

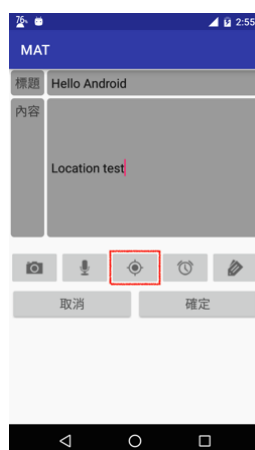
# Android Tutorial using Kotlin 第四堂

## （3）讀取裝置目前的位置 - Google Services Location

[Android Tutorial using Kotlin 第四堂（2）設計地圖應用程式 – Google Maps Android API << 前情](#)

目前的行動裝置大部份都有衛星定位的設備，在戶外適當的環境下，可以從衛星接收到精確度很高的位置資訊。在室內或遮蔽較多的環境，Android系統也可以從網路或電信服務，讀取誤差比較大一些的位置資訊。應用程式可以儲存使用這些位置資訊記錄與儲存目前的位置，在地圖元件中查詢與規劃路徑。

這一章說明最新的Google Services Location API，跟傳統的作法比較，這是一種比較省電與方便的技術，應用程式可以根據自需求，讀取需要的位置資訊。目前已經為記事應用程式完成地圖元件，現在為應用程式加入讀取與儲存目前位置資訊的功能，還沒有儲存位置資訊的記事資料，選擇位置功能，選擇允許位置資訊授權：



在地圖檢視目前的位置以後，點選目前位置的圖示，在對話框選擇「確定」就可以儲存位置資訊：



開啟已經儲存位置資訊的記事資料，可以在地圖上查詢位置，點選圖示以後再點選說明，可以在對話框選擇清除或記錄新的位置。





## 14-1 準備工作

依照下列的步驟，執行準備使用Google Services Location API的工作：

1. 啟動Android Studio並開啟MyAndroidTutorial應用程式。
2. 選擇Android Studio功能表「Tools -> Android -> SDK Manager」。
3. 在Android SDK Manager視窗，檢查「Extras -> Google Play services」是否已經安裝。如果還沒有安裝的話，勾選並執行安裝工作。
4. 開啟「Gradle Scripts -> build.gradle(Module:app)」，參考下面的內容，檢查是否已經加入需要的設定：

```
...
android {
    ...
}

dependencies {
    ...
    implementation 'com.google.android.gms:play-services-location:11.6.2'
    ...
}
```

5. 如果在上一步驟修改「build.gradle(Module: app)」檔案的內容，必須選擇功能表「Tools -> Android -> Sync Project with Files」執行同步的工作。
6. 開啟「ManifestAndroid.xml」，參考下面的內容，檢查在<application>標籤下是否已經加入需要的設定：

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

    ...

    <application ...>
```

```

...

    <meta-data android:name="com.google.android.gms.version"
        android:value="@integer/google_play_services_version" />
</application>

</manifest>

```

7. 同樣在「ManifestAndroid.xml」，參考下面的內容，檢查在<application>標籤下是否已經加入需要的設定：

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.macdidi.atk">

    ...

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

    <application ...>
        ...

    </application>

</manifest>

```

開啟「res/values/strings.xml」檔案，加入這一章需要的文字資源：

```

<string name="title_update_location">記事儲存的位置</string>
<string name="message_update_location">更新或清除儲存的位置資訊?</string>
<string name="update">更新</string>
<string name="clear">清除</string>

<string name="title_current_location">目前位置</string>
<string name="message_current_location">是否儲存目前位置?</string>

<string name="google_play_service_missing">裝置沒有安裝Google Play服務</string>

```

## 14-2 使用Google Services Location API

應用程式需要讀取位置資料，使用Google Services提供的Location API，是比較方便的作法。使用在

「com.google.android.gms.common.api」套件下的「GoogleApiClient」，可以連線與使用Google Services提供的服務。使用「com.google.android.gms.location」套件下的API，可以讀取裝置目前的位置資訊。

使用Google Services Location API讀取位置資訊，通常會採用整合在元件的作法，例如記事應用程式的地圖元件，讓它可以迴讀取位置資訊。開啟「net.macdidi.atk」套件下的「MapsActivity」，加入下列需要的欄位變數：

```
package net.macdidi.atk

...

class MapsActivity : AppCompatActivity(), OnMapReadyCallback {

    private lateinit var mMap: GoogleMap

    // Google API用戶端物件
    private lateinit var googleApiClient : GoogleApiClient
    // Location請求物件
    private lateinit var locationRequest : LocationRequest
    // 記錄目前最新的位置
    private lateinit var currentLocation : Location
    // 顯示目前位置的標記物件
    private var currentMarker : Marker? = null
    // 顯示儲存位置的標記物件
    private lateinit var itemMarker : Marker

    ...

}
```

### 14-2-1 使用Google API用戶端

地圖元件需要連線到Google API用戶端，使用位置資訊的服務。開啟「MapsActivity」，參考下列的程式片段，讓地圖元件類需要的介面，分別是在「com.google.android.gms.maps」套件下的ConnectionCallbacks與OnConnectionFailedListener：

```
package net.macdidi.atk

...

class MapsActivity : AppCompatActivity(), OnMapReadyCallback,
    GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener {

    ...

}
```

在元件加入介面需要實作的函式，後續會在函式中加入需要執行的工作：

```

package net.macdidi.atk

...

class MapsActivity : AppCompatActivity(), OnMapReadyCallback,
    GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener {

    ...

    // ConnectionCallbacks
    override fun onConnected(bundle: Bundle?) {
        // 已經連線到Google Services
    }

    // ConnectionCallbacks
    override fun onConnectionSuspended(i: Int) {
        // Google Services連線中斷
        // int參數是連線中斷的代號
    }

    // OnConnectionFailedListener
    override fun onConnectionFailed(connectionResult: ConnectionResult) {
        // Google Services連線失敗
        // ConnectionResult參數是連線失敗的資訊
    }

}

```

### 14-2-2 接收位置更新資訊

使用者需要為記事資料儲存位置的時候，需要在地圖顯示目前的位置讓使用者檢視與儲存，所以為地圖元件加入接收位置更新的功能。開啟「**MapsActivity**」，參考下列的程式片段，讓地圖元件類別實作需要的介面 `com.google.android.gms.location.LocationListener`：

```

package net.macdidi.atk

...

class MapsActivity : AppCompatActivity(), OnMapReadyCallback,
    GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener,
    LocationListener {

    ...

}

```

在元件加入介面需要實作的函式，後續會在函式中加入需要執行的工作：

```
package net.macdidi.atk

...

class MapsActivity : AppCompatActivity(), OnMapReadyCallback,
    GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener,
    LocationListener {

    ...

    // LocationListener
    override fun onLocationChanged(location: Location) {
        // 位置改變
        // Location參數是目前的位置
    }

}
```

### 14-3 Google API用戶端連線與接收位置更新資訊

需要使用Google Services Location服務，需要建立好需要的Google API用戶端物件，在「MapsActivity」加入下列建立Google API用戶端物件的函式：

```
package net.macdidi.atk

...

class MapsActivity : AppCompatActivity(), OnMapReadyCallback,
    GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener,
    LocationListener {

    ...

    // 建立Google API用戶端物件
    @Synchronized private fun configGoogleApiClient() {
        googleApiClient = GoogleApiClient.Builder(this)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .addApi(LocationServices.API)
            .build()
    }

}
```

```
}
```

在「MapsActivity」的「onCreate」函式加入呼叫上列函式的敘述：

```
package net.macdidi.atk

...

class MapsActivity : AppCompatActivity(), OnMapReadyCallback,
    GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener,
    LocationListener {

    ...

    override fun onCreate(savedInstanceState: Bundle?) {
        ...

        // 建立Google API用戶端物件
        configGoogleApiClient()
    }

    ...

}
```

應用程式啟動以後，就會建立好需要的Google API用戶端物件。在後續執行連線與運作的時候，應用程式會執行ConnectionCallbacks與OnConnectionFailedListener介面對應的函式。

應用程式需要接收最新的位置資訊，需要依照應用程式的需求，建立與啟動LocationRequest服務。在「MapsActivity」加入「建立LocationRequest物件」的函式：

```
package net.macdidi.atk

...

class MapsActivity : AppCompatActivity(), OnMapReadyCallback,
    GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener,
    LocationListener {

    ...

    // 建立Location請求物件
```



```

        private fun configLocationRequest() {
            locationRequest = LocationRequest()
            // 設定讀取位置資訊的間隔時間為一秒（1000ms）
            locationRequest.interval = 1000
            // 設定讀取位置資訊最快的間隔時間為一秒（1000ms）
            locationRequest.fastestInterval = 1000
            // 設定優先讀取高精確度的位置資訊（GPS）
            locationRequest.priority = LocationRequest.PRIORITY_HIGH_ACCURACY
        }

    }

```

在「MapsActivity」的「onCreate」函式加入呼叫上列函式的敘述：

```

package net.macdidi.atk

...

class MapsActivity : AppCompatActivity(), OnMapReadyCallback,
    GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener,
    LocationListener {

    ...

    override fun onCreate(savedInstanceState: Bundle?) {
        ...

        // 建立Location請求物件
        configLocationRequest()
    }

    ...

}

```

在「MapsActivity」的「onConnected」、「onConnectionFailed」與「onLocationChanged」函式，分別加入啟動位置更新錯誤處理的敘述：

```

package net.macdidi.atk

...

class MapsActivity : AppCompatActivity(), OnMapReadyCallback,
    GoogleApiClient.ConnectionCallbacks,

```

```

        GoogleApiClient.OnConnectionFailedListener,
        LocationListener {

...

// ConnectionCallbacks
override fun onConnected(bundle: Bundle?) {
    // 已經連線到Google Services
    // 取得授權狀態，參數是請求授權的名稱
    val hasPermission = ContextCompat.checkSelfPermission(
        this, Manifest.permission.ACCESS_FINE_LOCATION)

    // 如果已經授權
    if (hasPermission == PackageManager.PERMISSION_GRANTED) {
        // 啟動位置更新服務
        // 位置資訊更新的時候，應用程式會自動呼叫LocationListener.onLocationChanged
        LocationServices.FusedLocationApi.requestLocationUpdates(
            googleApiClient, locationRequest, this);
    }
}

...

// OnConnectionFailedListener
override fun onConnectionFailed(connectionResult: ConnectionResult) {
    // Google Services連線失敗
    // ConnectionResult參數是連線失敗的資訊
    val errorCode = connectionResult.errorCode

    // 裝置沒有安裝Google Play服務
    if (errorCode == ConnectionResult.SERVICE_MISSING) {
        Toast.makeText(this, R.string.google_play_service_missing,
            Toast.LENGTH_LONG).show()
    }
}

// LocationListener
override fun onLocationChanged(location: Location) {
    // 位置改變
    // Location參數是目前的位置
    currentLocation = location
    val latLng = LatLng(
        location.latitude, location.longitude)

    // 設定目前位置的標記
    if (currentMarker == null) {
        currentMarker = mMap.addMarker(MarkerOptions().position(latLng))
    } else {
        currentMarker?.setPosition(latLng)
    }

    // 移動地圖到目前的位置

```

```

        moveMap(latLng)
    }

    ...

}

```

Google API用戶端連線與接收位置更新資訊，是很耗用資源與電力的服務，所以會在元件的生命週期函式執行控制的工作。參閱下列的程式片段，修改「**MapsActivity**」的「**onResume**」函式，還有加入「**onPause**」與「**onStop**」兩個函式：

```

package net.macdidi.atk

...

class MapsActivity : AppCompatActivity(), OnMapReadyCallback,
    GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener,
    LocationListener {

    ...

    override fun onResume() {
        super.onResume()

        // 連線到Google API用戶端
        if (!googleApiClient.isConnected && currentMarker != null) {
            googleApiClient.connect()
        }
    }

    override fun onPause() {
        super.onPause()

        // 移除位置請求服務
        if (googleApiClient.isConnected) {
            LocationServices.FusedLocationApi.removeLocationUpdates(
                googleApiClient, this)
        }
    }

    override fun onStop() {
        super.onStop()

        // 移除Google API用戶端連線
        if (googleApiClient.isConnected) {
            googleApiClient.disconnect()
        }
    }

    ...
}

```

```
}
```

## 14-4 定位設備授權與位置資訊管理

為了處理Android 6的授權架構，開啟「ItemActivity」，加入下列的欄位變數與函式宣告：

```
package net.macdidi.atk

...

class ItemActivity : AppCompatActivity() {

    ...

    // 定位設備授權請求代碼
    private val REQUEST_FINE_LOCATION_PERMISSION = 102

    ...

    // 啟動地圖與定位元件
    private fun processLocation() {
        // 啟動地圖元件用的Intent物件
        val intentMap = Intent(this, MapsActivity::class.java)

        // 設定儲存的座標
        intentMap.putExtra("lat", item.latitude);
        intentMap.putExtra("lng", item.longitude);
        intentMap.putExtra("title", item.title);
        intentMap.putExtra("datetime", item.localeDatetime);

        // 啟動地圖元件
        startActivityForResult(intentMap, ItemAction.LOCATION.ordinal);
    }

}
```

同樣在「ItemActivity」類別，加入請求定位設備授權的函式：

```
package net.macdidi.atk

...

class ItemActivity : AppCompatActivity() {
```

```

...

// 讀取與處理定位設備授權請求
private fun requestLocationPermission() {
    // 如果裝置版本是6.0（包含）以上
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        // 取得授權狀態，參數是請求授權的名稱
        val hasPermission = ContextCompat.checkSelfPermission(
            this, Manifest.permission.ACCESS_FINE_LOCATION)

        // 如果未授權
        if (hasPermission != PackageManager.PERMISSION_GRANTED) {
            // 請求授權
            // 第一個參數是請求授權的名稱
            // 第二個參數是請求代碼
            requestPermissions(
                arrayOf(Manifest.permission.ACCESS_FINE_LOCATION),
                REQUEST_FINE_LOCATION_PERMISSION)
            return
        }
    }

    // 如果裝置版本是6.0以下，
    // 或是裝置版本是6.0（包含）以上，使用者已經授權，
    // 啟動地圖與定位元件
    processLocation()
}

}

```

同樣在「ItemActivity」類別，找到「onRequestPermissionsResult」函式，參考下列的說明加入需要的程式碼：

```

package net.macdidi.atk

...

class ItemActivity : AppCompatActivity() {

    ...

    override fun onRequestPermissionsResult(requestCode : Int,
                                           permissions : Array<String>,
                                           grantResults : IntArray) {
        if (requestCode == REQUEST_WRITE_EXTERNAL_STORAGE_PERMISSION) {
            ...
        }
        else if (requestCode == REQUEST_RECORD_AUDIO_PERMISSION) {
            ...
        }
    }
}

```

```

// 如果是定位設備授權請求
else if (requestCode == REQUEST_FINE_LOCATION_PERMISSION) {
    // 如果在授權請求選擇「允許」
    if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
        // 啟動地圖與定位元件
        processLocation();
    }
    // 如果在授權請求選擇「拒絕」
    else {
        // 顯示沒有授權的訊息
        Toast.makeText(this, R.string.write_external_storage_denied,
            Toast.LENGTH_SHORT).show();
    }
}
else {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults)
}
}

...

}

```

同樣在「ItemActivity」類別，找到「clickFunction」函式，參考下列的說明修改原來的程式碼：

```

package net.macdidi.atk

...

class ItemActivity : AppCompatActivity() {

    ...

    fun clickFunction(view: View) {
        when (view.id) {
            ...
            R.id.set_location -> {
                // 讀取與處理定位設備授權請求
                requestLocationPermission()
            }
            ...
        }
    }

    ...

}

```

同樣在「ItemActivity」，在「onActivityResult」函式加入接收位置資訊的程式碼：

```
package net.macdidi.atk

...

class ItemActivity : AppCompatActivity() {

    ...

    // 更改參數data的型態為Intent?
    override fun onActivityResult(requestCode: Int,
                                   resultCode: Int,
                                   data: Intent?) {

        if (resultCode == Activity.RESULT_OK) {
            val actionRequest = ItemAction.values()[requestCode]

            when (actionRequest) {
                ...
                ItemAction.LOCATION -> {
                    // 讀取與設定座標
                    val lat = data?.getDoubleExtra("lat", 0.0) ?: 0.0
                    val lng = data?.getDoubleExtra("lng", 0.0) ?: 0.0
                    item.latitude = lat;
                    item.longitude = lng;
                }
                ...
            }
        }
    }

    ...

}
```

完成上面的工作以後，使用者在已經儲存位置資訊的記事資料開啟地圖元件，就會在地圖畫面上顯示儲存的位置。使用者在地圖選擇儲存位置後，也可以儲存在記事資料庫中。

## 14-5 地圖元件的操作功能

最後的工作是在地圖元件提供使用者操作的功能，包含檢視與儲存目前的位置，還有更新或清除記事資料已經儲存的位置。回到「MapsActivity」，在「onCreate」函式加入需要的程式碼：

```
package net.macdidi.atk

...
```

```

class MapsActivity : AppCompatActivity(), OnMapReadyCallback,
    GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener,
    LocationListener {

    ...

    override fun onCreate(savedInstanceState: Bundle?) {
        ...

        // 連線到Google API用戶端
        if (!googleApiClient.isConnected()) {
            googleApiClient.connect();
        }
    }

    ...

}

```

同樣在「MapsActivity」類別，找到「onMapReady」函式，參考下列的說明修改原來的程式碼：

```

package net.macdidi.atk

...

class MapsActivity : AppCompatActivity(), OnMapReadyCallback,
    GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener,
    LocationListener {

    ...

    // 覆寫OnMapReadyCallback的函式
    override fun onMapReady(googleMap: GoogleMap) {
        mMap = googleMap

        // 讀取記事儲存的座標
        val lat = intent.getDoubleExtra("lat", 0.0)
        val lng = intent.getDoubleExtra("lng", 0.0)

        // 如果記事已經儲存座標
        if (lat != 0.0 && lng != 0.0) {
            // 建立座標物件
            val itemPlace = LatLng(lat, lng)
            // 加入地圖標記
            addMarker(itemPlace, intent.getStringExtra("title"),
                intent.getStringExtra("datetime"))
            // 移動地圖

```



```

        moveMap(itemPlace)
    }

    processController()
}

...

}

```

地圖元件需要提供使用者選擇標記與訊息框的操作功能，在「**MapsActivity**」加入下列的函式：

```

package net.macdidi.atk

...

class MapsActivity : AppCompatActivity(), OnMapReadyCallback,
    GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener,
    LocationListener {

    ...

    private fun processController() {
        // 對話框按鈕事件
        val listener = DialogInterface.OnClickListener { dialog, which ->
            when (which) {
                // 更新位置資訊
                DialogInterface.BUTTON_POSITIVE ->
                    // 連線到Google API用戶端
                    if (!googleApiClient.isConnected) {
                        googleApiClient.connect()
                    }
                // 清除位置資訊
                DialogInterface.BUTTON_NEUTRAL -> {
                    val result = Intent()
                    result.putExtra("lat", 0)
                    result.putExtra("lng", 0)
                    setResult(Activity.RESULT_OK, result)
                    finish()
                }
                // 取消
                DialogInterface.BUTTON_NEGATIVE -> {
                }
            }
        }

        // 標記訊息框點擊事件
        mMap.setOnInfoWindowClickListener { marker ->

```

```

// 如果是記事儲存的標記
if (marker.equals(itemMarker)) {
    val ab = AlertDialog.Builder(this@MapsActivity)

    ab.setTitle(R.string.title_update_location)
        .setMessage(R.string.message_update_location)
        .setCancelable(true)

    ab.setPositiveButton(R.string.update, listener)
    ab.setNeutralButton(R.string.clear, listener)
    ab.setNegativeButton(android.R.string.cancel, listener)

    ab.show()
}

// 標記點擊事件
mMap.setOnMarkerClickListener(GoogleMap.OnMarkerClickListener { marker ->
    // 如果是目前位置標記
    if (marker.equals(currentMarker)) {
        val ab = AlertDialog.Builder(this@MapsActivity)

        ab.setTitle(R.string.title_current_location)
            .setMessage(R.string.message_current_location)
            .setCancelable(true)

        ab.setPositiveButton(android.R.string.ok, DialogInterface.OnClickListener { _, _ ->
            val result = Intent()
            result.putExtra("lat", currentLocation.latitude)
            result.putExtra("lng", currentLocation.longitude)
            setResult(Activity.RESULT_OK, result)
            finish()
        })
        ab.setNegativeButton(android.R.string.cancel, null)

        ab.show()

        return@OnMarkerClickListener true
    }

    false
})
}

...
}

```

同樣在「MapsActivity」類別，參考下列的程式碼修改「addMarker」函式：

```

package net.macdidi.atk

...

class MapsActivity : AppCompatActivity(), OnMapReadyCallback,
    GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener,
    LocationListener {

    ...

    // 在地圖加入指定位置與標題的標記
    private fun addMarker(place: LatLng, title: String, context: String) {
        val icon : BitmapDescriptor =
            BitmapDescriptorFactory.fromResource(R.drawable.atk_launcher)

        val markerOptions = MarkerOptions()
        markerOptions.position(place)
            .title(title)
            .snippet(context)
            .icon(icon)

        // 加入並設定記事儲存的位置標記
        itemMarker = mMap.addMarker(markerOptions)
    }

    ...

}

```

完成所有工作了，在實體裝置執行應用程式，測試這一章完成的功能。

相關的檔案都可以在[GitHub](#)瀏覽與下載：

# GitHub

<https://github.com/macdidi5/Android-Tutorial-Kotlin>

[後續 >> Android Tutorial using Kotlin 第五堂（1）廣播接收元件 – BroadcastReceiver 與 AlarmManager](#)


---

Does Clearly work fine?

---



Shortcuts: **SHIFT+CTRL+C** to Toggle, **ESC** to Close.

 Give us feedback

Build upon  with Clearly

