

Android Tutorial using Kotlin 第一堂

（3）開始設計Android應用程式

[Android Tutorial using Kotlin 第一堂（2）建立Android Studio開發環境 << 前情](#)

一個Android應用程式，除了需要的原始程式碼，還必須包含一些需要的檔案，這些檔案都會放在規定的目錄，例如每一個Android應用程式模組，一定要有一個名為「AndroidManifest.xml」的應用程式設定檔，它是一個XML格式的檔案。所以開發Android應用程式，除了撰寫一些需要的程式碼，也要開始學習Android應用程式專案的結構和各種需要的檔案。

這一章會說明如何建立一個Android應用程式專案，瞭解基本的應用程式專案結構。如何在模擬裝置與實際裝置上執行與測試。還有認識Android應用程式基本元件與應用程式的種類。

3-1 應用程式專案介紹

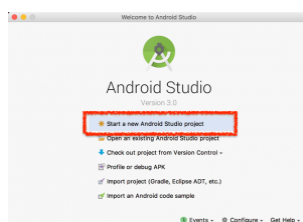
目前Android建議的應用程式開發工具為Android Studio，它整合所有開發Android 應用程式需要的工具，包含建立需要的模擬裝置，撰寫需要的程式與檔案，編譯、包裝、執行與測試應用程式，所有的工作都可以在Android Studio中完成。

3-1-1 建立應用程式專案

不管學習哪一種程式技術，第一個應用程式通常是在畫面上顯示一個簡單的問候訊息，例如「Hello world!」。接下來會說明建立一個Android應用程式專案，在模擬裝置執行以後，畫面上顯示簡單的問候訊息。雖然第一個Android應用程式專案非常簡單，但這裡說明的作法，都是開發Android應用程式基本的操作。

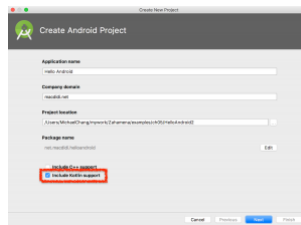
依照下列步驟建立一個Android應用程式，並且在Android模擬裝置中執行與測試：

1. 啟動Android Studio，選擇「Start a new Android Studio project」：

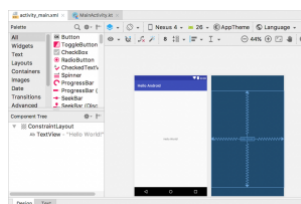


2. 依照下列的說明輸入應用程式基本資訊，後面會詳細的說明。輸入完後選擇「Next」：

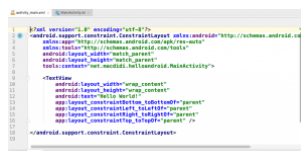
- Application Name：HelloAndroid。
- Company Domain：macdidi.net。
- Package Location：應用程式儲存的位置，使用原來的預設值。
- 勾選「Include Kotlin support」



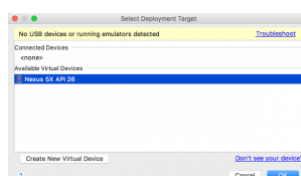
3. 在「Target Android Devices」畫面選擇應用程式模組的種類與版本，勾選「Phone and Tablet」，「Minimum SDK」選擇 16: Android 4.1 (Jelly Bean)」，最後選擇「Next」。
4. 在「Add an Activity to Mobile」畫面選擇「Empty Activity」，為應用程式加入一個基本的Activity元件，提供應用程式的畫選擇「Next」。
5. 在「Configure Activity」畫面，設定Activity元件的基本資訊，採用預設的名稱與設定，選擇「Finish」完成建立應用程式的定。
6. Android Studio啟動應用程式視窗，第一次啟動的時候會顯示一個操作功能提示的視窗，取消「Show Tips on Startup」選擇「Close」，下次就不會再出現。
7. 第一次啟動Android Studio應用程式視窗的時候，會執行一些初始化的工作，請耐心等待。
8. 選擇「activity_main.xml」標籤，開啟預設的畫面配置檔案，Android Studio顯示應用程式畫面設計的預覽：



9. 選擇「Text」標籤，可以開啟畫面配置檔案的原始內容：



10. 安裝好Android Studio以後，已經建立好一個預設的Android模擬裝置（Android Virtual Device、AVD）。選擇Android Studio能表「Run -> Run 'app」，準備在Android模擬裝置中執行與測試。在選擇裝置的對話框，勾選「Launch emulator」啟動Android模擬裝置，Android virtual device目前只有一個選項。最後選擇「OK」：



11. Android Studio啟動Android模擬裝置，並且在模擬裝置安裝與啟動應用程式，這些工作需要花費一些時間，請耐心等待。
12. 應用程式啟動以後可以看到這樣的畫面：



在建立應用程式過程中輸入的資訊，有一些設定必須先認識，否則建立專案以後再執行修改的工作，會比較麻煩一些，而且容易出現錯誤。在第一個視窗的應用程式專案基本資訊部份：

- **Application Name**：為這個應用程式取一個名稱，應用程式安裝在裝置以後，這個名稱會出現在應用程式列表的圖示下方。名稱應該要盡量簡短一些。
- **Company Domain**：輸入公司或個人的網域名稱，用來決定應用程式的主要套件名稱。Android規定應用程式一定需要套件，以把程式碼放在預設套件，而且最少要有兩層。這個套件名稱非常重要，因為Android應用程式是採用套件名稱來識別的，Google Play上的Android應用程式，不允許有同樣的套件名稱。Android Studio會把你輸入的網域名稱前後顛倒，當作前面名稱，後面再加上應用程式的名稱。例如網域名稱是macdidi.net，應用程式名稱是HelloAndroid，套件的名稱就是「net.macdidi.helloandroid」。
- **Package Location**：決定應用程式儲存的位置，預設是在使用者目錄下的「AndroidStudioProjects」，你也可以自己決定應用程式專案儲存的位置。

3-1-2 Android應用程式架構

一個Android應用程式專案除了必要的原始程式碼，還有各種應用程式需要的檔案，所以應用程式專案的目錄會比較多一些。Android Studio建立一個應用程式專案以後，會建立一些預設的目錄與檔案，你需要先認識一些基本的目錄。下列是在「app」Android應用程式模組目錄下的內容：

- **manifests**：應用程式模組的主要設定檔「AndroidManifest.xml」在這個目錄下。
- **java**：應用程式模組需要的Kotlin原始程式碼都放在這個目錄下，展開它以後會看到一個套件，這是在建立專案時決定的主名稱。目前這個套件下有一個預設元件類別「MainActivity.kt」。
- **res**：Android應用程式很重要的資源目錄，應用程式需要的資源，例如圖形（png）與音效（mp3）檔案，還有各種XML檔案都放在這個目錄。例如在「res/layout」目錄下，已經建立一個名為「activity_main.xml」的預設畫面配置檔案。

另外還有一個「Gradle Scripts」目錄，儲存與Gradle建置系統相關的設定檔，Gradle是Android Studio採用的全新應用程式建置系統。在Android Studio開發Android應用程式，一個應用程式可以有許多模組（Module）。例如一個音樂播放應用程式，可以分為行動電話、平板電腦、穿戴式三個模組，每一個模組都可以被建置成一個獨立的App。在Gradle Scripts目錄有下列兩個主要的設定檔，它們都是Groovy格式的文字檔：

- **build.gradle(Project:專案名稱)**：應用程式最頂端的Gradle設定檔。以前面建立的應用程式專案來說，名稱是「build.gradle(Project:HelloAndroid)」。
- **build.gradle(Module:模組名稱)**：每一個模組的Gradle設定檔。以前面建立的應用程式專案來說，名稱是「build.gradle(Module:app)」。

3-2 應用程式模組Gradle設定檔

每一個Android應用程式模組都有一個Gradle建置設定檔案，以前面建立的應用程式專案來說，這個設定檔是在「Gradle Scripts」目錄下的「build.gradle(Module:app)」。下列是這個檔案的基本格式：

```
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply plugin: 'kotlin-android-extensions'

android {
    compileSdkVersion 應用程式模組編譯的版本 (Android API Level)
    defaultConfig {
        applicationId "應用程式模組的主套件名稱"
        minSdkVersion 最低版本 (Android API Level)
        targetSdkVersion 主要版本 (Android API Level)
        versionCode 自定的應用程式模組版本編號
        versionName "自定的應用程式模組版本名稱"
        ...
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jre7:$kotlin_version"
    ...
}
```

在建立應用程式專案的時候，Android Studio會依照你的設定，建立一個像這樣的檔案：

```
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply plugin: 'kotlin-android-extensions'

android {
    compileSdkVersion 26
    defaultConfig {
        applicationId "net.macdidi.helloandroid"
        minSdkVersion 16
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
}
```

```

    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rule
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation"org.jetbrains.kotlin:kotlin-stdlib-jre7:$kotlin_version"
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
}

```

如果想要讓應用程式也可以在這一個版本運作，在2.3.x都支援應用程式使用的APIs的情況下，就可以把「defaultConfig」區塊「minSdkVersion」設定改為「10」。修改這個設定檔以後，Android Studio會提醒你執行專案同步的工作，選擇畫面右上角「Sync Now」，或是選擇功能表「Tools -> Android -> Sync Project with Gradle Files」。

在「defaultConfig」區塊中的「versionCode」與「versionName」，可以設定應用程式的版本資訊：

- versionCode="數字"：自己編製的版本流水號，例如1,2,3...。
- versionName="版本名稱"：自己命名的版本名稱，例如1.1或2.0。

在「android」區塊中的「buildToolsVersion」設定，可以用來決定建置工具的版本。因為Android建置工具會經常更新，最新版本可以在Android SDK Manager中查詢。

3-3 應用程式模組設定檔

每一個Android應用程式模組還有一個主要的設定檔案，檔案的名稱是「AndroidManifest.xml」，位置在應用程式模組的「manifests」目錄下。它是一個XML格式的檔案，檔案裡面都是一些應用程式非常重要的設定。設定檔的內容使用XML格式法，你會使用一些Android提供的標籤執行需要的設定。

Android應用程式中會包含許多XML格式的檔案，不同用途的檔案，都會使用一些「標籤」來執行一些設定，以應用程式設定說，它的基本內容會像這樣：

```

<?xml version="1.0" encoding="utf-8"?>
<!-- 上面這行表示這是一個XML文件，一定要在第一行 -->
<!-- 最外層一定是一個manifest的標籤 -->
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="應用程式主要套件" >

```

```
<!-- 在裡面使用不同的標籤執行一些設定 -->
...
<!-- manifest的結束標籤，表示它的範圍 -->
</manifest>
```

因為在設計Android應用程式的時候，會使用很多XML格式的檔案，所以你必須先瞭解XML基本的語法。XML是一個純文字檔案，不同技術的XML檔案，會有自己設定的各種標籤名稱。使用標籤有幾種固定的寫法，其中一種是包含開始和結束的標籤：標籤名稱的後面，通常會有一些設定這個標籤的資訊，設定值一定要放在雙引號之間：

```
<標籤名稱 設定名稱="設定值" ... >
...
</標籤名稱>
```

在執行不同設定的時候也有另外一種寫法，在標籤名稱後面同樣可以有一些設定資訊，不過它沒有結束的標籤，在結尾部份使用「/>」：

在XML文件中的標籤名稱和設定名稱都是分大小寫的，在一個標籤裡面執行一些設定的時候，它們的前後順序並沒有特別規定，如一個像這樣的標籤設定：

```
<Button android:color="#000"
        android:text="Hello!" />
```

跟這樣的標籤寫法效果是一樣的：

```
<Button android:text="Hello! "
        android:color="#000" />
```

3-3-1 設定應用程式基本資訊

使用Android Studio建立Android應用程式以後，會為你建立好應用程式設定檔案，裡面已經包含在建立專案過程輸入的一些基本資訊，例如應用程式模組的名稱。如果在建立專案的時候輸入錯誤的資訊，也可以開啟這個檔案後直接修改它：

```
<?xml version="1.0" encoding="utf-8"?>
<!-- 最外層一定是一個manifest的標籤 -->
<!-- xmlns:android="..."，這個設定一定要有，而且固定不變 -->
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    ... >
```

```
...
</manifest>
```

在manifest標籤中有這些應用程式的基本設定：

- **package="應用程式模組的主套件名稱"**：在建立應用程式的時候輸入的主套件名稱，儘可能不要修改它，如果真的要修改在原始程式碼src目錄下的套件也要一起修改；存檔後，ADT會自動幫你修改gen目錄下的套件名稱。
- **android:installLocation="安裝位置"**：設定應用程式安裝的位置，可以設定為「auto」、「internalOnly」或「preferExternal」，依序為由系統自動決定、只安裝在內建儲存位置和以外部儲存設備（記憶卡）為優先。

3-3-2 設定應用程式模組的設備資訊

在應用程式設定檔的「manifest」開始和結束標籤裡面，也可以使用這些標籤加入常用的設定：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest .. >
    <!-- 設定應用程式需要的操作設備 -->
    <uses-configuration ... />

    <!-- 設定應用程式需要的螢幕設備 -->
    <supports-screens ... />
</manifest>
```

如果應用程式需要一些特定的硬體設備才可以正確的運作，例如觸控螢幕或實體鍵盤，可以使用放在manifest標籤裡面的「uses-configuration」標籤執行相關的設定。它可以使用下列這些設定：

- **android:reqFiveWayNav="true|false"**：是否需要上下左右方向的控制設備。
- **android:reqHardKeyboard="true|false"**：是否需要實體鍵盤。
- **android:reqKeyboardType="設定"**：設定實體鍵盤種類，可以設定為undefined、nokeys、qwerty或twelvekey。
- **android:reqNavigation="設定"**：是否需要瀏覽資料用的控制設備，可以設定為undefined、nonav、dpad、trackball或wheel。
- **android:reqTouchScreen="設定"**：設定觸控螢幕設備種類，可以設定為undefined、notouch、stylus或finger，目前的實體部份都是finger這種觸控螢幕設備。

Android裝置有一個特點，就是各種裝置的螢幕尺寸有很大的差異，如果應用程式需要在特定的螢幕尺寸才可以正確的運作，用放在manifest標籤裡面的「supports-screens」標籤執行相關的設定。它可以使用這些設定：

- **android:smallScreens="true|false"**：是否支援比HVGA小的螢幕。
- **android:normalScreens="true|false"**：是否支援HVGA、WVGA和WQVGA螢幕。
- **android:largeScreens="true|false"**：是否支援比HVGA、WVGA和WQVGA大的螢幕。
- **android:xlargeScreens="true|false"**：是否支援像平板電腦的大型螢幕。
- **android:requiresSmallestWidthDp="設定"**：設定最低螢幕寬度，設定值為畫素，例如「480」。

- `android:compatibleWidthLimitDp="設定"`：設定相容的最大螢幕寬度，設定值為畫素，例如「800」。
- `android:largeWidthLimitDp="設定"`：設定最大螢幕寬度，設定值為畫素，例如「1024」。

3-3-3 設定應用程式元件資訊

Android應用程式設定檔裡面，一定要加入需要的應用程式設定，在`manifest`標籤裡面，一定有一個「`application`」標籤，在該標籤中執行應用程式元件的重要設定：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application ... >
        <!-- 應用程式包含的Activity元件 -->
        <activity
            android:name="包含套件名稱的元件類別名稱" >
            ...
        </activity>

        <!-- 應用程式包含的服務元件 -->
        <service
            android:name="包含套件名稱的元件類別名稱" >
            ...
        </service>

        <!-- 應用程式包含的廣播接收元件 -->
        <receiver
            android:name="包含套件名稱的元件類別名稱" >
            ...
        </receiver>

        <!-- 應用程式需要的額外資訊 -->
        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="..." />
    </application>
</manifest>
```

Android提供許多不同應用程式的元件，讓你可以建立需要的應用程式，Android元件必須依照規則實作一個類別，寫好元件類別後，一定要在應用程式設定檔的`application`標籤中加入對應的設定：

- **Activity**：設定Activity元件的相關資訊。
- **Service**：設定服務元件的相關資訊。
- **Receiver**：設定廣播接收與小工具元件的相關資訊。

這些標籤中有一個一定要加入的設定是「`android:name`」，使用它設定元件類別的名稱，這樣Android才可以正確使用這些元件。使用Android Studio新增Activity元件以後，會自動加入相關的設定。

3-4 Android應用程式介紹

就像個人電腦的應用程式一樣，Android應用程式也有幾種不同的類型，Android作業系統可以讓你依照應用程式的需求，開發不同種類的應用程式元件，這些元件可以組合成一個具有操作功能畫面和其它各種用途的應用程式。以應用程式的功能來說Android應用程式可以分為這些種類：

- 一般應用程式：和個人電腦中的計算機或文書處理類似，Android可以建立各種提供操作功能與瀏覽資料畫面的「Activity」元件，依照應用程式的需求，可能會有多個「Activity」元件。
- 背景服務應用程式：在個人電腦中，通常會有一些在背景中運作的應用程式，例如預防病毒的軟體，你不需要去操作它，在作業系統的背景中幫我們檢查網路或檔案是否有病毒的問題。Android可以建立在背景中運作的「Service」和「BroadcastReceiver」元件，這種元件會在Android作業系統的背景中運作，在遇到指定的情況時，它就會執行你指定的工作。例如需要在使用者開啟網際網路時執行資料更新的工作，就需要用到這類元件。
- 小工具：Android提供一種很特別的桌面元件，它可以讓使用者安裝在桌面指定的位置，執行功能操作或瀏覽資料，這種元件「AppWidget」。

這些只是依照元件的分類來介紹Android應用程式，功能比較單純一點的應用程式，例如計算機，它只需要設計Activity元件畫面就可以了；如果應用程式的功能比較複雜，就可能同時需要Activity、Service、BroadcastReceiver和AppWidget四種元件，才能完成應用程式需要的功能。

3-4-1 一般應用程式

如果一個Android應用程式，只有包含一些畫面讓使用者執行操作或瀏覽資料，就可以把它稱為一般應用程式。在一般Android程式中，一個畫面就是一個繼承自「android.app.Activity」的類別，通常會把它稱為Activity元件，如果應用程式的功能比較複雜些，需要的畫面，也就是Activity元件就會多一些。

在建立Android應用程式專案的時候，Android Studio會幫你建立好一個預設的Activity元件，如果你沒有特別修改名稱的話，別名稱會是「MainActivity」，而且它會放在這個專案設定的主套件下。一般的應用程式設計通常會把應用程式需要的Activity類別放在主套件下。開啟這個Activity元件類別後會像這樣：

```
package net.macdidi.helloandroid

// 匯入需要的Android API
import android.os.Bundle
import android.support.v7.app.AppCompatActivity

// 繼承自AppCompatActivity類別
class MainActivity : AppCompatActivity() {

    // 覆寫AppCompatActivity類別的onCreate函式
    override fun onCreate(savedInstanceState: Bundle?) {
        // 一定要加入呼叫父類別onCreate函式的敘述
        super.onCreate(savedInstanceState)
        // 指定這個元件使用的畫面配置資源
        setContentView(R.layout.activity_main)
    }
}
```

Android採用目前應用程式常見的框架（Framework）開發方式，這表示你在設計一些Android元件類別的時候，都會讓你繼承指定的類別，在撰寫元件類別的時候，也要依照這個元件的規則，覆寫一些需要的函式。以Activity元件類別來說，目前只需要畫面設計與指定的工作，Activity元件類別規定你覆寫「onCreate」函式以後，在函式中執行Activity元件的準備工作。Android啟動這個Activity元件的時候，會呼叫「onCreate」函式一次，執行你寫在這個函式中的程式碼。執行覆寫工作的時候，大部份情況下，都必須呼叫父類別被覆寫的函式，然後再執行其它需要的工作。

Android App跟一般應用程式的設計方式有很大的差異，尤其是在資源的部份。雖然Activity元件是提供一個應用程式中的畫面，但在上面的元件類別中你看不到跟畫面相關的程式碼，只有一行指定畫面配置資源的敘述，它呼叫父類別的「setContentView」方法，在參數中指定一個以「R.layout」開始的資源變數，這是Android用來代表畫面配置資源的格式，後面的名稱是畫面資源名稱。你可以使用這個名稱到「專案目錄\res\layout」目錄下找到這個附加檔名為「.xml」的檔案。開啟它後會是這樣的內容：

```
<?xml version="1.0" encoding="utf-8"?>
<!-- 決定畫面的配置方式 -->
<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="net.macdidi.helloandroid.MainActivity">

    <!-- 目前畫面只有一個文字元件 -->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

Android希望設計應用程式的畫面時，不是在Activity元件類別的程式碼中，使用撰寫程式碼的方式來設計需要的畫面，而是應用XML格式的畫面配置檔設計畫面，在這個畫面配置檔案中加入畫面需要的元件，在Activity元件類別把這個檔案引用進來當作畫面就可以了。

如果你執行過這個應用程式，出現的畫面應該是簡單的在畫面中顯示「Hello world!」的訊息，可是你在畫面配置檔案中也看到一個訊息內容。這又是Android一個比較不一樣的設計方式，在應用程式中常需要顯示各種文字，例如顯示在功能按鈕上的文字，Android也不希望你直接設定在畫面配置檔中。所以在上面的文字元件（TextView）中，有一個「android:text」的設定，裡面的是「@string/」開始的內容，這是Android另外一種文字資源，後面是它的資源名稱「hello_world」。你可以開啟「專案目錄\res\values\strings.xml」檔案，它的內容會像這樣：

```
<resources>
    <!-- 在畫面配置檔中使用的文字資源 -->
```

```
<string name="app_name">My Application</string>
</resources>
```

3-4-2 背景服務應用程式

一般應用程式大多是在使用者啟動以後，在畫面執行一些功能的操作與瀏覽資料，使用者結束這個應用程式以後，它就不會再運作了。以目前常見的社群應用程式來說，除了提供畫面讓使用者操作與瀏覽資料，它也需要提供服務（Service）元件在Android作業系統的背景運作，在接收到伺服器傳送的訊息後通知使用者，例如通知使用者發表的文章有新的回應。

服務元件是一個繼承自「android.app.Service」的Java類別，應用程式可以啟動或停止服務元件，啟動的服務元件會一直在系統運作，通常用來執行接收遠端伺服器資訊這類工作，在需要的時候通知使用者或執行後續的工作。

應用程式也經常需要處理一些「系統事件」的工作，例如在電池電量過低或收到簡訊這類事件發生的時候，應用程式需要執行特定的工作。Android作業系統在特定的情況發生時發送「廣播事件」，應用程式需要處理這類需求的時候，可以設計一個廣播元件，它是繼承自「android.content.BroadcastReceiver」的類別。廣播接收元件與服務元件同樣會在系統的背景運作，如果發生廣播接收元件指定的事件，Android就會啟動與執行指定的工作。

服務與廣播接收元件除了撰寫需要的Java類別，也需要執行一些相關的設定，它們的應用與設計方式，跟活動、小元件元件者一樣，後續的內容會詳細的說明。

3-4-3 小工具

Android作業系統在一般的裝置中，提供五個桌面讓使用者可以自己放一些應用程式的圖示，點選它們就可以開啟應用程式，其它作業系統都會有的功能。Android也提供一種很特別的小工具（AppWidget）元件，使用者把它安裝到桌面上指定的位置的小工具還可以讓使用者調整大小。使用小工具來執行一些像是開啟與關閉無線網路或藍芽的設定，或是使用其它各種應用的工具，使用者不用進入應用程式，就可以執行一些功能操作或瀏覽資料。Android內建許多好用的小工具，你也可以開發各種不同的小工具。

小工具元件是一個繼承自「android.appwidget.AppWidgetProvider」的類別，它的設計方式會跟一般應用程式中的Activity元件一樣，不過你也需要為它建立一個元件類別，依照小工具元件的規則撰寫程式，設計畫面配置檔和建立所有需要的資源。

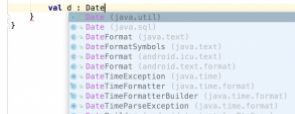
從Android 3.0開始，使用者安裝小工具元件的操作方式，改為從應用程式的列表進入後，選擇上方的小工具標籤，找到需要的小工具，長按小工具元件後再放到桌面上。從Android 3.1開始，如果小工具允許修改大小的話，使用者長按安裝在桌面上的小工具，等到調整大小的框線出現以後，就可以把小工具調整為需要的大小了。

3-5 在Android Studio撰寫Kotlin程式碼

Android應用程式元件需要搭配原始程式與其它各種檔案，撰寫程式碼是開發Android應用程式很主要的工作。安裝與設定好Android Studio，在輸入程式碼的時候，有許多方便的功能。在Android Studio應用程式視窗，開啟「app -> java -> net.macedidi.helloandroid」目錄下的「MainActivity」，它是一個Activity元件的Kotlin原始程式碼。在這個程式碼的「onCreate」方法最後面，輸入「val d : Date = Date()」敘述，輸入Date以後，Android Studio會自動列出可以選擇的項目，你可以注意到在java.util和java.sql套件都有Date類別，選擇java.util套件的Date類別，就會自動為你加入import敘述：

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val d : Date
    }
}
```




在撰寫程式碼的時候，也經常從別的地方複製後貼到Android Studio，例如下列這個程式片段：

```
...
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        ...
        // 複製與貼上下面兩行程式碼
        val message: TextView = TextView(this)
        message.text = "Hello"
    }
}
```

把上面的程式碼複製後貼到Android Studio，也會自動為你加入import敘述。如果貼上的程式碼，出現可以選擇的情況，例如個程式片段：

```
val listener : OnClickListener
```

Android Studio會顯示有多個Date可以選擇的訊息：



The screenshot shows a code editor with the line `val listener : OnClickListener`. A tooltip is visible above the code, displaying `android.view.View.OnClickListener? (multiple choices...) Ctrl`.

在上面的畫面按「Alt + Enter」，選擇正確的套件名稱後，就可以加入正確的import敘述：




The screenshot shows the same code editor with the line `val listener : OnClickListener`. A dropdown menu is open, showing the import path `android.view.View.OnClickListener` as the selected option.

後續 >> Android Tutorial using Kotlin 第一堂（4）開發Android應用程式的準備工作

Does Clearly work fine?



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

 Give us feedback

Build upon ♥ with Clearly

