Lab10
Service

本節目的：

- 了解什麼是 Service。
- 使用 Service 執行背景工作。

10.1 背景服務

Activity 在離開畫面後會進入停止狀態，在這狀態下我們是無法控制 Activity，除非透過調用 Thread 的方法，但是當 APP 完全結束關閉後，這些工作也會停止，如果希望能在背景中繼續執行工作，我們就會需要使用到 Service 來執行背景作業，像是在背景等待網路連線、背景作業等工作。

Service 最大的特色是他的執行任務與使用者的操作無關，Service 會獨立運行於背景，因此能夠在 APP 完全結束關閉後能保持啟動，甚至能在使用者操作別的 APP 時繼續執行，如等待網路訊息通知、下載資料與播放音樂...等作業，一旦任務完成 Service 就可以被結束，也可能就此常駐於裝置上。

10.1.1 創建 Service

Step1 要產生出一個新的 Service，首先選擇「File→New→Service→Service」來產生出空白的 Service，如圖 10-1 所示。

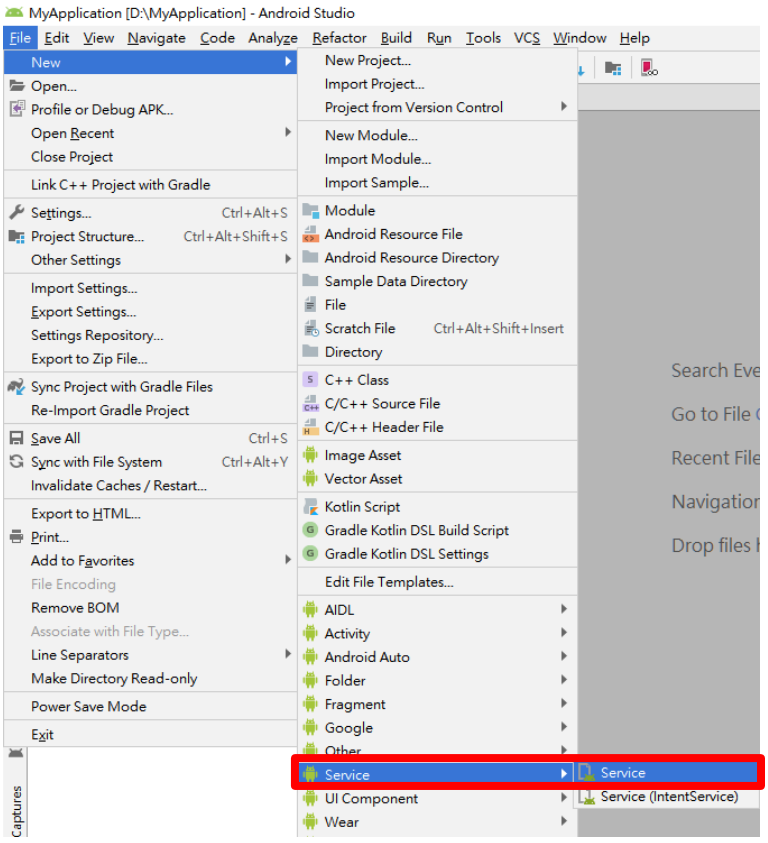

圖 10-1 產生新的 Service

Step2 在視窗中修改 Service 的名稱，完成後按下「Finish」，如圖 10-2 所示。

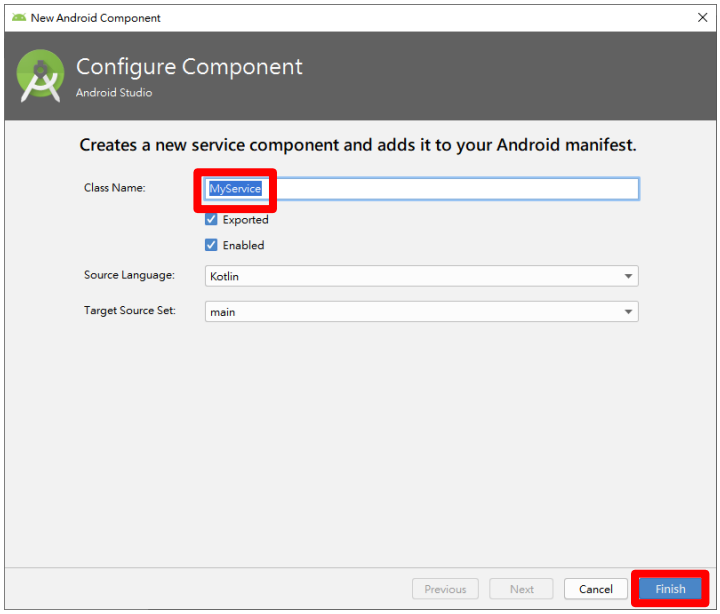

圖 10-2 輸入 Service 名稱並按下 Finish

完成後，系統會幫你產生出 Service 的類別檔，之後就可以在此編寫 Service。

```
class MyService : Service() {

    override fun onBind(intent: Intent): IBinder {
        TODO("Return the communication channel to the service.")
    }
}
```

AndroidManifest.xml 也會自動增加 Service 的資訊。

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

    <application
            android:allowBackup="true"
            android:icon="@mipmap/ic_launcher"
            android:label="@string/app_name"
            android:roundIcon="@mipmap/ic_launcher_round"
            android:supportsRtl="true"
            android:theme="@style/AppTheme">
```

```
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>

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

        <service
                android:name=".MyService"
                android:enabled="true"
                android:exported="true">
        </service>
    </application>
</manifest>
```

10.1.2 啟動 Service

Service 擁有獨立運行的能力，第一次產生的 Service 會執行 onCreate()方法，之後會自動調用 onStartCommand()方法，我們可以將要執行的工作寫於 onStartCommand()之中，而當需要結束時可以調用 stopSelf()來讓 Service 自行進入結束程序，觸發結束程序後皆會調用 onDestroy()來結束服務。

```
class MyService : Service() {
    //僅會在啟動時執行一次
    override fun onCreate() {
        super.onCreate()
    }

    override fun onBind(intent: Intent): IBinder {
        TODO("Return the communication channel to the
service.")
    }
    //每一次呼叫都會調用不論是否啟動過
    override fun onStartCommand(intent: Intent?, flags: Int,
startId: Int): Int {
        super.onStartCommand(intent, flags, startId)
        stopSelf() //結束 Service，呼叫 Service 類別的 onDestory()
        return super.onStartCommand(intent, flags, startId)
    }
}
```

Service 最重要的執行階段便是在調用 onStartCommand()，onStartCommand()會告訴系統如何重啟 Service，如異常終止後是否要重新啟動。而運行中的 Service 如果又接收到 Activity 發出 startService()的請求，Service 會執行 onStartCommand()而不會再次執行 onCreate()。換言之，onStartCommand()扮演著接收外來請求並操作 Service 的角色。

onStartCommand()的第一個參數 intent 可接收由 Activity 啟動時夾帶的資訊，第二參數 flags 表示啟動服務的方式，第三參數為啟動識別碼，而返回值主要有三種定義：

- **START_NOT_STICKY**：如果 Service 被結束時，便結束服務。
- **START_STICKY**：如果 Service 被結束時，系統會嘗試重啟並再次呼叫 onStartCommand()，不過 Intent 會被清空。
- **START_REDELIVER_INTENT**：如果 Service 被結束時，系統會嘗試重啟並再次呼叫 onStartCommand()，不過 Intent 會保留前次的並重新傳入。

```
override fun onStartCommand(intent: Intent?, flags: Int,
startId: Int): Int {
    super.onStartCommand(intent, flags, startId)
    //Service 被結束時會重啟並清空 Intent
    return START_STICKY
}
```

當要從 Activity 中啟用一個 Service，需要在 Activity 中呼叫 startService()方法啟動，透過 Intent 從目前的 activity（this）啟動 MyService::class.java 的 Service 元件。

```
startService(Intent(this, MyService::class.java))
```

如果要關閉 Service 可以從 Activity 呼叫 stopService()方法來做停止，透過意圖 Intent 從目前的 activity（this）停止 MyService::class.java 的 Service 元件。

```
stopService(Intent(this, MyService::class.java))
```

10.2 背景服務範例

- 本次範例要透過 Service 來啟動另一個 Activity。
- 在 MainActivity 中，按下「啟動 SERVICE」按鈕後啟動 Service，並結束 MainActivity。
- 在 Service 中，透過 Thread 延遲 5 秒後，啟動 Main2Activity。

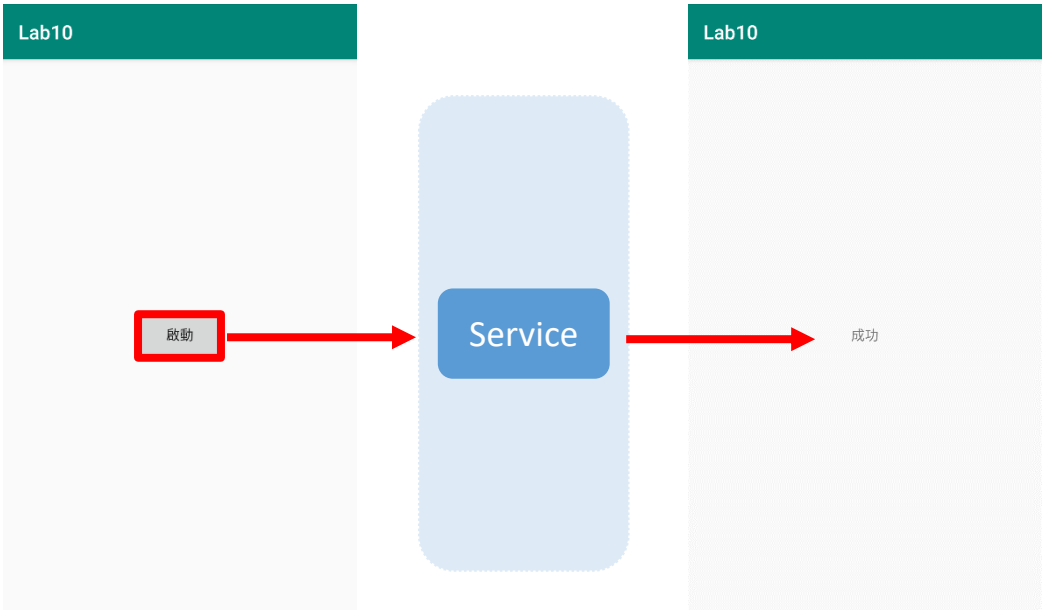

圖 10-3 MainActivity（左）與 Main2Activity（右）

10.2.1 設計步驟

Step1 新建專案，並建立兩個 Activity 與一個 Service，如圖 10-4 所示。

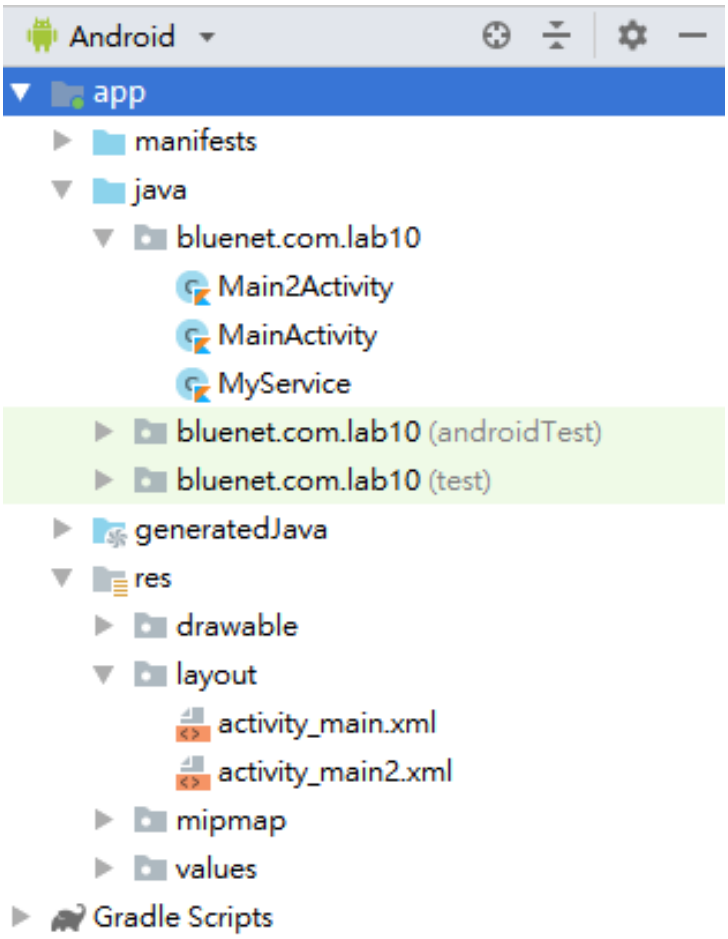

圖 10-4 APP 專案架構

Step2 繪製 activity_main.xml 檔，如圖 10-5 所示。

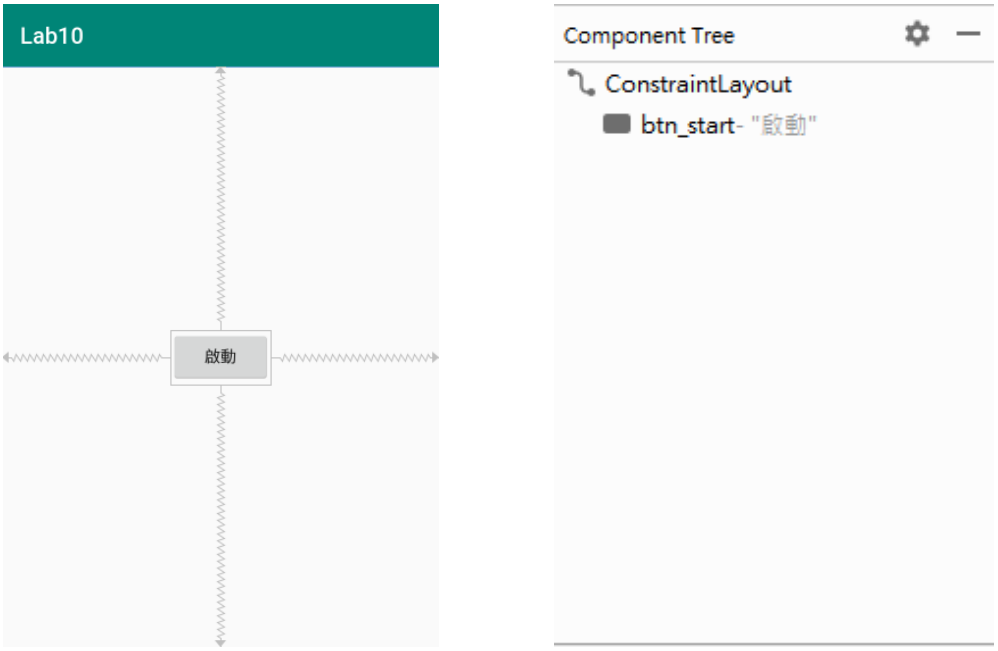

圖 10-5 MainActivity 預覽畫面（左）與佈局元件樹（右）

對應的 xml 如下：

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

    <Button
            android:id="@+id/btn_start"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="啟動"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
```

```
            app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Step3 繪製 activity_main2.xml 檔，如圖 10-6 所示。

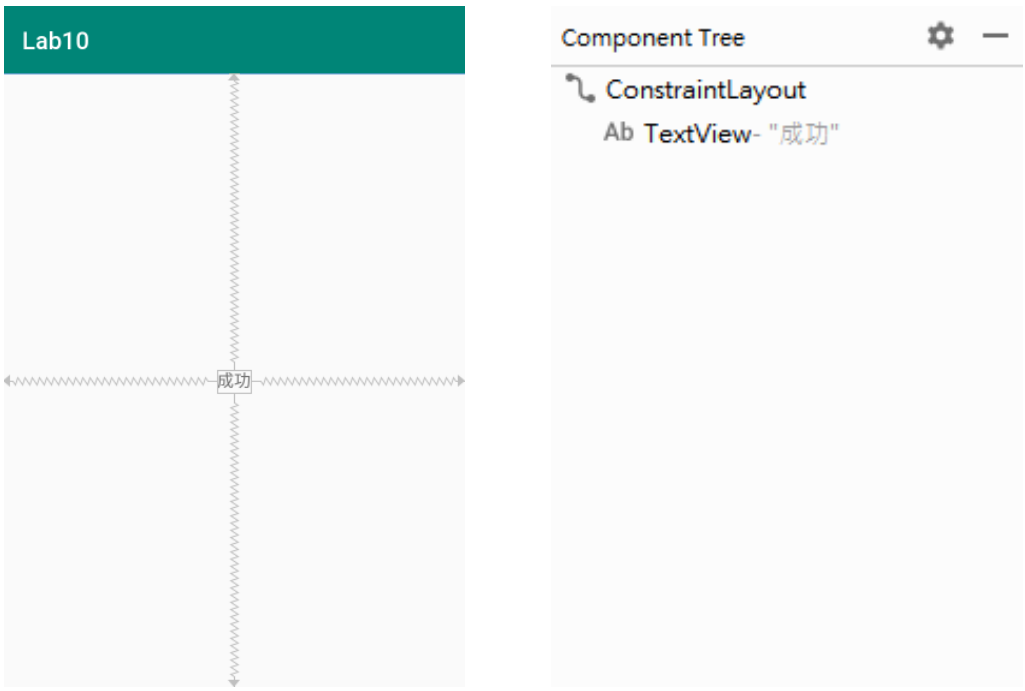

圖 10-6 Main2Activity 預覽畫面（左）與佈局元件樹（右）

對應的 xml 如下：

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".Main2Activity">

    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="成功"
            app:layout_constraintBottom_toBottomOf="parent"
```

```
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

10.2.2 程式設計

Step1 撰寫 MainActivity 按鈕監聽事件，按下按鈕後啟動後台 Service 並結束 MainActivity，將後續工作交給 Service。

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        //啟動按鈕監聽事件
        btn_start.setOnClickListener {
            //使用 startService，從目前 Activity(this)啟動 MyService 元件
            startService(Intent(this, MyService::class.java))
            Toast.makeText(this, "啟動 Service", Toast.LENGTH_SHORT)
            //關閉 Activity
            finish()
        }
    }
}
```

Step2 編寫 MyService 程式，在其中加入執行緒延遲 5 秒後，啟動 Main2Activity。

```
class MyService : Service() {
    override fun onCreate() {
        super.onCreate()
        //使用 Thread 來執行耗時工作
        Thread(Runnable {
            try {
                //使用 Sleep()延遲 5 秒
                Thread.sleep(5000)
                //宣告意圖，從 MyService 啟動 Main2Activity
                val intent = Intent(this@MyService,
                                                    Main2Activity::class.java)
                intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK
                this@MyService.startActivity(intent)
            } catch (e: InterruptedException) {
                e.printStackTrace()
            } //Service 要啟動 Activity 要加入 Flag 定義要去產生一個新的 Activity
        }).start()
```

```
    }

    override fun onStartCommand(intent: Intent, flags: Int, startid: Int):
Int {
        super.onStartCommand(intent, flags, startid)
        return Service.START_NOT_STICKY //Service 結束後不再重啟
    }

    override fun onBind(intent: Intent): IBinder? {
        return null
    }
}
```