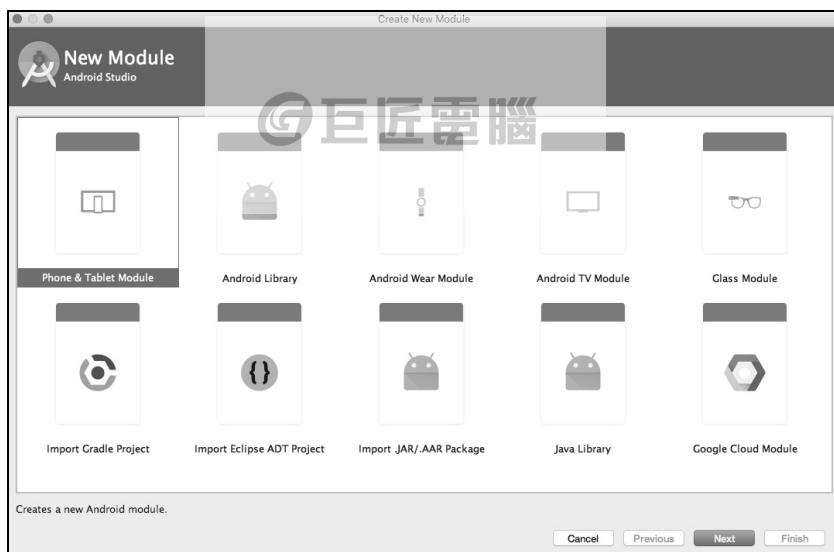
開啟/關閉 GPS 定位

14-2 Google 地圖使用許可權申請與 Hello Map

Android 在 2.x 版之後針對 App 內嵌 Google Map 的使用權必須要進行許可權申請（現行是免費申請不過僅針對 Android 裝置），目前最新是來到 Google Map Android API v2。申請流程一直是初次接觸的開發者頭痛的問題，不過這個問題在 Android Studio 得到了一個還算簡易的方法，接下來請跟著書本一步一步帶領進入 Google 地圖許可權申請，與看到 Google Map 在你的 Activity 上：

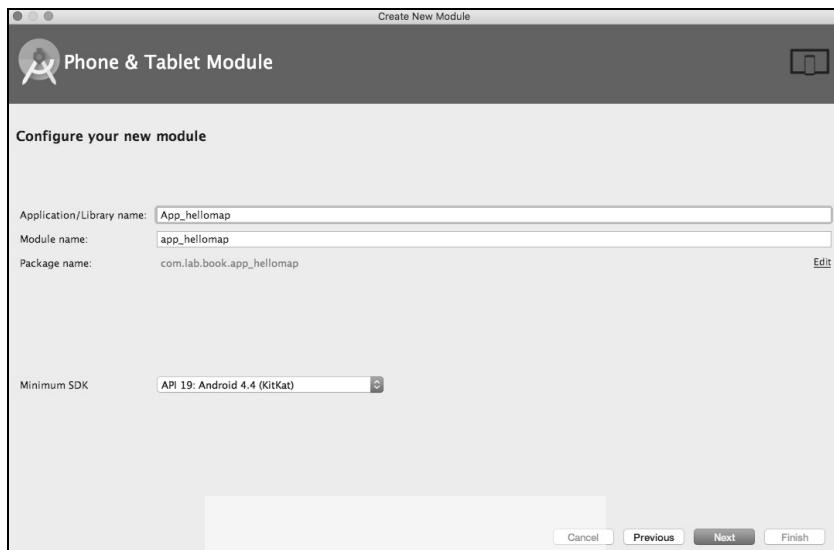
申請步驟如下：

STEP 1 File > New > New Module 在專案中建立一個新 Module 或另外新增一個專案。

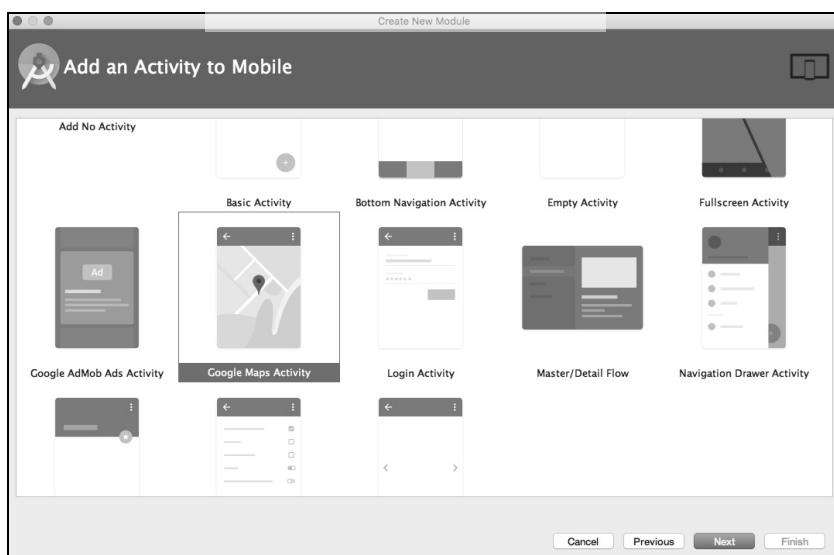


1
2
3
4
5
6
7
8
9
10
11
12
13
14

STEP 2 設定 Application 與 Module 名稱。Package name : com.lab.book.app_hellomap (申請 Google Map Api key 時也會用到)。



STEP 3 選擇 Google Maps Activity。



STEP 4 按下「Finish」鍵完成 Google Maps Activity 設定。



STEP 5 進入 res/values/google_maps_api.xml (預設會自動開啟)。



在該檔中程式第 7 行請「複製」下來並「開啟 Chrome 瀏覽器」後「貼入此 URL」進行 Google Map API 申請作業。

URL 參數說明：

```
https://console.developers.google.com/flows/enableapi?
apiid=maps_android_backend&
keyType=CLIENT_SIDE_ANDROID&
r=8D:5A:36:2C:3D:C9:C6:FC:42:B8:B8:53:59:4F:FB:90:EA:56:4E:8A%3B
com.lab.book.app_hellomap
```

其中 r 參數內容前半部就是本機 debug.keystore 的 SHA1

8D:5A:36:2C:3D:C9:C6:FC:42:B8:B8:53:59:4F:FB:90:EA:56:4E:8A

後半部就是專案的 package name。

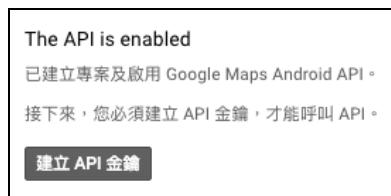
com.lab.book.app_hellomap

STEP 6 連入後按下「繼續」。若您沒有專案請先建立專案。



Google Developer Console 後台 <https://console.developers.google.com>

STEP 7 Google Maps Android API 啟用成功，按下「建立 API 金鑰」。



STEP 8 得到金鑰後按下「關閉」。



STEP 9 查看金鑰

金鑰 : AIzaSyAP5yusZ6HVdW3y9YPOw7L1SQ-QRiSRFIg



STEP 10 回到 res/values/google_maps_api.xml 將 API 金鑰字串貼入程式第 24 行中。

```
23 <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">
24 | AIzaSyAP5yusZ6HVdW3y9YPOw7L1SQ-QRiSRFIg
25 </string>
```

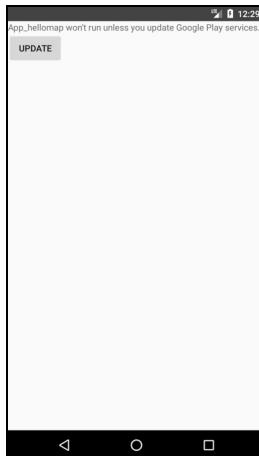
STEP 11 執行 app_hellomap 範例程式

執行結果



注意若您的模擬器執行本範例時產生以下訊息，請更新到 **Android Studio 2.4** 或以後的版本（2.4 之後模擬器恢復支援 Google Play Services），並請重新建立新的模擬器。

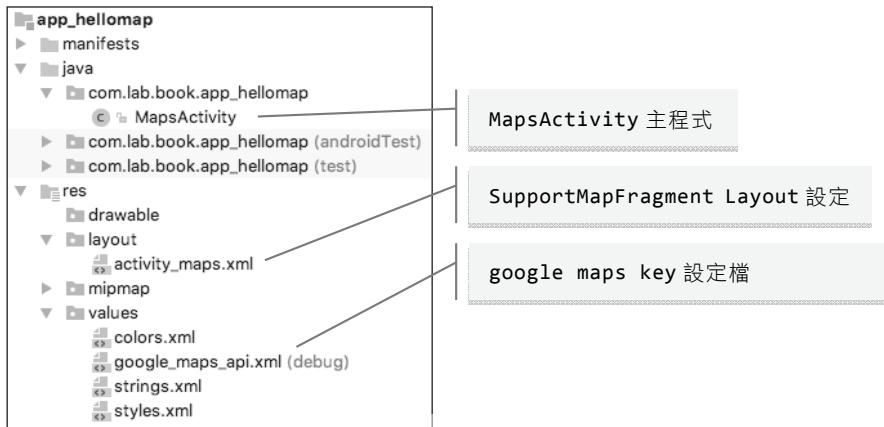
下載網址：<https://developer.android.com/studio/archive.html>



模擬器不支援 Google Play Services

14-3 Hello Map 範例結構與基本使用

範例程式碼（Map/app_hellomap）部署與功能說明



Build.gradle (Module : app_hellomap)

```
dependencies {  
    compile fileTree(dir: 'libs', include: ['*.jar'])  
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {  
        exclude group: 'com.android.support', module: 'support-annotations'  
    })  
    compile 'com.android.support:appcompat-v7:25.3.1'  
    compile 'com.google.android.gms:play-services-maps:10.2.4'  
    testCompile 'junit:junit:4.12'  
}
```

載入 google-play-services 資源

UI 界面部署與設計

activity_maps.xml

設置一個 fragment 來呈現 Google 地圖

```
1 <fragment xmlns:android="http://schemas.android.com/apk/res/android"  
2     xmlns:map="http://schemas.android.com/apk/res-auto"  
3     xmlns:tools="http://schemas.android.com/tools"  
4     android:id="@+id/map"  
5     android:name="com.google.android.gms.maps.SupportMapFragment"  
6     android:layout_width="match_parent"  
7     android:layout_height="match_parent"  
8     tools:context="com.lab.book.app_hellomapMapsActivity" />
```

建立一個 fragment 標籤且類別設定為 com.google.android.gms.maps.SupportMapFragment，此 fragment 就是用來呈現與存放 Google 地圖的容器。

關於 SupportMapFragment 相容支援類別：

Google Maps Android API 基本要求 API level 12 (Android 3.1) 或更高的 API 可支援 MapFragment。若要使用較舊的 API (API level < 12)，可以透過 SupportMapFragment 相容支援類別來達到同樣的功能。

AndroidManifest.xml 設定

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.lab.book.app_helloworld">
4
5      <!--
6          The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
7          Google Maps Android API v2, but you must specify either coarse or fine
8          location permissions for the 'MyLocation' functionality.
9      -->
10     <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
11
12     <application android:allowBackup="true" android:icon="@mipmap/ic_launcher"
13         android:label="App_helloworld" android:roundIcon="@mipmap/ic_launcher_round"
14         android:supportsRtl="true" android:theme="@style/AppTheme">
15
16         <!--
17             The API key for Google Maps-based APIs is defined as a string resource.
18             (See the file "res/values/google_maps_api.xml").
19             Note that the API key is linked to the encryption key used to sign the APK.
20             You need a different API key for each encryption key, including the
21             release key that is used to sign the APK for publishing.
22             You can define the keys for the debug and release targets
23             in src/debug/ and src/release/.
24         -->
25         <meta-data android:name="com.google.android.geo.API_KEY"
26             android:value="AIzaSyAP5yusZ6HVdW3y9YP0w7LlsQ-QRiSRF1g" />
27
28         <activity android:name=".MapsActivity" android:label="Map">
29             <intent-filter>
30                 <action android:name="android.intent.action.MAIN" />
31
32                 <category android:name="android.intent.category.LAUNCHER" />
33             </intent-filter>
34         </activity>
35     </application>
36
37 </manifest>
```

重點說明

第 10 行加入 GPS 權限。

第 25、26 行設定 Google Geo API Key。

重點程式碼

```
MapsActivity.java
...
13  public class MapsActivity extends FragmentActivity
14      implements OnMapReadyCallback {
15
16     private GoogleMap mMap;
17
18     @Override
19     protected void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.activity_maps);
22         // 取得 SupportMapFragment
23         SupportMapFragment mapFragment =
24             (SupportMapFragment) getSupportFragmentManager()
25             .findFragmentById(R.id.map);
26         // 非同步地圖通知
27         mapFragment.getMapAsync(this);
28     }
29
30     // 地圖回呼函式
31     @Override
32     public void onMapReady(GoogleMap googleMap) {
33         mMap = googleMap;
34
35         // Add a marker in Sydney and move the camera
36         LatLng sydney = new LatLng(-34, 151);
37         mMap.addMarker(new MarkerOptions()
38             .position(sydney).title("Marker in Sydney"));
39         mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
40     }
41 }
...
...
```



重點說明

程式第 27 行為防止 App 鎖死，執行 `getMapAsync(this)`非同步地圖通知(這個方法必須在主執行緒中呼叫)，待地圖準備好之後會主動調用程式第 32 行的 `onMapReady()`方法。

程式第 26~34 行，當地圖備妥後所回呼的方法(此方法也是於主執行緒中執行)。

程式第 36 行，宣告一個經緯度物件 `LatLng`，用來存放經緯度資料。

```
new LatLng(緯度, 經度);
```

程式第 37、28 行，建立一個地圖圖標 `addMarker()`。`MarkerOptions` 是一個圖標參數物件，基本用法如下：

```
new MarkerOptions().position(sydney).title("Marker in Sydney")
```



地圖圖標與參數設定

程式第 34 行將圖標透過 `Camera` 效果移動到地圖指定位置，並且置於畫面正中心。

```
mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
```



`moveCamera()` → 移動展演方式。

`CameraUpdateFactory` → `Camera` 移動展演參數。

`CameraUpdateFactory.newLatLng(sydney)` → 移動到指定經緯度(`sydney`)。

`sydney` 在這邊指的是：`LatLang sydney = new LatLang(-34, 151);`



如何讓 `MapActivity` 有 `ActionBar` ?

實作方式：

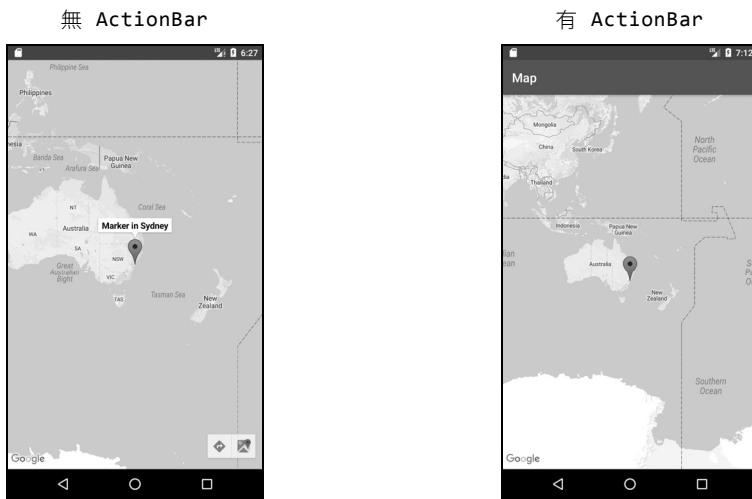
將原本

```
public class MapsActivity extends FragmentActivity ...
```

改為

```
public class MapsActivity extends AppCompatActivity ...
```

有無 `ActionBar` 的 `MapActivity` 樣式呈現如下：

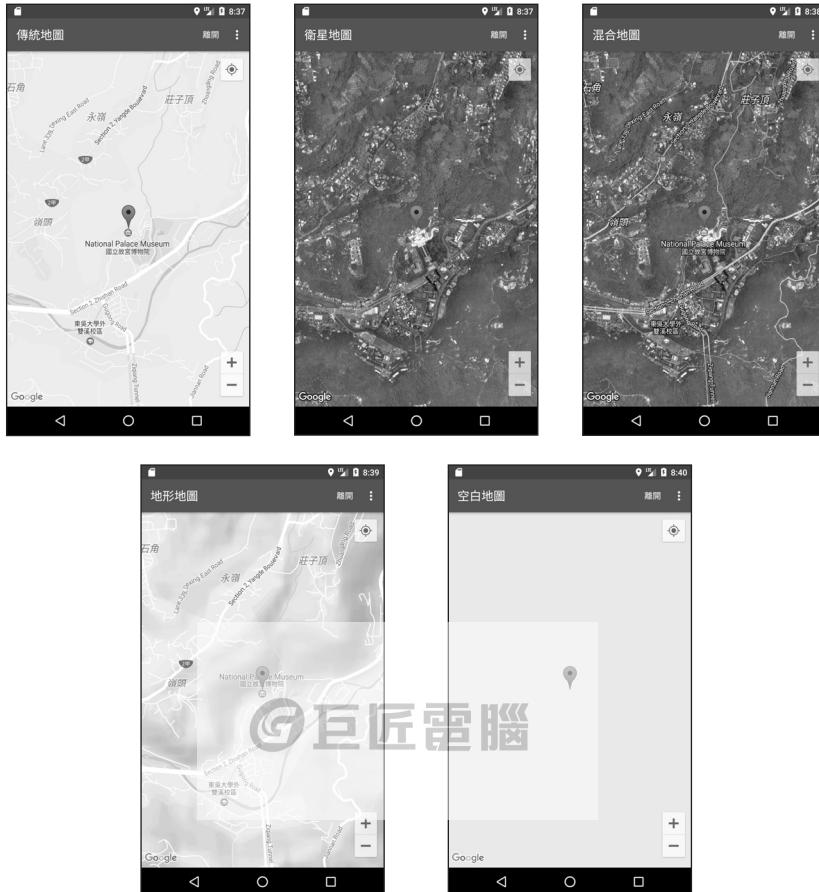


14-3-1 Use Map (地圖使用)

Google 地圖在顯示類型上可分為五種：

1. `GoogleMap.MAP_TYPE_NORMAL`：傳統典型的地圖。
2. `GoogleMap.MAP_TYPE_HYBRID`：混合衛星照片及道路地圖。
3. `GoogleMap.MAP_TYPE_SATELLITE`：衛星照片。
4. `GoogleMap.MAP_TYPE_TERRAIN`：地形圖。
5. `GoogleMap.MAP_TYPE_NONE`：什麼都沒有。

五種地圖顯示樣式如下圖，請參考：



Google 地圖五種顯示類型

而在顯示視點的效果動畫上是以攝影機 (Camera) 的概念來設計瀏覽地圖，可以運用的範圍非常多，包含：動畫效果執行期間 (微秒)、zoom (縮放)、target/location (標的位置)、bearing (旋轉方位) 與 tilt (傾斜角度) 等...，更多請參考 [Changing the View https://developers.google.com/maps/documentation/android/views?hl=zh-TW](https://developers.google.com/maps/documentation/android/views?hl=zh-TW)。



Google 地圖顯示視點

◆ 範例程式 (Map/app_usemap) 相關 Java API

public final class GoogleMap

Google 地圖完整 Android API，該類別不可以直接實例化，必須透過應用程式中的 **MapFragment** 或 **MapView** 所提供的 **getMap()**方法來取得實例對象。

public final class UiSettings

設置 Google 地圖的使用者操作界面，可以調用 **map.getUiSettings()**來取得此接口。常用在地圖上的界面設定有：

setZoomControlsEnabled(true);

開啟/關閉縮放鈕

setScrollGesturesEnabled(true);

開啟/關閉地圖捲動手勢

setZoomGesturesEnabled(true);

開啟/關閉地圖縮放手勢

setTiltGesturesEnabled(true);

開啟/關閉地圖傾斜手勢

```
setRotateGesturesEnabled(true);
```

開啟/關閉地圖旋轉手勢

```
public final class LatLng
```

用以封裝儲存一組經緯度值（單位：度）。

```
public final class CameraPosition
```

統合設置所有地圖相機鏡頭相關設定參數，包含：`target` 標的位置、`zoom` 縮放大小、`bearing` 旋轉方位與 `tilt` 傾斜角度等...。範例設定如下，請參考：

```
CameraPosition cameraPos = new CameraPosition.Builder()  
    .target(latlng).zoom(17.0f).bearing(300).tilt(45).build();
```

其中 `zoom` 的設定要 $>=17$ 才會顯示 3D 建築物。3D 建築物呈現需配合 `mMap.setBuildingsEnabled(true);`；然而不是每一個地點都能顯示。`bearing` 旋轉方位設定是從北開始順時針 $0\sim360$ 度的參數設置，`tilt` 又稱為顯示視角 (`viewing angle`)，顯示作用如下：

.bearing(300).tilt(0) 



```
.bearing(300).tilt(45)
```



有傾斜角度的設置並搭配適當的 zoom level 就有機會看到地圖上的建築物

tilt(45)

3



4

Google 地圖相機鏡頭的 `tilt` 傾斜角度效果實現與 `zoom` 縮放大小設定有關，整理如下：

zoom 與 tilt 在參數搭配上的規則：

1. 要有 `tilt` 傾斜效果則 `zoom level` 必須介於 10~30 之間。
2. `zoom level` 設定在 10~14 則最大 `tilt` 必須設定 30~45（以線性比率增加），例如：`zoom level=12`，最大 `tile` 數值=37.5。
3. `zoom level` 設定在 14~15.5 則最大 `tilt` 必須設定 45~67.5（以線性比率增加）。
4. `zoom level` 設定超過 15.5 則最大 `tilt` 必須設定為 67.5。

`public final class CameraUpdate`

定義地圖相機鏡頭的移動情境，使用上可以調用 `animateCamera (CameraUpdate)`，`animateCamera (CameraUpdate , GoogleMap.CancelableCallback)` 或 `moveCamera (CameraUpdate)` 來修改 Google 地圖上新變更的相機移動情境。取得 `CameraUpdate` 實例可以使用 `CameraUpdateFactory` 這個工廠類別來得到。例如：變更設定地圖相機鏡頭 `zoom in` 的動畫實作程式碼如下：

```
GoogleMap map = ...;
map.animateCamera(CameraUpdateFactory.zoomIn());
```

```
public final void animateCamera (CameraUpdate update, int
durationMs, GoogleMap.CancelableCallback callback)
```

根據 CameraUpdate 的設置與指定時間 durationMs (微秒) 來進行地圖動畫顯示。
durationMs：設定動畫進行時間 (微秒)。

callback：是一個動畫進行中的回呼，藉以監聽地圖動畫正常完成 onFinish() 與被取消 onCancel()的任務回調。

範例程式 (Map/app_usemap)：地圖樣式、地點查找、地圖旋轉動畫

↓ 執行結果



 點程式碼

```
MapsActivity.java
...
25 // 權限請求碼
26 private static final int GPS_REQUEST_CODE = 101;
27
28 private GoogleMap mMap;
29
30 @Override
31 protected void onCreate(Bundle savedInstanceState) {
32     super.onCreate(savedInstanceState);
33     setContentView(R.layout.activity_maps);
34     SupportMapFragment mapFragment =
35         (SupportMapFragment) getSupportFragmentManager()
36             .findFragmentById(R.id.map);
37     mapFragment.getMapAsync(this);
38 }
39
40 @Override
41 public void onMapReady(GoogleMap googleMap) {
42     mMap = googleMap;
43     // 地圖參數設置
44     setMapArgs();
45     // 預設地點
46     LatLng museum = new LatLng(25.102623, 121.548514);
47     mMap.addMarker(new MarkerOptions().position(museum).title("故宮博物院"));
48     mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(museum, 15f));
49 }
50
51 // 地圖參數設置
52 private void setMapArgs() {
53     // 地圖操作界面功能設定
54     UiSettings ui = mMap.getUiSettings();
55     // 開啟/關閉縮放鈕
56     ui.setZoomControlsEnabled(true);
57     // 開啟/關閉地圖捲動手勢
58     ui.setScrollGesturesEnabled(true);
59     // 開啟/關閉地圖縮放手勢
60     ui.setZoomGesturesEnabled(true);
61     // 開啟/關閉地圖傾斜手勢
62     ui.setTiltGesturesEnabled(true);
63     // 開啟/關閉地圖旋轉手勢
64     ui.setRotateGesturesEnabled(true);
```

```
65 // 設定地圖類型
66 mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
67 // 地圖上顯示建築物
68 // 注意：zoom 的設定要 >=17 才會顯示建築物
69 mMap.setBuildingsEnabled(true);
70 // 確認使用者是否允許開啟 GPS
71 if (ContextCompat.checkSelfPermission(this, ACCESS_FINE_LOCATION) ==
72     PackageManager.PERMISSION_GRANTED ) {
73     // 顯示目前所在位置鈕
74     mMap.setMyLocationEnabled(true);
75     // 檢查 GPS 是否有開啟 ?
76     openGPS();
77 } else { // 若之前沒有授權過
78     // 彈跳出授權小視窗，讓使用者當場授權
79     ActivityCompat.requestPermissions(this,
80             new String[] {ACCESS_FINE_LOCATION},
81             GPS_REQUEST_CODE // 權限請求碼
82         );
83 }
84 }
85
86
87 @Override
88 public void onRequestPermissionsResult(int requestCode,
89                                     String[] permissions, int[] grantResults) {
90     switch (requestCode) {
91         case GPS_REQUEST_CODE: // 權限請求碼
92             if (grantResults.length > 0
93                 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
94                 setTitle("GPS 授權驗證成功 ...");
95                 // 地圖參數設置
96                 setMapArgs();
97             } else {
98                 setTitle("GPS 授權驗證失敗 ...");
99             }
100        }
101    }
102
103
104 @Override
105 public boolean onCreateOptionsMenu(Menu menu) {
106     MenuItem item1 = menu.add(0, 1, 1, "傳統地圖");
107     MenuItem item2 = menu.add(0, 2, 2, "衛星地圖");
108     MenuItem item3 = menu.add(0, 3, 3, "混合地圖");
109     MenuItem item4 = menu.add(0, 4, 4, "地形地圖");
```



1
2
3
4
5
6
7
8
9
10
11
12
13
14

```
109     MenuItem item5 = menu.add(0, 5, 5, "空白地圖");
110     MenuItem item6 = menu.add(0, 6, 6, "台北 101");
111     MenuItem item7 = menu.add(0, 7, 7, "日月潭");
112     MenuItem exit = menu.add(0, 9, 9, "離開");
113     exit.setShowAsAction(MenuItem.SHOW_AS_ACTION_ALWAYS);
114     return true;
115 }
116
117 @Override
118 public boolean onOptionsItemSelected(MenuItem item) {
119     switch (item.getItemId()) {
120         case 1:
121             mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
122             break;
123         case 2:
124             mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
125             break;
126         case 3:
127             mMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
128             break;
129         case 4:
130             mMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
131             break;
132         case 5:
133             mMap.setMapType(GoogleMap.MAP_TYPE_NONE);
134             break;
135         case 6:
136             // 台北 101 緯經度 : (25.033924, 121.564547)
137             LatLng latlng1 = new LatLng(25.033924, 121.564547);
138             // 演示地圖相機鏡頭動畫效果
139             playAnimateCamera(latlng1, 10000);
140             break;
141         case 7:
142             // 日月潭緯經度 : (23.859758, 120.916221)
143             LatLng latlng2 = new LatLng(23.859758, 120.916221);
144             // 演示地圖相機鏡頭動畫效果
145             playAnimateCamera(latlng2, 10000);
146             break;
147         case 9:
148             finish();
149             break;
150     }
151     setTitle(item.getTitle());
152     return super.onOptionsItemSelected(item);
```

```

153 }
154
155 // 在地圖上演示地圖相機鏡頭動畫效果
156 private void playAnimateCamera(LatLng latlng, int durationMs) {
157     // 設置相關地圖相機鏡頭位置參數
158     CameraPosition cameraPos = new CameraPosition.Builder().target(latlng)
159         .zoom(17.0f).bearing(300).tilt(45).build();
160     // 定義地圖相機鏡頭移動
161     CameraUpdate cameraUpt = CameraUpdateFactory
162         .newCameraPosition(cameraPos);
163     // 地圖相機鏡頭動畫行程設定
164     mMap.animateCamera(cameraUpt, durationMs, null);
165
166 }
167
168 // 檢查 GPS 是否有開啟 ?
169 public void openGPS() {
170     LocationManager mLocationManager =
171         (LocationManager) getSystemService(Context.LOCATION_SERVICE);
172     boolean gps = mLocationManager
173         .isProviderEnabled(LocationManager.GPS_PROVIDER);
174     boolean network = mLocationManager
175         .isProviderEnabled(LocationManager.NETWORK_PROVIDER);
176     Toast.makeText(this, "GPS : " + gps + ", Network : " + network,
177         Toast.LENGTH_SHORT).show();
178     if (gps || network) {
179         return;
180     } else {
181         // 開啟手動 GPS 設定畫面
182         Intent gpsOptionsIntent = new Intent(
183             android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS);
184         startActivity(gpsOptionsIntent);
185     }
186 }
187 ...

```

重點說明

程式第 52~85 行 `setMapArgs()`方法實作了各種地圖使用者操作界面功能設定。其中若要在地圖中顯示「目前所在位置鈕」，必須要有 GPS 權限的授權：

1
2
3
4
5
6
7
8
9
10
11
12
13
14

```
// 確認使用者是否允許開啟 GPS
if (ContextCompat.checkSelfPermission(this, ACCESS_FINE_LOCATION) ==
        PackageManager.PERMISSION_GRANTED ) {
    // 顯示目前所在位置鈕
    mMap.setMyLocationEnabled(true);
} else { // 若之前沒有授權過
    // 彈跳出授權小視窗，讓使用者當場授權
    ActivityCompat.requestPermissions(this,
            new String[] {ACCESS_FINE_LOCATION},
            GPS_REQUEST_CODE // 權限請求碼
    );
}
```

程式第 156~166 行在地圖上演示地圖相機動畫效果的方法實作。

程式第 169~186 檢查手機是否有開啟 GPS ?

14-3-2 經緯度與地址/地標的反查

Google Web Service API 提供了經緯度與地址/地標的反查服務，其回傳資料格式可以選擇 XML 或 JSON 如下所示：

回傳 XML 資料格式 URL：

```
http://maps.googleapis.com/maps/api/geocode/xml
```

回傳 JSON 資料格式 URL：

```
http://maps.googleapis.com/maps/api/geocode/json
```

Google Web Service API 使用範例如下（使用 JSON 回傳格式）：

1. 已知經緯度（25.047795, 121.516900）反查地址/地標：

```
http://maps.googleapis.com/maps/api/geocode/json?latlng=25.047795,
121.516900&sensor=false&language=zh_tw。
```

```
  "types" : [ "country", "political" ],
  },
  {
    "long_name" : "100",
    "short_name" : "100",
    "types" : [ "postal_code" ]
  }
],
"formatted_address" : "100台灣台北市中正區北平西路3號",
"geometry" : {
  "location" : {
    "lat" : 25.0479239,
    "lng" : 121.517081
  },
  "location_type" : "ROOFTOP",
  "viewport" : {
    "northeast" : {
      "lat" : 25.0492728802915,
      "lng" : 121.5184299802915
    },
    "southwest" : {
      "lat" : 25.04657491970849,
      "lng" : 121.5157320197085
    }
  }
}
```

2. 已知地址 / 地標（台灣台北市中正區北平西路 3 號）反查經緯度：
[http://maps.googleapis.com/maps/api/geocode/json?address=台灣台北市中正區北平西路 3 號&sensor=false&language=zh_tw²](http://maps.googleapis.com/maps/api/geocode/json?address=%E5%8D%8A%E5%8F%A3%E5%8C%97%E5%B8%82%E4%B8%AD%E6%AD%A3%E5%8D%80%E5%8C%97%E5%B9%B3%E8%A5%BF%E8%B7%AF3%E8%99%9F)。

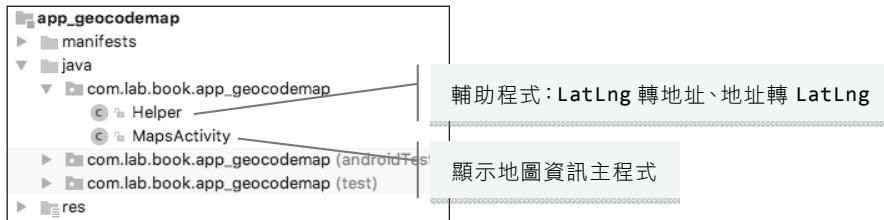
```
  "types" : [ "country", "political" ],
  },
  {
    "long_name" : "100",
    "short_name" : "100",
    "types" : [ "postal_code" ]
  }
],
"formatted_address" : "100台灣台北市中正區北平西路3號",
"geometry" : {
  "location" : {
    "lat" : 25.0479239,
    "lng" : 121.517081
  },
  "location_type" : "ROOFTOP",
  "viewport" : {
    "northeast" : {
      "lat" : 25.0492728802915,
      "lng" : 121.5184299802915
    },
    "southwest" : {
      "lat" : 25.04657491970849,
      "lng" : 121.5157320197085
    }
  }
}
```

² "台灣台北市中正區北平西路 3 號"會被 URL 編碼為
"%E5%8F%B0%E7%81%A3%E5%8F%B0%E5%8C%97%E5%B8%82%E4%B8%AD%E6%AD%A3%E5%8D%80%E5%8C%97%E5%B9%B3%E8%A5%BF%E8%B7%AF3%E8%99%9F"。

範例程式 (Map/app_geocodemap)

將說明 **LatLng** 轉地址與地址轉 **LatLng** 的功能使用（使用者可以任意點擊地圖上任何一點就可以反查該點所代表的地址/地標）。

➔ 程式碼類別與功能說明



範例程式 **Helper.java** 實作了經緯度與地址/地標的反查機制，其重點原始碼內容如下：

Helper.java	
..	...
13	import org.json.JSONArray;
14	import org.json.JSONObject;
..	...
18	public class Helper {
19	
20	public static String DOMAIN =
21	"http://maps.googleapis.com/maps/api/geocode/json";
22	
23	// 同步取得 Address
24	public static String getAddressByLatLng(LatLng latLng) {
25	String addr = "";
26	Helper.LatLngToAddress ga = new Helper.LatLngToAddress(latLng);
27	FutureTask<String> future = new FutureTask<String>(ga);
28	new Thread(future).start();
29	try {
30	addr = future.get();
31	} catch (Exception e) {
32	
33	}
34	return addr;
35	}
36	

```
37 // 同步取得 LatLng
38 public static LatLng getLatLngByAddress(String address) {
39     LatLng latLng = null;
40     Helper.AddressToLatLng ga = new Helper.AddressToLatLng(address);
41     FutureTask<LatLng> future = new FutureTask<LatLng>(ga);
42     new Thread(future).start();
43     try {
44         latLng = future.get();
45     } catch (Exception e) {
46
47     }
48     return latLng;
49 }
50
51 private static class HttpUtil {
52     public static String get(String urlString) throws Exception {
53         InputStream is = null;
54         Reader reader = null;
55         StringBuilder str = new StringBuilder();
56         URL url = new URL(urlString);
57         URLConnection URLConn = url.openConnection();
58         URLConn.setRequestProperty("User-agent", "IE/6.0");
59         is = URLConn.getInputStream();
60         reader = new InputStreamReader(is, "UTF-8");
61         char[] buffer = new char[1];
62         while (reader.read(buffer) != -1) {
63             str.append(new String(buffer));
64         }
65         return str.toString();
66     }
67 }
68
69 // LatLng 轉 Address
70 private static class LatLngToAddress implements Callable<String> {
71
72     private String queryURLString = DOMAIN
73         + "?latlng=%s,%s&sensor=true&language=zh_tw";
74     private String address = null;
75     private LatLng latLng;
76
77     LatLngToAddress(LatLng latLng) {
78         this.latLng = latLng;
79     }
80 }
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14

```
81     @Override
82     public String call() {
83         // 輸入緯經度得到地址
84         address = getAddressByLocation();
85         return address;
86     }
87
88     private String getAddressByLocation() {
89         String urlString = String.format(queryURLString, latLng.latitude,
90             latLng.longitude);
91         try {
92             // 取得 json string
93             String jsonStr = HttpUtil.get(urlString);
94             // 取得 json 根陣列節點 results
95             JSONArray results = new JSONObject(jsonStr)
96                 .getJSONArray("results");
97             if (results.length() >= 1) {
98                 // 取得 results[0]
99                 JSONObject jsonObject = results.getJSONObject(0);
100                // 取得 formatted_address 屬性內容
101                address = jsonObject.optString("formatted_address");
102            }
103        } catch (Exception e) {
104            e.printStackTrace(System.err);
105        }
106        return address;
107    }
108
109    // Address 轉 LatLng
110    private static class AddressToLatLng implements Callable<LatLng> {
111        private String queryURLString = DOMAIN
112            + "?address=%s&sensor=false&language=zh_tw";
113        private String address;
114
115        AddressToLatLng(String address) {
116            this.address = address;
117        }
118
119        @Override
120        public LatLng call() {
121            LatLng latLng = null;
122            try {
123                // 輸入地址得到緯經度(中文地址需透過 URLEncoder 編碼)
```

```
125     latLng = getLocationByAddress(URLEncoder.encode(address,
126             "UTF-8"));
127     } catch (UnsupportedEncodingException e) {
128     }
129     return latLng;
130 }
131
132 private LatLng getLocationByAddress(String address) {
133     String urlString = String.format(queryURLString, address);
134     LatLng latLng = null;
135     try {
136         // 取得 json string
137         String jsonStr = HttpUtil.get(urlString);
138         // 取得 json 根陣列節點 results
139         JSONArray results = new JSONObject(jsonStr)
140             .getJSONArray("results");
141         System.out.println("results.length() : " + results.length());
142         if (results.length() >= 1) {
143             // 取得 results[0]
144             JSONObject jsonObject = results.getJSONObject(0);
145             // 取得 geometry --> location 物件
146             JSONObject laln = jsonObject.getJSONObject("geometry")
147                 .getJSONObject("location");
148
149             latLng = new LatLng(Double.parseDouble(laln
150                 .getString("lat")), Double.parseDouble(laln
151                 .getString("lng")));
152         }
153     } catch (Exception e) {
154         e.printStackTrace(System.err);
155     }
156     return latLng;
157 }
158 }
159 }
160 }
```

➔ 重點說明

程式第 13、14 行 `import org.json.JSONArray;`
與 `import org.json.JSONObject;` Android 內建剖析 JSON API。

程式第 70~108 行實作經緯度轉地址/地標。程式第 91~102 行分析 json 並取得 `formatted_address` 屬性內容也就是欲求得的地址/地標。

```
// 取得 json string
String jsonStr = HttpUtil.get(urlString);
// 取得 json 根陣列節點 results
JSONArray results = new JSONObject(jsonStr)
    .getJSONArray("results");
if (results.length() >= 1) {
    // 取得 results[0]
    JSONObject jsonObject = results.getJSONObject(0);
    // 取得 formatted_address 屬性內容
    address = jsonObject.optString("formatted_address");
}

{
    "results": [
        {"address_components": [...],
         "formatted_address": "100 台灣台北市中正區北平西路 3 號",
        "geometry": {
            "location": {
                "lat": 25.0479239,
                "lng": 121.517081
            },
            ...
        }
    ]
}
```

程式第 111~159 行實作地址/地標轉經緯度。程式第 136~158 行分析 json 並取得 geometry > location 的屬性 lat 與 lng 內容也就是欲求得的經緯度值。

```
// 取得 json string
String jsonStr = HttpUtil.get(urlString);
// 取得 json 根陣列節點 results
JSONArray results = new JSONObject(jsonStr)
    .getJSONArray("results");
System.out.println("results.length() : " + results.length());
if (results.length() >= 1) {
    // 取得 results[0]
    JSONObject jsonObject = results.getJSONObject(0);
    // 取得 geometry --> location 物件
    JSONObject laln = jsonObject.getJSONObject("geometry")
        .getJSONObject("location");

    latLng = new LatLng(Double.parseDouble(laln
        .getString("lat")), Double.parseDouble(laln
        .getString("lng")));
}
{
    "results" : [
        {"address_components" : [...]
        ,
        "formatted_address" : "100 台灣台北市中正區北平西路 3 號",
        "geometry" : [
            "location" : {
                "lat" : 25.0479239,
                "lng" : 121.517081
            },
            ...
        ]
    }
}
```

重點程式碼

MapsActivity.java

```
.. ...
19  public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {
20
21      private GoogleMap mMap;
22      private EditText editText, editText2;
23      @Override
24      protected void onCreate(Bundle savedInstanceState) {
25          super.onCreate(savedInstanceState);
26          setContentView(R.layout.activity_maps);
27      }
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14

```
28         editText = (EditText) findViewById(R.id.editText);
29         editText2 = (EditText) findViewById(R.id.editText2);
30
31         SupportMapFragment mapFragment =
32             (SupportMapFragment) getSupportFragmentManager()
33                 .findFragmentById(R.id.map);
34         mapFragment.getMapAsync(this);
35     }
36
37     @Override
38     public void onMapReady(GoogleMap googleMap) {
39         mMap = googleMap;
40         // Google 地圖使用者操作界面功能設定
41         UiSettings ui = mMap.getUiSettings();
42         // 開啟/關閉縮放鈕
43         ui.setZoomControlsEnabled(true);
44         // 開啟/關閉地圖捲動手勢
45         ui.setScrollGesturesEnabled(true);
46         // 開啟/關閉地圖縮放手勢
47         ui.setZoomGesturesEnabled(true);
48         // 開啟/關閉地圖傾斜手勢
49         ui.setTiltGesturesEnabled(true);
50         // 開啟/關閉地圖旋轉手勢
51         ui.setRotateGesturesEnabled(true);
52     }
53
54
55     public void onClick(View view) {
56         String address = null;
57         LatLng latLng= null;
58         switch (view.getId()) {
59             case R.id.button:
60                 address = editText.getText().toString();
61                 // 根據地址資料取得經緯度物件
62                 latLng = Helper.getLatLngByAddress(address);
63                 Toast.makeText(this, "經緯度：" + latLng, Toast.LENGTH_SHORT).show();
64                 break;
65             case R.id.button2:
66                 // 將輸入框欄位內容解析為 LatLng 物件資料
67                 String[] pos = editText2.getText().toString().split(",");
68                 double lat = Double.parseDouble(pos[0].trim());
69                 double lng = Double.parseDouble(pos[1].trim());
70                 latLng = new LatLng(lat, lng);
71                 // 根據經緯度物件取得地址資料
```

```

72         address = Helper.getAddressByLatLng(latLng);
73         Toast.makeText(this, "地址：" + address, Toast.LENGTH_SHORT).show();
74         break;
75     }
76
77     // 清除地圖
78     mMap.clear();
79     // 新增 Marker
80     mMap.addMarker(new MarkerOptions().position(latLng).title(address));
81     // 在地圖上演示地圖相機動畫效果
82     playAnimateCamera(latLng, 3000);
83
84 }
85
86 // 在地圖上演示地圖相機動畫效果
87 private void playAnimateCamera(LatLng latLng, int durationMs) {
88     // 設置相關地圖相機位置參數，其中 zoom 的設定要 >=17 才會顯示建築物
89     CameraPosition cameraPos = new CameraPosition.Builder().target(latlng)
90         .zoom(17.0f).bearing(300).tilt(45).build();
91     // 定義地圖相機移動
92     CameraUpdate cameraUpt = CameraUpdateFactory
93         .newCameraPosition(cameraPos);
94     // 地圖相機動畫行程設定
95     mMap.animateCamera(cameraUpt, durationMs, null);
96 }
97 }
```

重點說明

程式第 62 行：

```
latLng = Helper.getLatLngByAddress(address);
```

根據地址資料取得經緯度物件

程式第 67~70 行：

```
String[] pos = editText2.getText().toString().split(",");
double lat = Double.parseDouble(pos[0].trim());
double lng = Double.parseDouble(pos[1].trim());
latLng = new LatLng(lat, lng);
```

程式第 72 行：

```
address = Helper.getAddressByLatLng(latLang);
```

根據經緯度物件取得地址資料

程式第 82 行：

```
playAnimateCamera(latLang, 3000);
```

在地圖上顯示地圖相機動畫效果

➡ 執行結果



➡ 設置地界 Setting boundaries

LatLang 物件可以用來封裝一個經緯位置其基本上就是一個點，倘若希望在某一塊區域中查詢到有幾個加油站或餐廳等這不是一個點而是一個面的問題，解決方法就是要有地界（區域）的資訊，地界資訊的封裝可以利用 **LatLangBounds** 來實現，範例如下：

```
GoogleMap mMap = ...;
LatLangBounds location = new LatLangBounds(
    new LatLang(24.989513, 121.312015), // 地界西南點的經緯度 LatLang 物件
    new LatLang(24.990571, 121.313075)); // 地界東北點的經緯度 LatLang 物件
mMap.moveCamera(CameraUpdateFactory.newLatLangZoom(location.getCenter(), 17));
```

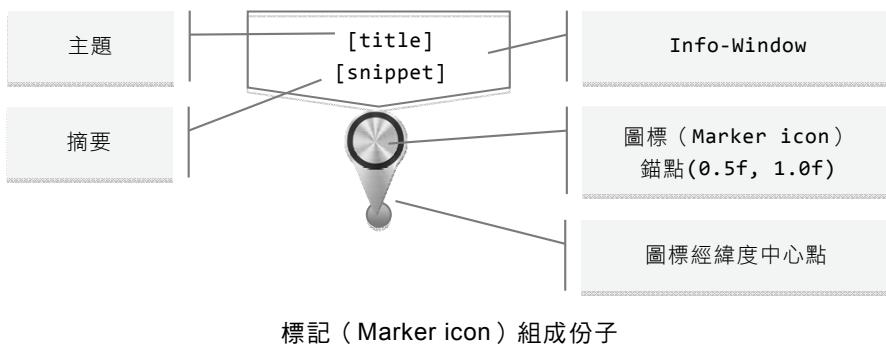
設置地界範例：



14-3-3 Marker (地圖標記)

之前我們已經了解如何將地圖導入到專案裡，不過若只有單純地圖呈現是無法明確告知使用者我們所要表達的是什麼，此時若能在地圖上做標記 (Marker icon) 且又不傷害地圖本身，讓使用者能清楚明白應該要在意與關注的事項，不失為一貼心的小幫手。

Google Map 上的標記組成份子：



標記 (Marker icon) 組成份子

► 範例程式 (CH14_04_Marker) 相關 Java API

public static interface GoogleMap.OnMarkerClickListener

標記選項按壓事件監聽器，設置該監聽器可調用

`setOnMarkerClickListener()`方法。

public static interface GoogleMap.OnMarkerDragListener

標記選項按壓拖曳監聽器，設置該監聽器可調用 `setOnMarkerDragListener()`

方法。

public static interface GoogleMap.OnInfoWindowClickListener

標記選項之訊息視窗按壓事件監聽器，設置該監聽器可調用

`setOnInfoWindowClickListener()`方法。

public final class MarkerOptions

定義標記選項資訊，一般常用的設定與說明如下：

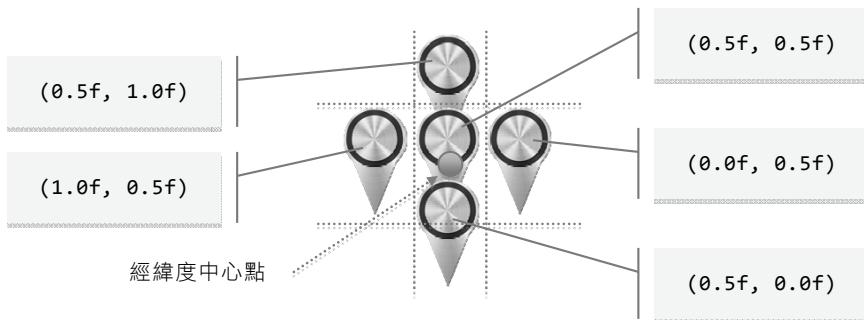
```
MarkerOptions options = new MarkerOptions(); // 建立標記選項實例
options.position(latlng); // 設定標記經緯度
options.title("桃園火車站"); // 設定 Info-Window 標題
options.snippet("緯經度：" + latlng); // 設定 Info-Window 標記摘要
options.anchor(0.5f, 1.0f); // 設定錨點
options.draggable(true); // 是否可以拖曳標記?
```

// 將 `BitmapDescriptor` 物件傳遞至 `icon()` 方法，即可自訂預設標記影像的色彩。

```
// 您可以使用 BitmapDescriptorFactory 物件中的一組預先定義色彩，  
// 或使用 BitmapDescriptorFactory.defaultMarker(float hue) 方法設定自訂標記色彩。  
options.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_ORANGE));
```

public MarkerOptions anchor(float u, float v)

配置圖標影像相對於圖標經緯度中心之錨點 (u, v)，u 表示圖像寬度的比率，v 表示圖像高度的比率，其相關設定位置與 (u, v) 值請參考以下範例：



anchor (u, v) 設定位置圖解範例說明

範例程式 (Map/app_youbikemap)：多點標記，新北市 Youbike 站場一覽圖。
新北市政府資料開放平台 - Youbike 開放資料查詢位置：

Google 關鍵字查詢：「YouBike - 新北市政府資料開放平台」

資料集名稱	新北市公共自行車租賃系統(YouBike)	平均評分 :	★★★★★	評分人數 : 9
資料集描述	為提升市民使用新北市公共自行車租賃系統(YouBike)服務品質，透過資料透明化，提供各資訊平台介接取得系統服務即時資訊，滿足市民第一哩及最後一哩公共運輸需求。			
主題分類	交通			
圖說會分類	交通及通訊			
主要欄位說明	sno:站點代號、sna:站點名稱(中文)、tot:場站總停車格、sbi:可借車位數、sarea:場站區域(中文)、mday:資料更新時間、lat:緯度、lng:經度、ar:地址(中文)、sareen:場站區域(英文)、snaen:站點名稱(英文);aren:地址(英文);bemp:可還空位數、act:場站是否暫停營運			
資料集說明檔案				
授權方式	本資料集採用政府資料開放授權修款-第1版授權。			
更新頻率	每5分鐘			
資料量	383			
全檔下載	ZIP	JSON	XML	CSV
API介接	請參考開發指引			
更多資訊				

API 介接「JSON」網址：

[http://data.ntpc.gov.tw/od/data/api/54DDDC93-589C-4858-9C95-18B2046CC1FC?\\$format=json](http://data.ntpc.gov.tw/od/data/api/54DDDC93-589C-4858-9C95-18B2046CC1FC?$format=json)

```
← → C ⓘ data.ntpc.gov.tw/od/data/api/54DDDC93-589C-4858-9C95-18B2046CC1FC?$format=json
```

```
[  
- {  
    sno: "1001",  
    sna: "大鵬華城",  
    tot: "38",  
    sbi: "14",  
    sareas: "新店區",  
    mday: "20170611183921",  
    lat: "24.99116",  
    lng: "121.53398",  
    ar: "新北市新店區中正路700巷3號",  
    sareasen: "Xindian Dist.",  
    snaen: "Dapeng Community",  
    aren: "No. 3, Lane 700 Chung Cheng Road, Xindian District",  
    bemp: "24",  
    act: "1"  
},  
- {  
    sno: "1002",  
    sna: "汐止火車站",  
    tot: "56",  
    sbi: "36",  
    sareas: "汐止區",  
    mday: "20170611183933",  
    lat: "25.068914",  
    lng: "121.662748",  
    ar: "南昌街/新昌路口(西側廣場)",  
    sareasen: "Xizhi Dist.",  
    snaen: "Xizhi Railway Station",  
    aren: "Nanchang St./Xinchang Rd.",  
    bemp: "19",  
    act: "1"  
},  
- {  
    sno: "1003",  
    sna: "汐止區公所",  
    tot: "46",  
    sbi: "27",  
    sareas: "汐止區",  
    mday: "20170611183933",  
    lat: "25.064162",  
    lng: "121.658301",  
    ar: "新台五路一段/仁愛路口(新台五路側汐止地政事務所前機車停車場)"  
}
```

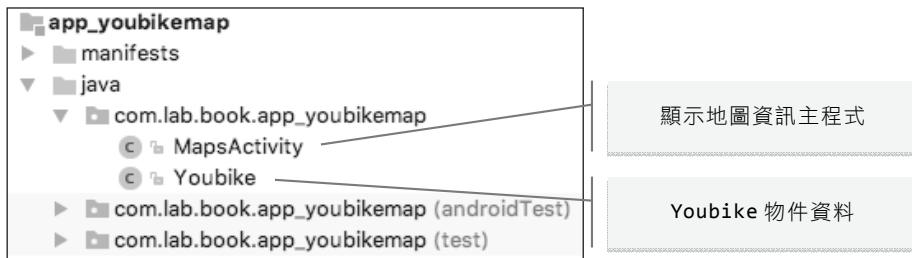


➥ 主要欄位說明

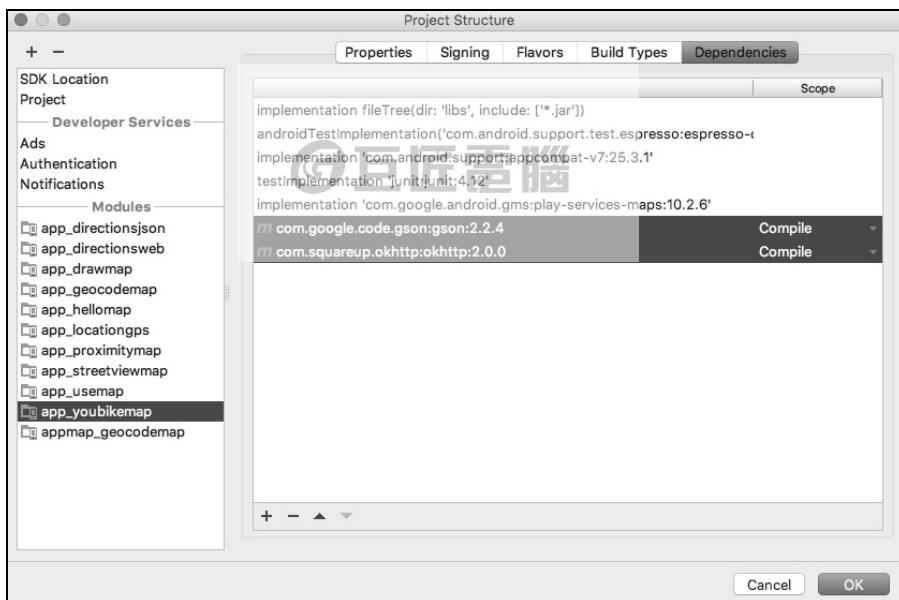
sno:站點代號、**sna:**場站名稱(中文)、**tot:**場站總停車格、**sbi:**可借車位數、
sarea:場站區域(中文)、**mday:**資料更新時間、**lat:**緯度、**lng:**經度、**ar:**地址

(中文)、**sareaen**:場站區域(英文)、**snaen**:場站名稱(英文):**aren**、地址(英文):**bemp**:可還空位數、**act**:場站是否暫停營運。

◆ 範例程式 (Map/app_youbikemap) 程式碼類別與功能說明



使用 Library : OKHttp 與 GSON



 重點程式碼 (1/2)：Youbike.java 用來封裝 Youbike 站場資訊**Youbike.java**

```
.. ...
05 public class Youbike {
06
07     private String sno; //:站點代號
08     private String sna; //:場站名稱(中文)
09     private String tot; // :場站總停車格
10     private String sbi; //:可借車位數
11     private String sarea;//場站區域(中文)
12     private String mday; // 資料更新時間
13     private String lat; // 緯度
14     private String lng; //經度
15     private String ar; // 地址(中文)
16     private String sareen; // 場站區域(英文);
17     private String snaen; //:場站名稱(英文);
18     private String aren; //地址(英文)
19     private String bemp; //可還空位數
20     private String act; //場站是否暫停營運
21     // setter / getter
22     ...
23 }
```

 重點程式碼 (1/2)：主程式**MapsActivity.java**

```
.. ...
19 public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {
20
21     private GoogleMap mMap;
22
23     @Override
24     protected void onCreate(Bundle savedInstanceState) {
25         super.onCreate(savedInstanceState);
26         setContentView(R.layout.activity_maps);
27         SupportMapFragment mapFragment =
28             (SupportMapFragment) getSupportFragmentManager()
29             .findFragmentById(R.id.map);
30         mapFragment.getMapAsync(this);
31     }
32
33     @Override
```

```

34     public void onMapReady(GoogleMap googleMap) {
35         mMap = googleMap;
36         new RunYoubikeWork().start();
37     }
38
39     class RunYoubikeWork extends Thread {
40         Youbike[] youbikes = null;
41
42         OkHttpClient client = new OkHttpClient();
43
44         String run(String url) throws IOException {
45             Request request = new Request.Builder()
46                 .url(url)
47                 .build();
48
49             Response response = client.newCall(request).execute();
50             return response.body().string();
51         }
52
53         Runnable r = new Runnable() {
54             @Override
55             public void run() {
56                 mMap.clear();
57                 for(Youbike bike : youbikes) {
58                     double lat = Double.parseDouble(bike.getLatitude());
59                     double lng = Double.parseDouble(bike.getLongitude());
60                     LatLng latLng = new LatLng(lat, lng);
61
62                     MarkerOptions options = new MarkerOptions(); // 建立標記選項實例
63                     options.position(latLng); // 設定標記經緯度
64                     options.title(bike.getStationName()); // 設定 Info-Window 標題
65                     // 設定 Info-Window 標記摘要
66                     options.snippet("可借車位數：" + bike.getAvailableBikes());
67                     // 將 BitmapDescriptor 物件傳遞至 icon() 方法，
68                     // 即可自訂預設標記影像的色彩。
69                     // 您可以使用 BitmapDescriptorFactory 物件中的一組預先定義色彩，
70                     // 或使用 BitmapDescriptorFactory.defaultMarker(float hue) 方法
71                     // 來設定自訂標記色彩。
72                     options.icon(BitmapDescriptorFactory
73                         .defaultMarker(BitmapDescriptorFactory.HUE_ORANGE));
74                     mMap.addMarker(options);
75                     mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng, 12));
76                 }
77             }

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14

```
78     };  
79  
80     @Override  
81     public void run() {  
82         try {  
83             String urlstring = getResources().getString(R.string.url);  
84             String json = run(urlstring);  
85             youbikes = new Gson().fromJson(json, Youbike[].class);  
86             runOnUiThread(r);  
87         } catch (Exception e) {  
88             }  
89         }  
90     }  
91 }
```

重點說明

程式第 84 行：

```
String json = run(urlstring);
```

透過 `OKHttp` 的 `run(string)` 方法來取得 `Youbike` `Json` 字串。

程式第 85 行：

```
youbikes = new Gson().fromJson(json, Youbike[].class);
```

利用 `Gson` 的 `fromJson()` 方法，將陣列型 `Json` 字串轉成 `Youbike` 陣列。

程式第 57~76 行：

透過 `for-each` 將每一個 `Youbike` 站點標示在 `Map` 上，並於 `Marker` 中揭露站場名稱(`sna`)與可借車位數(`sbi`)。

↓ 執行結果



14-3-4 Draw Map (地圖塗鴉)

除了可在 Google map 上建立圖標 (Marker icon) 外，本章將進一步說明如何在 Google map 上塗鴉 (Drawing)。



► 範例程式（Map/app_drawmap）相關 Java API

```
public final class PolylineOptions
```

定義單一與多連續線段的類別並調用 GoogleMap 物件上的 addPolyline()方法將所定義圖形繪製在地圖上。

繪製單一線段：

```
private void drawPolyLine() {  
    LatLng p1 = new LatLng(24.990194, 121.311767);  
    LatLng p2 = new LatLng(24.989206, 121.313549);  
  
    PolylineOptions options = new PolylineOptions();  
    options.add(p1, p2);  
    options.width(10); // 線條寬度  
    options.color(Color.MAGENTA); // 顏色  
    options.zIndex(1); // 設定疊層位置  
  
    mMap.addPolyline(options);  
}
```

zIndex() 設定疊層位置，數值越高越往上疊，index 數字低的疊層將會被數字高的疊層蓋住

繪製多點連續線段：

```
private void drawPolyLineAll() {  
    // 製作 LatLng 物件集合  
    List<LatLng> points = Arrays.asList(  
        new LatLng(24.99159, 121.31449),  
        new LatLng(24.99108, 121.31497),  
        new LatLng(24.99059, 121.31548),  
        new LatLng(24.99046, 121.31558),  
        new LatLng(24.99028, 121.31576),  
        new LatLng(24.99024, 121.3158),  
        new LatLng(24.99018, 121.31588),  
        new LatLng(24.99014, 121.31595),  
        new LatLng(24.99008, 121.31606));  
  
    PolylineOptions options = new PolylineOptions();  
    options.addAll(points); // 將 LatLng 物件集合注入，成為繪製的座標資料
```

```
    options.width(10);
    options.color(Color.RED);
    options.zIndex(1);

    mMap.addPolyline(options);
}
```

public final classPolygonOptions

定義多邊形的類別並調用 `GoogleMap` 物件上的 `addPloygon()`方法將所定義圖形繪製在地圖上。

繪製多邊形：

```
private void drawPolygon() {
    LatLng p1 = new LatLng(24.990194, 121.311767);
    LatLng p2 = new LatLng(24.989513, 121.312015);
    LatLng p3 = new LatLng(24.989916, 121.313306);
    LatLng p4 = new LatLng(24.990571, 121.313075);

    PolygonOptions options = new PolygonOptions();
    options.add(p1, p2, p3, p4);
    options.strokeWidth(5);
    options.strokeColor(Color.BLUE);
    options.fillColor(Color.argb(200, 100, 150, 0));
    options.zIndex(1);

    mMap.addPolygon(options);
}
```

public final classCircleOptions

定義圓形的類別並調用 `GoogleMap` 物件上的 `addCircle()`方法將所定義圖形繪製在地圖上。

繪製圓型：

```
private void drawCircle() {
    LatLng latlng = new LatLng(24.989206, 121.313549);
    // 以 latlng 位置為中心畫圓
    CircleOptions options = new CircleOptions();
    options.center(latlng); // 圓心位置
    options.radius(100); // 半徑(公尺)
```

```
options.strokeWidth(5); // 圓形外框寬度
options.strokeColor(Color.TRANSPARENT); // 圓形外框顏色
options.fillColor(Color.argb(150, 255, 0, 0));
options.zIndex(1); // 若有疊圖發生，疊層 id 數字越高則圖層越上層

mMap.addCircle(options);
}
```

範例程式 (Map/app_drawmap)

在 Google map 上繪製自製圖形，包含線段、多邊形、圓形繪製與路徑繪製（含 kml。基本上路徑繪製就是線段繪製的多點連續組合）。

重點程式碼

MapsActivity.java

```
.. ...
28 public class MapsActivity extends AppCompatActivity
29         implements OnMapReadyCallback {
30     private GoogleMap mMap;
31     @Override
32     protected void onCreate(Bundle savedInstanceState) {
33         super.onCreate(savedInstanceState);
34         setContentView(R.layout.activity_maps);
35         SupportMapFragment mapFragment =
36             (SupportMapFragment) getSupportFragmentManager()
37             .findFragmentById(R.id.map);
38         mapFragment.getMapAsync(this);
39     }
40     @Override
41     public void onMapReady(GoogleMap googleMap) {
42         mMap = googleMap;
43         LatLng latLng = new LatLng(24.989206, 121.313549);
44         mMap.addMarker(new MarkerOptions()
45             .position(latLng).title("台灣桃園火車站"));
46         mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng, 17));
47     }
48     @Override
49     public boolean onCreateOptionsMenu(Menu menu) {
50         MenuItem item1 = menu.add(0, 1, 1, "畫圓");
51         MenuItem item2 = menu.add(0, 2, 2, "畫線");
52         MenuItem item3 = menu.add(0, 3, 3, "畫多邊形");
```

```
53     MenuItem item4 = menu.add(0, 4, 4, "連續線段");
54     MenuItem item5 = menu.add(0, 5, 5, "連續線段-KML");
55     MenuItem exit  = menu.add(0, 9, 9, "離開");
56     exit.setShowAsAction(MenuItem.SHOW_AS_ACTION_ALWAYS);
57     return true;
58 }
59 @Override
60 public boolean onOptionsItemSelected(MenuItem item) {
61     mMap.clear();
62     switch (item.getItemId()) {
63         case 1:
64             drawCircle();
65             break;
66         case 2:
67             drawPolyLine();
68             break;
69         case 3:
70             drawPolygon();
71             break;
72         case 4:
73             drawPolyLineAll();
74             break;
75         case 5:
76             drawPolyLineKML();
77             break;
78         case 9:
79             finish();
80             break;
81     }
82     setTitle(item.getTitle());
83     return super.onOptionsItemSelected(item);
84 }
85
86 // 畫圓
87 private void drawCircle() {
88     // 略...
89 }
90
91 // 畫單一線段
92 private void drawPolyLine() {
93     // 略...
94 }
95
96 // 畫多邊形
97
```

```
115     private void drawPolygon() {  
116         ..  
117         // 略...  
118     }  
119  
120     // 畫連續線段  
121     private void drawPolyLineAll() {  
122         ..  
123         // 略...  
124     }  
125  
126     // 畫連續線段-KML  
127     private void drawPolyLineKML() {  
128         List<LatLng> points = new ArrayList<LatLng>();  
129         try {  
130             // 取得 res\router.kml 資源  
131             InputStream inStream =  
132                 getResources().openRawResource(R.raw.router);  
133             // 建立 DOM 實例  
134             DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();  
135             DocumentBuilder db = dbf.newDocumentBuilder();  
136             // 取得 DOM 文件  
137             Document doc = db.parse(inStream);  
138             // 找到 "coordinates" XML 節點元素  
139             NodeList nl = doc.getElementsByTagName("coordinates");  
140             if(nl.getLength() == 0) {  
141                 return;  
142             }  
143             // 取得節點  
144             Node node = nl.item(0);  
145             // 分析節點元素內容  
146             String[] routers = node.getTextContent().trim().split(" ");  
147             for(String router : routers) {  
148                 String[] r = router.split(",");  
149                 // 將經緯度物件加入到 points 集合中  
150                 points.add(new LatLng(Double.parseDouble(r[1]),  
151                               Double.parseDouble(r[0])));  
152             }  
153         } catch (Exception e) {  
154             e.printStackTrace(System.out);  
155             return;  
156         }  
157         PolylineOptions options = new PolylineOptions();  
158         options.addAll(points);  
159         options.width(10);  
160     }
```

```
185         options.color(Color.RED);
186         options.zIndex(1);
187         mMap.addPolyline(options);
188     }
189 }
```

➔ 重點說明

程式 151~188 行將.kml 檔案³資訊所描繪的經緯度資訊繪製在地圖上，同樣是調用 `addAll()` 的方法來達成。

程式第 158~176 行利用 DOM 來剖析 `R.raw.router` (KML 檔案的格式是 XML) 並取得經緯度集合資料，.kml 檔案中經緯度的資料集合是位於 `coordinates` 節點內。

`<coordinates>` XML 節點經緯度內容擺放順序：

```
<coordinates>
    121.3114136219668,24.99003820138407,0
    121.3114500859793,24.99003844737789,0
    121.3114864549488,24.99004539170917,0
    ...
</coordinates>
```

最後在程式第 187 行調用 `addPolyline()` 方法，將所有的「點」透過線段繪出。

³ KML 全稱：Keyhole Markup Language，是基於 XML(eXtensible Markup Language, 可擴展標記語言)語法標準的一種標記語言 (markup language)，採用標記結構，含有嵌套的元素和屬性。由 Google(谷歌)旗下的 Keyhole 公司發展並維護，用來表達地理標記。根據 KML 語言編寫的文件則為 KML 文件，格式同樣採用 XML 文件格式，應用於 Google 地球相關軟體中 (Google Earth, Google Map, Google Maps for mobile...)，用於顯示地理數據(包括點、線、面、多邊形，多面體以及模型...)。而現在很多 GIS 相關企業也追隨 Google 開始採用此種格式進行地理數據的交換。另外與.kml 檔相關的還有一個稱為.kmz 檔，.kmz 檔就是多組.kml 的壓縮集合。

在 Google Earth 上繪製路徑來產生.kml 檔。



執行結果



畫線



畫多邊形



連續線段

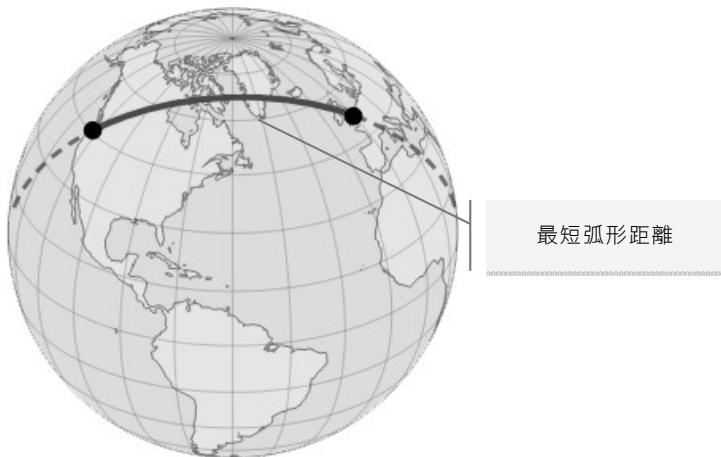


連續線段 - KML



二經緯度坐標之大圓距離

在 Google Map 上任意二經緯度坐標間的距離計算並非為平面坐標上單純二點間的平面距離，因為地球是圓的所以應該於球面坐標來取得**最短**弧形線段的距離做為結果，如下圖所示。



球面坐標上任意二點之最短弧形線段

從球面的一點 A 出發到達球面上另一點 B，所經過的最短路徑的長度稱為**大圓距離**。由於經緯度計算不是平面坐標而是球面坐標，以下是圓球體大圓距離的計算公式：

從弦長求大圓距離（資料來源：[維基百科](#)）

連結球面上兩點之間的線段就是這兩點所在大圓上兩點之間的弦，這條弦所對的圓心角可通過幾何關係求出，然後再通過弧長公式求出這條弧的弧長，即兩點間的大圓距離。

令 $\phi_s, \lambda_s; \phi_f, \lambda_f$ 分別代表球面上兩點的經緯度，(s 代表出發點，f 代表前往點)

$$\Delta X = \cos(\phi_f) \cos(\lambda_f) - \cos(\phi_s) \cos(\lambda_s);$$

$$\Delta Y = \cos(\phi_f) \sin(\lambda_f) - \cos(\phi_s) \sin(\lambda_s);$$

$$\Delta Z = \sin(\phi_f) - \sin(\phi_s);$$

$$C_h = \sqrt{(\Delta X)^2 + (\Delta Y)^2 + (\Delta Z)^2}$$

$$\Delta\hat{\sigma} = 2 \arcsin \left(\frac{C_h}{2} \right) .$$

圓心角

$$d = r \Delta\hat{\sigma}.$$

大圓距離

➔ Java 實作程式碼

```
private double distanceBetween(double startLatitude, double startLongitude,
    double endLatitude, double endLongitude) {
    double R = 6371; // 地球平均半徑 = 6,371km
    double dlat = (endLatitude - startLatitude) * Math.PI / 180;
    double dlon = (endLongitude - startLongitude) * Math.PI / 180;
    double aDouble = Math.sin(dlat / 2) * Math.sin(dlat / 2)
        + Math.cos(startLatitude * Math.PI / 180)
        * Math.cos(endLatitude * Math.PI / 180) * Math.sin(dlon / 2)
        * Math.sin(dlon / 2);
    double cDouble = 2 * Math.atan2(Math.sqrt(aDouble),
        Math.sqrt(1 - aDouble));
    double d = (R * cDouble) * 1000;
    return d;
}
```

除了自行實作大圓距離公式之外，也可以調用 `Location.distanceBetween()` 來求得二經緯度坐標之大圓距離，其 API 定義如下：

```
public static void distanceBetween(double startLatitude, double startLongitude,
    double endLatitude, double endLongitude, float[] results)
```

計算二經緯度坐標之大圓距離並保存結果。

➔ 參數說明

起點經緯度值：`startLatitude, startLongitude`

終點經緯度值：`endLatitude, endLongitude`

保存計算結果：`results`

計算二經緯坐標（25.04, 121.54）與（25.04, 121.51）之大圓距離：

```
// 調用 Location.distanceBetween API 求得大圓距離
float[] results = new float[1]; // 大圓距離計算結果放置處
Location.distanceBetween(25.04, 121.54, 25.04, 121.51, results);
System.out.println(results[0]);
// 從弧長公式求得大圓距離
System.out.println(distanceBetween(25.04, 121.54, 25.04, 121.51));
```

保存計算結果

⬇ 執行結果（單位：公尺）

3027.5215

3022.3198748133

任何計算大圓距離的公式所得結果一定是近似值，當然就造就出從不同角度衍生出的許多演算法與公式，因此不同演算法間的計算結果一定都會有誤差，不過唯一相同的是這些公式與演算法最終的計算結果一定都是逼近於真值！

14-4 ProximityAlert 接近偵測提醒

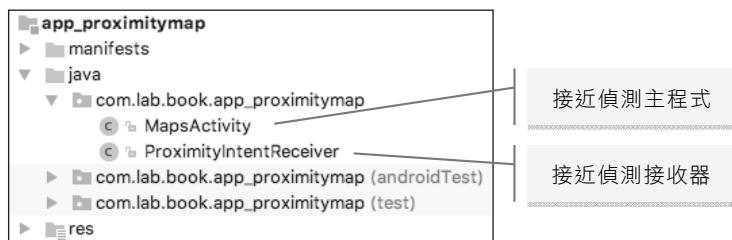
LocationManager 提供了 `addProximityAlert()` 方法讓開發人員可以透過此方法來註冊偵測區域用以監聽所有移動坐標針對該地界的進入與離開事件。



接近偵測移入移出示意圖

範例（Map/app_proximitymap）說明了接近偵測的使用方式

⬇ 程式碼類別與功能說明



app_proximitymap 範例類別說明

範例程式 (Map/app_proximitymap)：接近偵測

使用者利用模擬器輸入最新經緯度並觀察該經緯度是否進入（離開）偵測範圍。

↓ 執行結果



[已進入模式]



[已離開模式]

透過模擬器輸入經緯度偵測「已進入...」與「已離開...」，中間請間隔至少 5 秒鐘。

➔ 重點程式碼 1/2

接近偵測接收器：ProximityIntentReceiver.java

ProximityIntentReceiver.java

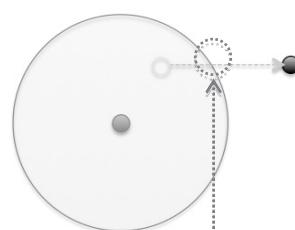
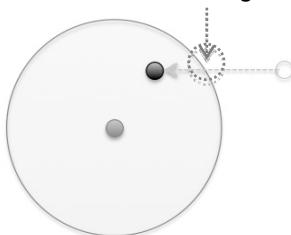
```
..  
09  public class ProximityIntentReceiver extends BroadcastReceiver {  
10  
11     @Override  
12     public void onReceive(Context context, Intent intent) {  
13  
14         String key = LocationManager.KEY_PROXIMITY_ENTERING;  
15  
16         Boolean entering = intent.getBooleanExtra(key, false);  
17  
18         if (entering) {  
19             System.out.println("已進入...");  
20             Toast.makeText(context, "已進入...", Toast.LENGTH_SHORT).show();  
21         }  
22         else {  
23             System.out.println("已離開...");  
24             Toast.makeText(context, "已離開...", Toast.LENGTH_SHORT).show();  
25         }  
26     }  
27 }
```

取得接近偵測的廣播 boolean 參數

➔ 重點說明

程式第 14~16 行利用接近偵測的廣播 boolean 參數（**LocationManager.KEY_PROXIMITY_ENTERING**）用以判斷特定實點是否進入或離開所偵測的區域。

移入發生：**LocationManager.KEY_PROXIMITY_ENTERING=true**



移出發生：**LocationManager.KEY_PROXIMITY_ENTERING=false**

接近偵測移入移出廣播觸發之參數設置示意圖

接近偵測觸發測試注意：第一次執行時若給予的經緯度實點是落在偵測範圍外，則接近偵測服務不會有任何動作，唯有當給予的經緯度實點有一次是落在偵測範圍內（移入事件觸發），接下來接近偵測機制就會進行實點進入/移出事件的廣播監控。

👉 重點程式碼 2/2

MapsActivity.java

```
.. ...
26 public class MapsActivity extends FragmentActivity
27     implements OnMapReadyCallback {
28     private static final int GPS_REQUEST_CODE = 101;
29     // 最短距離(公尺單位)
30     private final long MINIMUM_DISTANCECHANGE_FOR_UPDATE = 1;
31     // 最小時間(微秒單位)
32     private final long MINIMUM_TIME_BETWEEN_UPDATE = 1000;
33     // 預設地界半徑公尺單位
34     private final long POINT_RADIUS = 100;
35     // 預設地界中心坐標
36     private final LatLng DEFAULT_LTS = new LatLng(25.0417443, 121.5503917);
37     // 偵測有效期限, -1 表示永久有效
38     private final long PROX_ALERT_EXPIRATION = -1;
39     private final String PROX_ALERT_INTENT = "This.is.my.ProximityAlert";
40     private final NumberFormat nf = new DecimalFormat("###,###.##");
41     private GoogleMap mMap;
42     private LocationManager locationManager;
43     private ProximityIntentReceiver receiver;
44
45     @Override
46     protected void onCreate(Bundle savedInstanceState) {
47         super.onCreate(savedInstanceState);
48         setContentView(R.layout.activity_maps);
49         SupportMapFragment mapFragment =
50             (SupportMapFragment) getSupportFragmentManager()
51             .findFragmentById(R.id.map);
52         mapFragment.getMapAsync(this);
53     }
54
55     @Override
56     public void onMapReady(GoogleMap googleMap) {
57         mMap = googleMap;
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14

```
58     locationManager =
59         (LocationManager) getSystemService(Context.LOCATION_SERVICE);
60     // 接近偵測設置
61     addProximityAlert();
62     // 繪製偵測範圍
63     drawCircle();
64     // 移動到指定偵測範圍坐標中心
65     mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(DEFAULT_TTS, 17));
66 }
67
68 // 設置/加入接近偵測警示
69 private void addProximityAlert() {
70     // 確認是否之前已經授權過此認證
71     if (ContextCompat.checkSelfPermission(this,
72             ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
73         // LocationManager 設置
74         locationManager.requestLocationUpdates(
75             LocationManager.GPS_PROVIDER,
76             MINIMUM_TIME_BETWEEN_UPDATE,
77             MINIMUM_DISTANCECHANGE_FOR_UPDATE,
78             new MyLocationListener()
79         );
80         // 建立廣播 PendingIntent
81         Intent intent = new Intent(PROX_ALERT_INTENT);
82         PendingIntent proximityIntent = PendingIntent.getBroadcast(
83             this, 0, intent, 0);
84         // 加入接近偵測警示
85         locationManager.addProximityAlert(
86             DEFAULT_TTS.latitude, // 地界中心點緯度坐標
87             DEFAULT_TTS.longitude, // 地界中心點經度坐標
88             POINT_RADIUS, // 地界半徑(公尺)
89             PROX_ALERT_EXPIRATION, // 偵測有效期限, -1 表示永久有效
90             proximityIntent // 偵測到移入或移出時所應觸發的 intent
91         );
92
93         // 註冊接近偵測接收器
94         IntentFilter filter = new IntentFilter(PROX_ALERT_INTENT);
95         receiver = new ProximityIntentReceiver();
96         registerReceiver(receiver, filter);
97     } else { // 若之前沒有授權過
98         // 彈跳出授權小視窗，讓使用者當場授權
99         ActivityCompat.requestPermissions(this,
100             new String[]{ACCESS_FINE_LOCATION},
101             GPS_REQUEST_CODE // 權限請求碼
102         );
103     }
104 }
```

```
102         );
103     }
104     // 檢查 GPS 是否有開啟 ?
105     openGPS();
106 }
107
108 // 授權回呼結果 (授權驗證 SDK >= 23 (Android 6.0))
109 @Override
110 public void onRequestPermissionsResult(int requestCode,
111                                         String[] permissions, int[] grantResults) {
112     switch (requestCode) {
113         case GPS_REQUEST_CODE: // 權限請求碼
114             if (grantResults.length > 0
115                 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
116                 setTitle("授權驗證成功 ...");
117                 addProximityAlert();
118             } else {
119                 setTitle("授權驗證失敗 ...");
120             }
121         }
122     }
123
124 // 與地界中心的距離(公尺), 用以計算偵測半徑
125 private float getDistanceBetween(LatLng latlng) {
126     float[] results = new float[1];
127     Location.distanceBetween(DEFAULT_TTS.latitude, DEFAULT_TTS.longitude,
128                             latlng.latitude, latlng.longitude, results);
129     return results[0];
130 }
131
132 // 繪製地界
133 private void drawCircle() {
134     CircleOptions options = new CircleOptions();
135     options.center(DEFAULT_TTS);
136     options.radius(POINT_RADIUS);
137     options.strokeWidth(5);
138     options.strokeColor(Color.TRANSPARENT);
139     options.fillColor(Color.argb(100, 0, 0, 255));
140     options.zIndex(1);
141     mMap.addCircle(options);
142 }
143
144 // 繪製標記
145 private void drawMarkers(LatLng latlng) {
```

```
146     // 建立 Marker
147     MarkerOptions mo = new MarkerOptions();
148     mo.position(latlng);
149     mo.title("新地點");
150     mo.snippet("緯經度：" + latlng);
151     mo.draggable(true);
152     // 將 Marker 加入到地圖中
153     mMap.addMarker(mo);
154 }
155
156 @Override
157 protected void onDestroy() {
158     // 卸載接近偵測接收器
159     if (receiver != null) {
160         unregisterReceiver(receiver);
161     }
162     super.onDestroy();
163 }
164
165 // 開啟 GPS
166 public void openGPS() {
167     boolean gps = locationManager
168         .isProviderEnabled(LocationManager.GPS_PROVIDER);
169     boolean network = locationManager
170         .isProviderEnabled(LocationManager.NETWORK_PROVIDER);
171     Toast.makeText(this, "GPS：" + gps + "，Network：" + network,
172                 Toast.LENGTH_SHORT).show();
173     if (gps || network) {
174         return;
175     } else {
176         // 開啟手動 GPS 設定畫面
177         Intent gpsOptionsIntent = new Intent(
178             android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS);
179         startActivityForResult(gpsOptionsIntent);
180     }
181 }
182
183 // Location 監聽器
184 public class MyLocationListener implements LocationListener {
185     public void onLocationChanged(Location location) {
186         // 清除地圖上所有物件
187         mMap.clear();
188         // 繪製標記
189         LatLng latlng = new LatLng(location.getLatitude(),
```

```
190         location.getLongitude());
191         drawMarkers(latlng);
192         // 重繪地界
193         drawCircle();
194         // 計算距離
195         float distance = getDistanceBetween(latlng);
196         setTitle("與地界中心距離：" + nf.format(distance) + " M");
197     }
198     public void onStatusChanged(String s, int i, Bundle b) {
199     }
200     public void onProviderDisabled(String s) {
201     }
202     public void onProviderEnabled(String s) {
203     }
204 }
205 }
```

👉 重點說明

程式第 61 行增加設置接近偵測方法 `addProximityAlert()`。

程式第 81 行建立一個廣播 `PendingIntent` 當有任何含經緯度的實點進入或移出接近偵測區域範圍時觸發。

程式第 125~130 行計算偵測半徑。

MapsActivity.java 中最重要的就是廣播通知與接受器之間的互動，透過接近偵測所設置的廣播通知（程式第 85~91 行）讓接近偵測接收器得以收到並作出適時的回應。

1
2
3
4
5
6
7
8
9
10
11
12
13
14

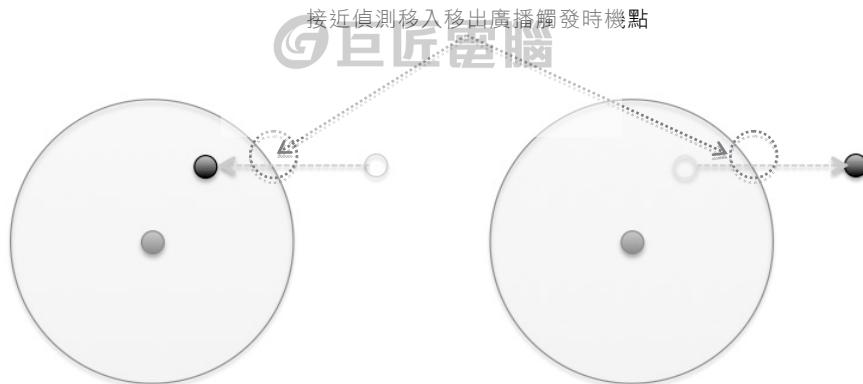
```
// 加入接近偵測警示
locationManager.addProximityAlert(
    latlng.latitude, // 地界中心點緯度坐標
    latlng.longitude, // 地界中心點經度坐標
    POINT_RADIUS, // 地界半徑(公尺)
    PROX_ALERT_EXPIRATION, // 偵測有效期限, -1 表示永久有效
    proximityIntent // 偵測到移入或移出時所應觸發的 intent
);
```

我是一個 PendingIntent 接近偵測廣播服務

```
PendingIntent proximityIntent = PendingIntent.getBroadcast(this, 0, intent, 0);
```

```
IntentFilter filter = new IntentFilter(PROX_ALERT_INTENT);
receiver = new ProximityIntentReceiver();
registerReceiver(receiver, filter);
```

我是一個接近偵測接收器



14-5 地圖與街景

Google 地圖的「街景服務」提供 360 度全景街頭影像，讓您探索世界各個角落。您可以逛逛各地的餐廳、規劃下一趟旅遊行程，甚至造訪瑞士的阿爾卑斯山或亞馬遜河！地圖服務中若加入了 Google 街景的加持，無疑是讓使用者有更好的使用體驗。拍攝街景因為地形上的差異在載具設備上也會有所不同，舉凡汽車、自行車、雪橇、人力、動物甚至是海底拍攝裝置都有，包羅萬象。

表：各種 Google 街景載具設備



值得注意的一點是，因為 Google 拍攝街景會有安全、國防與隱私權上的問題，目前 Google 街景僅提供數月前的資料，若是敏感資訊則會利用模糊或降低解析度的方式來避免紛爭。在 Google 靜態街景圖上可以看到拍攝年份，如下圖所示：



臺北火車站前 Google 靜態街景圖 (25.046254,121.517532)

Map/app_streetviewmap 範例將說明 Google 地圖配合呼叫 Google 街景的應用提高使用者經驗。



➔ 範例程式 (Map/app_streetviewmap) 相關 API

取得 Google 動態街景圖 URI API。

"google.streetview:cbll=緯度,經度&mz=縮放"

範例（臺北火車站前的動態街景圖）：

```
// 取得街景
String path = "google.streetview:cbll=" +
    "25.046254,121.517532" +
    "&mz=21";

Uri uri = Uri.parse(path);
Intent intent = new Intent(android.content.Intent.ACTION_VIEW, uri);
startActivity(intent);
```

取得 Google 靜態街景影像 Web API (Google Street View Image API)。

<http://maps.googleapis.com/maps/api/streetview?parameters>
parameters 必要參數

size：設置影像大小（單位：像素），`width * height`

location：可以是字串（地址或地標）或經緯度

範例（臺北火車站前的靜態街景圖）：

`http://maps.googleapis.com/maps/api/streetview?size=450x250&location=25.046254,121.517532`



其它 **parameters** 選用參數包含：`heading`、`fov`、`pitch` 與 `key` 的用法請參考

<https://developers.google.com/maps/documentation/streetview/>。

API Key 金鑰：用以識別您的應用程式，可以於 Google API Console 中申請「Street View Image API」並加入到 **parameter** 參數中來追蹤應用程式使用該服務之狀況，必要時可以購買配額（每日預設配額 `1,000,000 requests/day`）。

public final class GroundOverlayOptions

定義地圖上的區域疊層（又名疊加層），例如：在地圖上放置縮略圖。

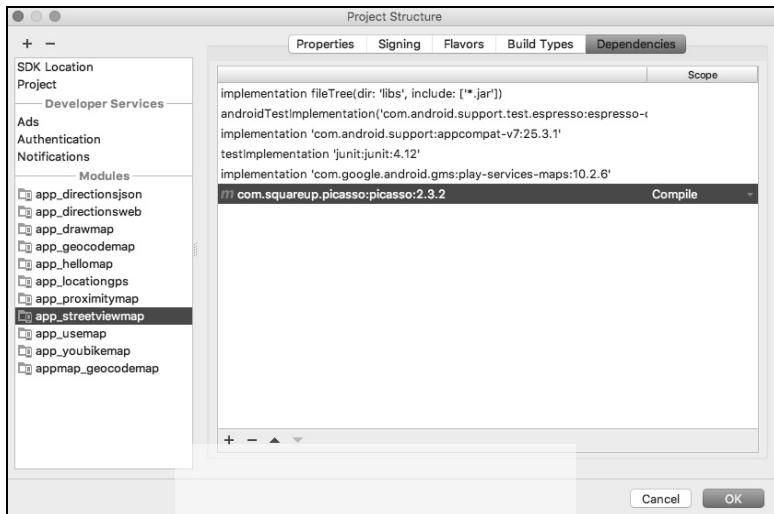
public final GroundOverlay addGroundOverlay (GroundOverlayOptions options)

在地圖上增加一個疊層，例如：影像 `image`。

範例程式 (Map/app_streetviewmap)

Google 地圖與 Google 街景的搭配應用。

使用 Library : Picasso



執行結果 (實機測試)



➔ 重點程式碼

MapsActivity.java

```
.. ...
30  public class MapsActivity extends FragmentActivity
31          implements OnMapReadyCallback {
32      private final String ADDRESS = "台北市忠孝東路四段 169 號";
33      private final LatLng latLng = new LatLng(25.0415471,121.5512709);
34      private GoogleMap mMap;
35      private Context context;
36      private GroundOverlay groundOverlay;
37      @Override
38      protected void onCreate(Bundle savedInstanceState) {
39          super.onCreate(savedInstanceState);
40          setContentView(R.layout.activity_maps);
41          context = this;
42          SupportMapFragment mapFragment =
43              (SupportMapFragment) getSupportFragmentManager()
44                  .findFragmentById(R.id.map);
45          mapFragment.getMapAsync(this);
46      }
47      @Override
48      public void onMapReady(GoogleMap googleMap) {
49          mMap = googleMap;
50          mMap.addMarker(new MarkerOptions().position(latLng).title(ADDRESS));
51          mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng, 17));
52          // 按下 Marker
53          mMap.setOnMarkerClickListener(new GoogleMap.OnMarkerClickListener() {
54              @Override
55              public boolean onMarkerClick(Marker marker) {
56                  LatLng latLng = marker.getPosition();
57                  // 顯示街景縮略圖
58                  setStreetViewThumbnails(latLng);
59                  return false;
60              }
61          });
62
63          // 按下 InfoWindow
64          mMap.setOnInfoWindowClickListener(new GoogleMap.OnInfoWindowClickListener() {
65              @Override
66              public void onInfoWindowClick(Marker marker) {
67                  LatLng latLng = marker.getPosition();
68                  // 取得街景
69                  String path = "google.streetview:cbll=%s,%s&mz=21";
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14

```
70         path = String.format(path, latLng.latitude, latLng.longitude);
71         Uri uri = Uri.parse(path);
72         Intent intent = new Intent(android.content.Intent.ACTION_VIEW,
73             uri);
74         startActivity(intent);
75     }
76 });
77 }
78 // 街景縮略圖
79 private void setStreetViewThumbnails(final LatLng latlng) {
80     // Google Street View Image API
81     String url = "http://maps.googleapis.com/maps/api/streetview?" +
82         "size=450x250&location=%s,%s";
83     url = String.format(url, latlng.latitude, latlng.longitude);
84     // 利用 Picasso 取得街景縮略圖
85     Target target = new Target() {
86         @Override
87         public void onBitmapLoaded(Bitmap bitmap, Picasso.LoadedFrom from) {
88             groundOverlay = mMap.addGroundOverlay (
89                 new GroundOverlayOptions()
90                     .image(BitmapDescriptorFactory.fromBitmap(bitmap))
91                     .anchor(0, 0)
92                     .position(latlng, 150f, 100f));
93             groundOverlay.setTransparency(0.3f); // 透明度
94         }
95         @Override
96         public void onBitmapFailed(Drawable errorDrawable) {
97         }
98         @Override
99         public void onPrepareLoad(Drawable placeHolderDrawable) {
100             }
101         };
102         Picasso.with(context).load(url).into(target);
103     }
104 }
```

➔ 重點說明

程式第 69、71 行設置街景 URI。

程式第 81~83 行利用 Google Street View Image API 設置街景縮略圖 URI。

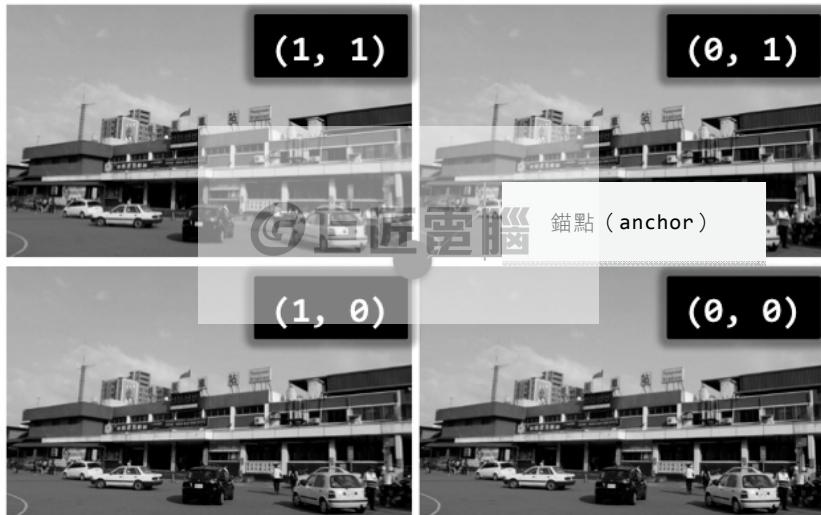
程式第 85~102 行利用 Picasso 取得街景圖。

程式第 88~93 行設置地圖疊層。

```
groundOverlay = mMap.addGroundOverlay (
    new GroundOverlayOptions()
        .image(BitmapDescriptorFactory.fromBitmap(bitmap))
        .anchor(0, 0)
        .position(latlng, 150f, 100f));
groundOverlay.setTransparency(0.3f); // 透明度
```

透過 `BitmapDescriptorFactory.fromBitmap(bitmap)` 方法製造一個 `BitmapDescriptor`，用來協助在地圖疊層中設置一個圖標（marker icon）。

程式第 92 行 `.anchor(0, 1.5f)` 設定縮略圖在錨點（anchor）的相對應位置。



縮略圖在錨點（anchor）的相對應位置參數設置

動動腦 試著修改 Map/app_streetviewmap 讓 Marker 移動到新地點時不要僅顯示“新地點”而是將新地點的地址或地標於 Info-Window 中顯示出來。



14-6 地圖與導航

Google Directions API 服務使用 HTTP 要求來計算位置之間的導航⁴。支援多種運輸模式的導航，包括大眾運輸(**transit**)、行車(**driving**)、步行(**walking**)或單車(**bicycling**)。「導航」將會以文字字串或經度/緯度座標來指定導航的起點、目的地與路點。

使用限制

導航請求：目前每天只能發出 2,500 次請求，不論任何導航模式皆共用此配額，注意大眾運輸導航搜尋則會計入 4 次請求。

⁴ Google 主要導航伺服器 ditu.google.com (美國) 與 ditu.google.cn (中國-北京)。

中繼路點

行車、步行或單車導航要求時，每個要求最多可包含 8 個中繼路點（大眾運輸導航則不可使用路點）。

Google Maps API for Business 客戶

每天最多可查詢 100,000 個導航要求，而每個要求中最多可包含 23 個路點。

Directions API 網址長度

在進行網址編碼之前，Directions API 網址長度不得超過 2,048 個字元。部分導航服務網址可能涉及多個地點，因此在建立網址時請注意這項限制。

資料來源：

Google Directions API <https://developers.google.com/maps/documentation/directions/?hl=zh-tw>

 
在實作導航 App 一般可以分為 Web 版與 JSON 版，Web 版本在於開發者僅需要針對 URL 參數定義，之後就交由 Web 導航瀏覽器處理開發者無法過問，此方法雖建構快速，但後續若要開發者自行處理導航與 Map 之間的關係能力卻薄弱。JSON 版則是 Google 將導航原始資料以 JSON 格式回傳，開發者根據回傳資料自行分析並繪製在 Google Map 當中，開發者可以完全掌控導航資料與 Map 之間的關係。

以下就針對 Web 版與 JSON 版來做範例說明。

14-6-1 Intent 啟用導航瀏覽器

最快的方式開發導航 App，就是直接 Intent 啟用內建的導航瀏覽器，並搭配相關參數。

內建 Map App 導航瀏覽器「URL 格式」：

- WGS-84 坐標系：

<https://www.google.com/maps/preview?f=d&dirflg=> 交通工具

&saddr=起點經緯度（緯度，經度）&daddr=終點經緯度（緯度，經度）
 &hl=zh_TW

- GCJ-02 坐標系（主要用於中國地區）：

[https://ditu.google.com/maps?f=d&dirflg=交通工具&saddr=起點經緯度\(緯度,經度\)&daddr=終點經緯度\(緯度,經度\)&hl=zh_TW&t=m⁵。](https://ditu.google.com/maps?f=d&dirflg=交通工具&saddr=起點經緯度(緯度,經度)&daddr=終點經緯度(緯度,經度)&hl=zh_TW&t=m⁵。)

內建 Map App 導航瀏覽器「相關參數」說明【如下表】：

表：Web 版導航 URL 相關參數說明

參數名稱	說明	範例
f	顯示路徑規劃表單	d
dirflg	交通工具： d=汽車、r=大眾捷運、w=步行	dirflg=d
saddr	起點經緯度（緯度，經度）	saddr=25.041746,121.550392
daddr	終點經緯度（緯度，經度）	daddr=25.047924,121.517081
hl	設定語系	hl=zh_TW
t	地圖樣式： m：一般街道地圖、e：Google Earth 樣式等...。	t=m 或 t=e

⁵ Google Web 導航 URL 一般常見的 Domain 有 www.google.com 與 ditu.google.com，其差別在於 www.google.com 是使用 WGS-84 坐標系，而 ditu.google.com 是使用 GCJ-02 坐標系（主要針對中國地區）。

範例程式 (Map/app_directionsweb)

利用呼叫內建 Map App 導航瀏覽器執行導航工作。

⬇ 執行結果 (實機測試)



⬇ 重點程式碼

MainActivity.java

```
...
11  public class MainActivity extends AppCompatActivity {
12      private EditText editText1, editText2;
13      private Context context;
14      @Override
15      protected void onCreate(Bundle savedInstanceState) {
16          super.onCreate(savedInstanceState);
17          setContentView(R.layout.activity_main);
18          context = this;
19          editText1 = (EditText) findViewById(R.id.editText1);
20          editText2 = (EditText) findViewById(R.id.editText2);
21      }
```

```
22     public void onClick(View view) {  
23         String urlString = "https://ditu.google.com/maps?" +  
24             "f=d&dirflg=%s&saddr=%s&daddr=%s&hl=zh_TW&t=m";  
25         String dirflg = "";  
26         String from = editText1.getText().toString();  
27         String to = editText2.getText().toString();  
28         switch(view.getId()) {  
29             case R.id.imageButton1:  
30                 dirflg = "d";  
31                 break;  
32             case R.id.imageButton2:  
33                 dirflg = "r";  
34                 break;  
35             case R.id.imageButton3:  
36                 dirflg = "w";  
37                 break;  
38         }  
39         urlString = urlString.format(urlString, dirflg, from, to);  
40         Uri uri = Uri.parse(urlString);  
41         Intent intent = new Intent(Intent.ACTION_VIEW, uri);  
42         startActivity(intent);  
43     }  
44 }
```



👉 重點說明

程式第 23、24 行設定導航 URL。

程式第 28~38 行決定所使用的交通工具 (d=汽車、r=大眾捷運、w=步行)。

程式第 39~42 行利用 `String` 中的 `format()` 方法來注入經緯度參數，並取得 `URI` 物件啟動導航瀏覽器。

14-6-2 JSON 版自製導航 App

JSON 版導航範例 URL：

Google Directions API

<https://developers.google.com/maps/documentation/directions/?hl=zh-tw#Waypoints>

URL 參數說明：

`https://maps.googleapis.com/maps/api/directions/json?`
`origin=`起點經緯度（緯度，經度）`&destination=`終點經緯度（緯度，經度）
`&sensor=`是否支援感應器（true/false）

JSON 版導航相關參數說明【如下表】：

表：JSON 版導航 URL 相關參數說明

參數名稱	說明	範例
origin	起點經緯度（緯度，經度）	saddr=25.041746,121.550392
destination	終點經緯度（緯度，經度）	daddr=25.047924,121.517081
sensor	是否支援感應器	sensor=true

以下針對 Google 回傳的導航 JSON 做範例演示：

URL 導航請求範例：

格式：JSON。



起點：`origin=24.991452089392176,121.31352759897709`。

終點：`destination=24.98814608545636,121.31451431661844`。

模式：`mode=driving`。

感應器：是否支援感應器。

完整的 URL 字串：

```
https://maps.googleapis.com/maps/api/directions/json?origin=24.991452089392176,121.31352759897709&destination=24.98814608545636,121.31451431661844&sensor=true&mode=driving。
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14



```
{  
  "routes": [  
    {  
      "bounds": {  
        "northeast": {  
          "lat": 24.9915907,  
          "lng": 121.3170986  
        },  
        "southwest": {  
          "lat": 24.9877677,  
          "lng": 121.3135807  
        }  
      },  
      "copyrights": "地圖資料©2014 Google",  
      "legs": [  
        {  
          "distance": {  
            "text": "0.8 公里",  
            "value": 841  
          },  
          "duration": {  
            "text": "3 分",  
            "value": 162  
          },  
          "end_address": "330台灣桃園縣桃園市大林路60-62號",  
          "end_location": {  
            "lat": 24.9877677,  
            "lng": 121.3146297  
          },  
          "start_address": "330台灣桃園縣桃園市民生路67巷",  
          "start_location": {  
            "lat": 24.9914632,  
            "lng": 121.3135807  
          },  
          "steps": [  
            {  
              "distance": {  
                "text": "18 公尺",  
                "value": 18  
              },  
              "duration": {  
                "text": "1 分",  
                "value": 3  
              }  
            }  
          ]  
        }  
      ]  
    }  
  ]  
}
```

以 JSON 格式回傳 Google 導航資料

JSON 導航格式概覽：

```
{  
  "routes": [  
    {  
      "bounds": {...}, (此路線的檢視區邊框)  
      "copyrights": "地圖資料©2014 Google, Kingway",  
      "legs": [  
        {  
          ... (包含指定路線中兩個位置之間路線的航段相關資訊)  
        }  
      ],  
      "overview_polyline": {...} (存放編碼的 points 陣列用來代表導航路徑)  
    }  
  ]  
}
```

```

        "summary" : "民生路、長安街和大林路",
        "warnings" : [...], (導航時應顯示的警告資訊)
        "waypoint_order" : [...] (指定計算的路線中任何路點的順序)
    }
],
"status": "OK" (導航回應狀態)
}

```

以此 JSON 格式來說，因為要繪出每一段航段的導航路線並將這些航段接起，所以必須分析 "legs" 陣列。"legs" : [] 存放的就是每一段航段的細部資料，範例如下：

01	"legs" : [
02	{
03	"distance" : { (導航距離)
04	"text" : "0.8 公里",
05	"value" : 841 (單位公尺)
06	},
07	"duration" : { (導航時間)
08	"text" : "3 分",
09	"value" : 162 (單位秒)
10	},
11	"end_address" : " 330 台灣桃園縣桃園市大林路 60-62 號",
12	"end_location" : { (目的地經緯度)
13	"lat" : 24.9877677,
14	"lng" : 121.3146297 },
15	},
16	"start_address" : "330 台灣桃園縣桃園市民生路 67 巷",
17	"start_location" : { (出發地經緯度)
18	"lat" : 24.9914632,
19	"lng" : 121.3135807 },
20	},
21	"steps" : [(航段資訊陣列)
22	{ (第一段航段資訊)
23	"distance" : { (第一段航段距離)
24	"text" : "18 公尺",
25	"value" : 18 },
26	},
27	"duration" : { (第一段航段時間)
28	"text" : "1 分",
29	"value" : 3 },
30	"end_location" : { (第一段航段結束經緯度)
31	}] }

航段結束位置

```

32         "lat" : 24.991305,
33         "lng" : 121.3136209
34     },
35     "html_instructions" : "往\u003cb\u003e南\u003c/b\u003e
36             \u003c/b\u003e民生路 55 巷\u003c/b\u003e前進",
37     "polyline" : { (第一段航段折線編碼資訊)
38         "points" : "scpwC{\`mcV^G"
39     },
40     "start_location" : { (第一段航段開始經緯度)
41         "lat" : 24.9914632,
42         "lng" : 121.3135807
43     },
44     "travel_mode" : "DRIVING" (旅行模式)
45 },
46 { (第二段航段資訊) },
47 { (第三段航段資訊) },
48 { (第 N 段航段資訊) }
49 ],
50 "via_waypoint" : []
51 }]

```

航段起始位置

航段編碼折線路徑



```

"polyline" : {
    "points" : "scpwC{\`mcV^G"
},

```

航段編碼折線路徑就是一連串經緯度的集合後的編碼。

範例（一）："scpwC{\`mcV^G" 經過解碼之後如下：

```

( 24.99146, 121.31358 )
( 24.9913, 121.31362 )

```

範例（二）：" mdpwCqfmcVdB_B`BeBXSb@c@FGJOFMJU" 經過解碼之後如下：

```

( 24.99159, 121.31449 )
( 24.99108, 121.31497 )
( 24.99059, 121.31548 )
( 24.99046, 121.31558 )
( 24.99028, 121.31576 )
( 24.99024, 121.3158 )
( 24.99018, 121.31588 )
( 24.99014, 121.31595 )

```

```
( 24.99008, 121.31606 )
```

折線路徑解碼程式如下請參考：

```
private List<LatLng> decodePoly(String encoded) {  
  
    List<LatLng> poly = new ArrayList<LatLng>();  
    int index = 0, len = encoded.length();  
    int lat = 0, lng = 0;  
  
    while (index < len) {  
        int b, shift = 0, result = 0;  
        do {  
            b = encoded.charAt(index++) - 63;  
            result |= (b & 0x1f) << shift;  
            shift += 5;  
        } while (b >= 0x20);  
        int dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));  
        lat += dlat;  
  
        shift = 0;  
        result = 0;  
        do {  
            b = encoded.charAt(index++) - 63;  
            result |= (b & 0x1f) << shift;  
            shift += 5;  
        } while (b >= 0x20);  
        int dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));  
        lng += dlng;  
  
        LatLng p = new LatLng(((double) lat / 1E5),  
                             (((double) lng / 1E5)));  
        poly.add(p);  
    }  
  
    return poly;  
}
```



最後於程式中取得這些經緯度位置並繪製在 Google map 上。

```
lineOptions.addAll(points); // points 存放著該線段所有的緯經度值。
```

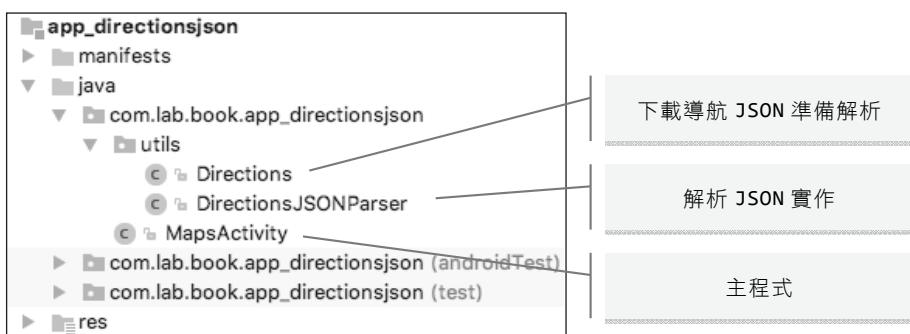
範例程式 (Map/app_directionsjson)

在 Google Map 中任意點選二點（起點+終點）自動繪出導航路徑圖。

↓ 範例結果



↓ 範例程式碼類別與功能說明



➔ 重點程式碼 1/3：解析 JSON 實作

utils.DirectionsJSONParser.java

```
..    ...
13   public class DirectionsJSONParser {
14
15   /** 接收一個 JSONObject 並返回一個列表的列表，包含經緯度 */
16   public List<List<HashMap<String, String>>> parse(JSONObject jobject) {
17
18       List<List<HashMap<String, String>>> routes =
19           new ArrayList<List<HashMap<String, String>>>();
20       JSONArray jRoutes = null;
21       JSONArray jLegs = null;
22       JSONArray jSteps = null;
23
24       try {
25
26           jRoutes = jobject.getJSONArray("routes");
27
28           /** Traversing all routes */
29           for (int i = 0; i < jRoutes.length(); i++) {
30               jLegs = ((JSONObject) jRoutes.get(i)).getJSONArray("legs");
31               List path = new ArrayList<HashMap<String, String>>();
32
33               /** Traversing all legs */
34               for (int j = 0; j < jLegs.length(); j++) {
35                   jSteps = ((JSONObject) jLegs.get(j)).getJSONArray("steps");
36
37                   /** Traversing all steps */
38                   for (int k = 0; k < jSteps.length(); k++) {
39                       String polyline = "";
40                       polyline = (String) ((JSONObject) ((JSONObject) jSteps
41                           .get(k)).get("polyline")).get("points");
42                       List<LatLng> list = decodePoly(polyline);
43
44                       /** Traversing all points */
45                       for (int l = 0; l < list.size(); l++) {
46                           HashMap<String, String> hm = new HashMap<String, String>();
47                           hm.put("lat",
48                               Double.toString(((LatLng) list.get(l)).latitude));
49                           hm.put("lng",
50                               Double.toString(((LatLng) list.get(l)).longitude));
51                           path.add(hm);
52                       }
53                   }
54               }
55           }
56       }
57   }
```

折線點線段解碼

1
2
3
4
5
6
7
8
9
10
11
12
13
14

```
53         }
54         routes.add(path);
55     }
56 }
57 } catch (JSONException e) {
58     e.printStackTrace();
59 } catch (Exception e) {
60 }
61 }
62
63     return routes;
64 }
65
66
67 // 解碼折線點的方法
68
69 private List<LatLng> decodePoly(String encoded) {
70
71     List<LatLng> poly = new ArrayList<LatLng>();
72     int index = 0, len = encoded.length();
73     int lat = 0, lng = 0;
74
75     while (index < len) {
76         int b, shift = 0, result = 0;
77         do {
78             b = encoded.charAt(index++) - 63;
79             result |= (b & 0x1f) << shift;
80             shift += 5;
81         } while (b >= 0x20);
82         int dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
83         lat += dlat;
84
85         shift = 0;
86         result = 0;
87         do {
88             b = encoded.charAt(index++) - 63;
89             result |= (b & 0x1f) << shift;
90             shift += 5;
91         } while (b >= 0x20);
92         int dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
93         lng += dlng;
94
95         LatLng p = new LatLng(((double) lat / 1E5)),
96                         (((double) lng / 1E5)));
```

```
97     poly.add(p);
98 }
99
100    return poly;
101 }
102 }
```

➡ 重點說明

程式第 30 行

```
jLegs = ((JSONObject) jRoutes.get(i)).getJSONArray("legs");
```

程式第 35 行

```
jSteps = ((JSONObject) jLegs.get(j)).getJSONArray("steps");
```

程式第 41 行 .get(k).get
("polyline")).get("points"
);



```
"legs" : [
  {
    "distance" : {
      "text" : "0.8 公里",
      "value" : 841
    },
    "duration" : {
      "text" : "3 分",
      "value" : 162
    },
    "end_address" : "330台灣桃園縣桃園市大林路60-62號",
    "end_location" : {
      "lat" : 24.9877677,
      "lng" : 121.3146297
    },
    "start_address" : "330台灣桃園縣市民生路67巷",
    "start_location" : {
      "lat" : 24.9914632,
      "lng" : 121.3135807
    },
    "steps" : [
      {
        "distance" : {
          "text" : "18 公尺",
          "value" : 18
        },
        "duration" : {
          "text" : "1 分",
          "value" : 3
        },
        "end_location" : {
          "lat" : 24.991305,
          "lng" : 121.3136209
        },
        "html_instructions" : "往\u0003cb\u0003e南\u0003c",
        "polyline" : {
          "points" : "scpwC(`mcV^G"
        },
        "start_location" : {
          "lat" : 24.9914632,
          "lng" : 121.3135807
        },
        "travel_mode" : "DRIVING"
      }
    ]
  }
]
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14

程式第 42 行 `List<LatLng> list = decodePoly(polyline);`; 折線點線段解碼。

程式第 69~101 行折線點線段解碼實施方法。

➔ 重點程式碼 2/3：負責下載導航 JSON 與準備解析

utils.Directions.java

```
..  
25     ...  
26     public class Directions {  
27         private GoogleMap map;  
28         private Context context;  
29         private String mode;  
30         public final static String MODE_DRIVING = "driving";  
31         public final static String MODE_WALKING = "walking";  
32         public final static String MODE_BICYCLING = "bicycling";  
33         public final static String MODE_TRANSIT = "transit";  
34  
35         private Directions() {}  
36         private static Directions _instance = new Directions();  
37         public static Directions getInstance() {  
38             return _instance;  
39         }  
40         public void draw(Context context, LatLng origin, LatLng dest,  
41             GoogleMap map, String mode) {  
42             this.context = context;  
43             this.map = map;  
44             this.mode = mode;  
45             // 取得 Google Directions API URL  
46             String url = getDirectionsUrl(origin, dest, mode);  
47             System.out.println(url);  
48             DownloadTask downloadTask = new DownloadTask();  
49             // 開始下載 json 經由 Google Directions  
50             downloadTask.execute(url);  
51         }  
52         private String getDirectionsUrl(LatLng origin, LatLng dest,  
53             String mode) {  
54  
55             // 起點位置  
56             String str_origin = "origin=" + origin.latitude + ","  
57                 + origin.longitude;  
58  
59             // 終點位置
```

```
60     String str_dest = "destination=" + dest.latitude + ","
61             + dest.longitude;
62
63     // Sensor enabled
64     String sensor = "sensor=true";
65
66     // 參數接合
67     String parameters = str_origin + "&" + str_dest + "&"
68             + sensor + "&mode=" + mode;
69
70     // 輸出格式
71     String output = "json";
72
73     // 建立完整 URL
74     String url = "https://maps.googleapis.com/maps/api/directions/"
75             + output + "?" + parameters;
76
77     return url;
78 }
79 // 下載 JSON 資料的實作方法
80 private String downloadUrl(String strUrl) throws IOException {
81     String data = "";
82     InputStream iStream = null;
83     HttpURLConnection urlConnection = null;
84     try {
85         URL url = new URL(strUrl);
86         // 建立 http connection
87         urlConnection = (HttpURLConnection) url.openConnection();
88         // 啟動連線
89         urlConnection.connect();
90         // 讀取資料
91         iStream = urlConnection.getInputStream();
92         BufferedReader br = new BufferedReader(new InputStreamReader(
93                 iStream));
94         StringBuffer sb = new StringBuffer();
95         String line = "";
96         while ((line = br.readLine()) != null) {
97             sb.append(line);
98         }
99         data = sb.toString();
100        br.close();
101    } catch (Exception e) {
102        Log.d("Exception while downloading url", e.toString());
103    } finally {
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14

```
104         iStream.close();
105         urlConnection.disconnect();
106     }
107     return data;
108 }
// 下載並解析 JSON
110 private class DownloadTask extends AsyncTask<String, Void, String> {
111     @Override
112     protected String doInBackground(String... url) {
113         String data = "";
114         try {
115             // 取得所下載的資料
116             data = downloadUrl(url[0]);
117         } catch (Exception e) {
118             Log.d("Background Task", e.toString());
119         }
120         return data;
121     }
122
123     @Override
124     protected void onPostExecute(String result) {
125         super.onPostExecute(result);
126         ParserTask parserTask = new ParserTask();
127         // 解析 JSON
128         parserTask.execute(result);
129     }
130 }
// 解析 JSON 格式
131 private class ParserTask extends
132     AsyncTask<String, Integer, List<List<HashMap<String, String>>> {
133     @Override
134     protected List<List<HashMap<String, String>>> doInBackground(
135         String... jsonData) {
136         JSONObject jObject;
137         List<List<HashMap<String, String>>> routes = null;
138         try {
139             jObject = new JSONObject(jsonData[0]);
140             DirectionsJSONParser parser = new DirectionsJSONParser();
141             // 解析 JSON 資料
142             routes = parser.parse(jObject);
143         } catch (Exception e) {
144             e.printStackTrace();
145         }
146         return routes;
147 }
```

```
148     }
149     @Override
150     protected void onPostExecute(List<List<HashMap<String, String>>> result) {
151         ArrayList<LatLng> points = null;
152         PolylineOptions lineOptions = null;
153         // 走訪所有 result
154         for (int i = 0; i < result.size(); i++) {
155             points = new ArrayList<LatLng>();
156             lineOptions = new PolylineOptions();
157             List<HashMap<String, String>> path = result.get(i);
158             // 得到每一個位置(經緯度)資料
159             for (int j = 0; j < path.size(); j++) {
160                 HashMap<String, String> point = path.get(j);
161                 double lat = Double.parseDouble(point.get("lat"));
162                 double lng = Double.parseDouble(point.get("lng"));
163                 LatLng position = new LatLng(lat, lng);
164                 // 放置折線點經緯度集合
165                 points.add(position);
166             }
167             // 繪製折線點經緯度集合
168             lineOptions.addAll(points);
169             lineOptions.width(5); // 導航路徑寬度
170             lineOptions.color(Color.BLUE); // 導航路徑顏色
171         }
172         if(lineOptions != null) {
173             map.addPolyline(lineOptions);
174         } else {
175             Toast.makeText(context, mode + "模式下無導航路徑 !",
176                         Toast.LENGTH_SHORT).show();
177         }
178     }
179 }
180 }
```

👉 重點說明

程式第 52~78 行取得導航 URL，參考 URL：

```
https://maps.googleapis.com/maps/api/directions/json?origin=24.991452089392176,121.31352759897709&destination=24.98814608545636,121.31451431661844&sensor=true&mode=driving
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14

程式第 80~108 行下載 JSON 資料。

程式第 110~130 行下載並準備解析 JSON 資料。

程式第 141~143 行解析 JSON。

➔ 重點程式碼 3/3：主程式

MapsActivity.java

```
..  
22 public class MapsActivity extends FragmentActivity  
23     implements OnMapReadyCallback {  
24     private GoogleMap mMap;  
25     private ArrayList<LatLng> markerPoints;  
26     private Context context;  
27     @Override  
28     protected void onCreate(Bundle savedInstanceState) {  
29         super.onCreate(savedInstanceState);  
30         setContentView(R.layout.activity_maps);  
31         SupportMapFragment mapFragment =  
32             (SupportMapFragment) getSupportFragmentManager()  
33             .findFragmentById(R.id.map);  
34         mapFragment.getMapAsync(this);  
35         context = this;  
36         // 儲存使用者點選的任意二點  
37         markerPoints = new ArrayList<LatLng>();  
38     }  
39     @Override  
40     public void onMapReady(GoogleMap googleMap) {  
41         mMap = googleMap;  
42         // 預設點 (桃園火車站)  
43         LatLng latLng = new LatLng(24.989206, 121.313549);  
44         mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng, 17));  
45         mMap.setOnMapLoadedCallback(new GoogleMap.OnMapLoadedCallback() {  
46             @Override  
47             public void onMapLoaded() {  
48                 setTitle(R.string.app_name);  
49                 Toast.makeText(context, "地圖載入完成!請任意點二點進行導航.",  
50                             Toast.LENGTH_SHORT).show();  
51             }  
52         });  
53         // 地圖 onClick 監聽器  
54         mMap.setOnMapClickListener(new GoogleMap.OnMapClickListener() {
```

```

55         @Override
56         public void onMapClick(LatLng point) {
57             // 確認地圖上的標記是否 >= 2
58             // 若二點(含)以上就清除地圖
59             if (markerPoints.size() >= 2) {
60                 markerPoints.clear();
61                 mMap.clear();
62             }
63             // 增加新標記
64             markerPoints.add(point);
65             // 建立標記選項
66             MarkerOptions options = new MarkerOptions();
67             // 設置標記選項位置
68             options.position(point);
69             // 起始及終點位置符號顏色
70             if (markerPoints.size() == 1) {
71                 // 起點符號顏色
72                 options.icon(BitmapDescriptorFactory
73                             .defaultMarker(BitmapDescriptorFactory.HUE_GREEN));
74             } else if (markerPoints.size() == 2) {
75                 // 終點符號顏色
76                 options.icon(BitmapDescriptorFactory
77                             .defaultMarker(BitmapDescriptorFactory.HUE_RED));
78             }
79             mMap.addMarker(options);
80             // 判斷地圖上的 marker 數量是否>=2
81             if (markerPoints.size() >= 2) {
82                 // 繪製導航路線
83                 Directions.getInstance().draw(context, markerPoints.get(0),
84                                              markerPoints.get(1), mMap, Directions.MODE_DRIVING);
85             }
86         }
87     });
88 }
89 }
```

➔ 重點說明

程式第 37 行：儲存使用者所點選的任意二點。

程式第 43 行：預設車站位置（桃園火車站）。

程式第 54~87 行：地圖 onClick 監聽器。

1
2
3
4
5
6
7
8
9
10
11
12
13
14

程式第 70 與 74 行：使用者 `onClick` 地圖之後判斷點擊的是第一點還是第二點，若地圖上只有一點，該點擊則為第二點，若地圖上已有二點，則會先清空地圖上所有標記再加入該點，則該點為第一點。

程式第 81~85 行：若地圖上的標記已經 ≥ 2 點則進行導航程序。



