Lab1
Android 環境建置與專案架構

本節目的：

- 建置環境並實做出第一個 APP。
- 了解 Android Studio 的開發環境與查看專案。

1.1 Android 環境建置

- 硬體方面：
  - 一台電腦。
  - 一支 Android 手機。

說明

要注意手機是否有驅動程式，可以到手機廠商的官方網站下載其所提供的驅動程式。若沒有手機也可以使用模擬器，以下教學會教導如何使用模擬器。

- 軟體方面：
  - JAVA 開發工具（Java Development kit - JDK）。
  - Android Studio 開發工具。
  - Android 開發工具（stand-alone Android SDK）。

1.1.1 JDK 配置

Step1 開啟 Java JDK 網址：http://www.oracle.com/technetwork/java/javase/downloads/index.html，下載 Java JDK 檔案，如圖 1-1 所示。

圖 1-1 Java JDK 下載

Step2 依照所使用的作業系統來選擇安裝 32 位元或是 64 位元，如圖 1-2 所示。

Java SE Development Kit 13

You must accept the Oracle Technology Network License Agreement for Oracle Java SE to download this software.
Thank you for accepting the Oracle Technology Network License Agreement for Oracle Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux	155.95 MB	jdk-13_linux-x64_bin.deb
Linux	163.02 MB	jdk-13_linux-x64_bin.rpm
Linux	179.97 MB	jdk-13_linux-x64_bin.tar.gz
mac OS	173.33 MB	jdk-13_osx-x64_bin.dmg
mac OS	173.68 MB	jdk-13_osx-x64_bin.tar.gz
Windows	159.82 MB	jdk-13_windows-x64_bin.exe
Windows	178.97 MB	jdk-13_windows-x64_bin.zip

圖 1-2 依照作業系統本身選擇適當的 JDK

Step3 開啟 JAVA JDK 安裝檔，然後點選「Next」，如圖 1-3 所示。

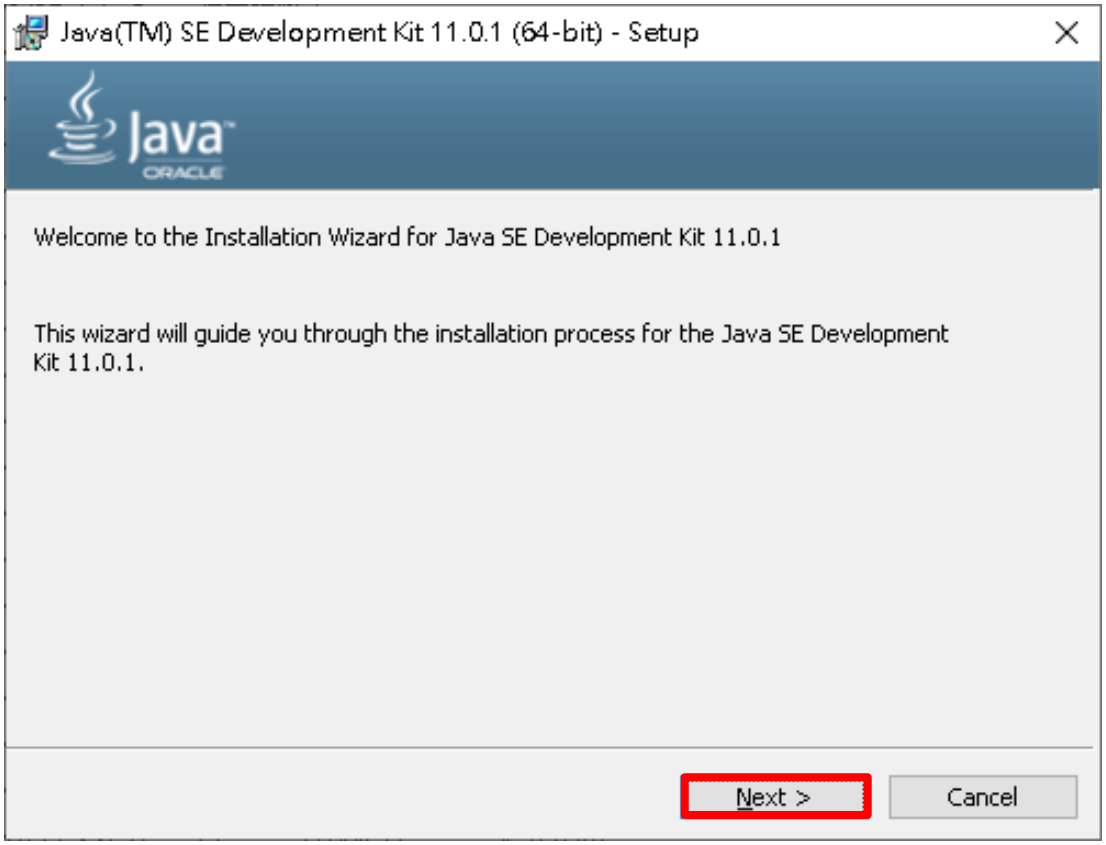

圖 1-3 開啟 JDK 安裝檔

Step4 選擇安裝的套件，這裡直接按下「Next」，如圖 1-4 所示。

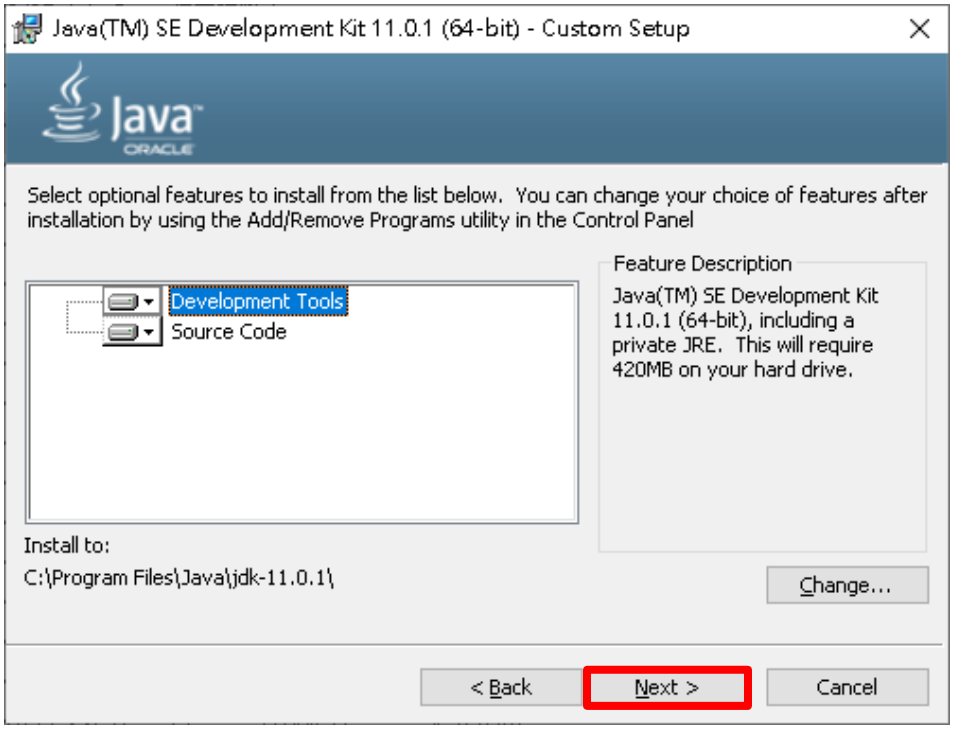

圖 1-4 選擇要安裝的 JDK 套件

Step5 圖 1-5 安裝完成，點擊「Finish」結束安裝程式，如圖 1-5 所示。

圖 1-5 JDK 安裝成功

1.1.2 Android Studio 開發工具

Step1 開啟 Android Studio 網址：https://developer.android.com/studio/，下載 Android Studio 開發環境，如圖 1-6 所示。

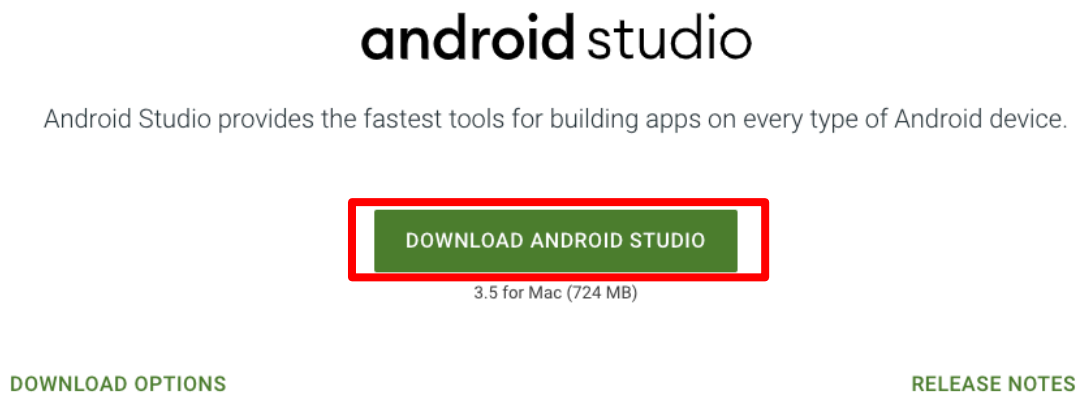

圖 1-6 下載 Android Studio

Step2 閱讀並同意 Android Studio 下載聲明，如圖 1-7 所示。

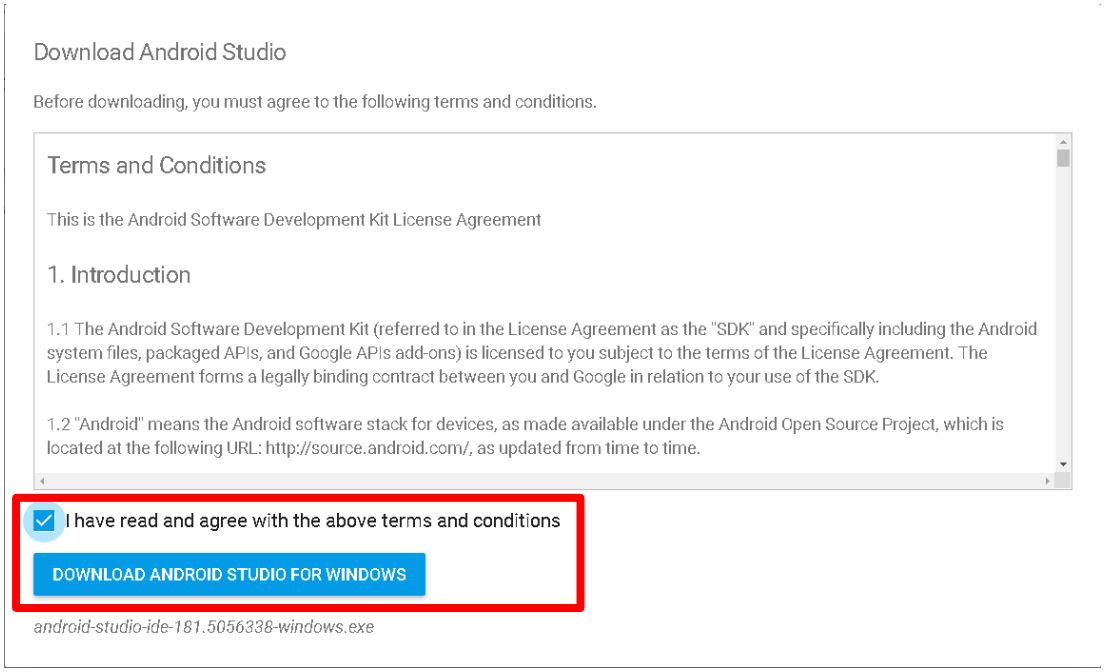

圖 1-7 閱讀並同意下載聲明

Step3 開啟 Android Studio 安裝檔，然後點擊「NEXT」，如圖 1-8 所示。

圖 1-8 開始安裝 Android Studio

Step4 選擇安裝套件，此處勾選「Android SDK」與「Android Virtual Device」，如圖 1-9 所示。

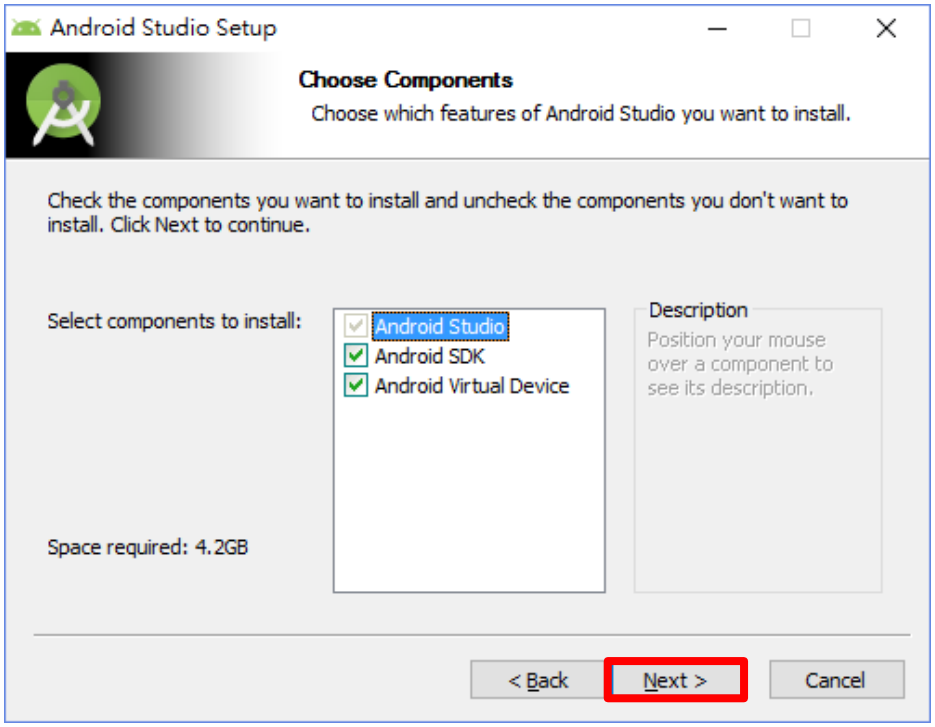

圖 1-9 選擇安裝 SDK 與 AVD 套件

Step5 閱讀並同意 Android Studio 安裝聲明，如圖 1-10 所示。

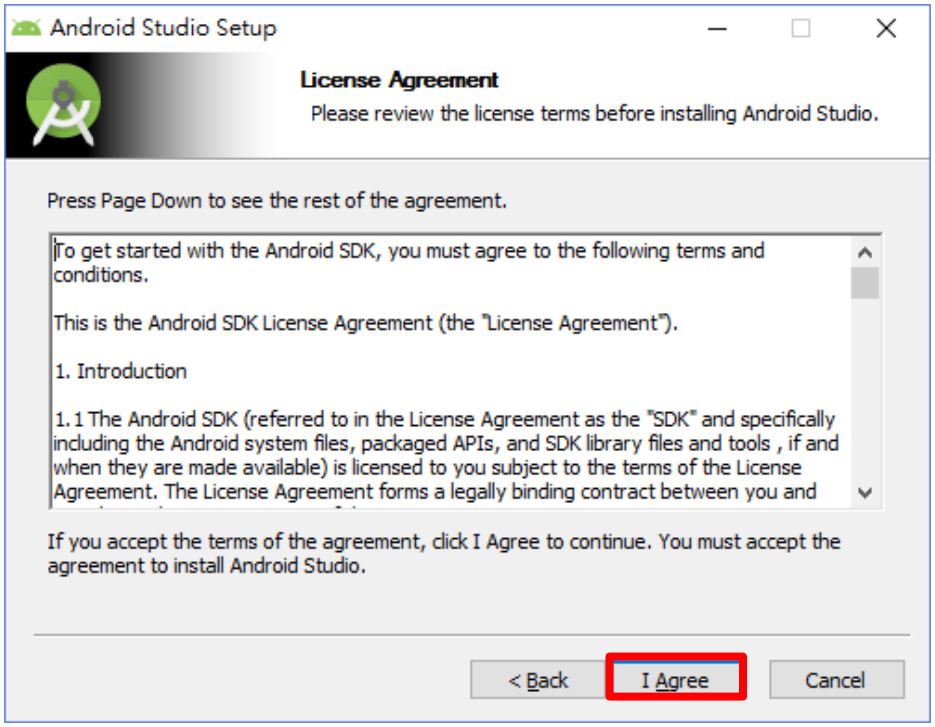

圖 1-10 閱讀並同意安裝聲明

Step6 設定 Android Studio 安裝路徑和 Android SDK 安裝路徑，然後點選「Next」，如圖 1-11 所示。

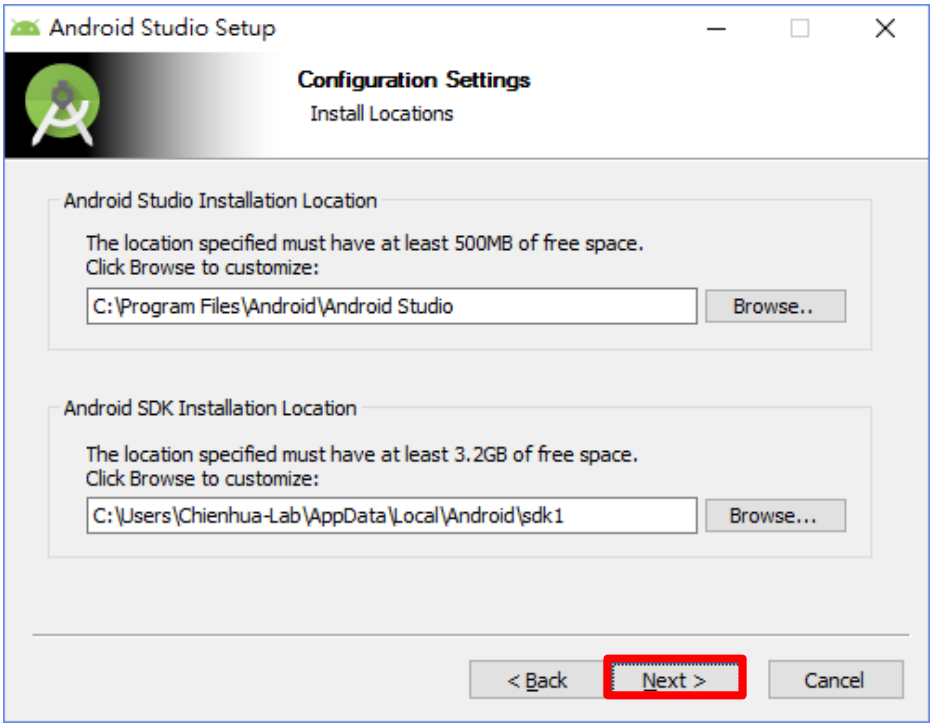

圖 1-11 設定安裝路徑

Step7 設定 Android Studio 於開始目錄的名稱，如圖 1-12 所示。

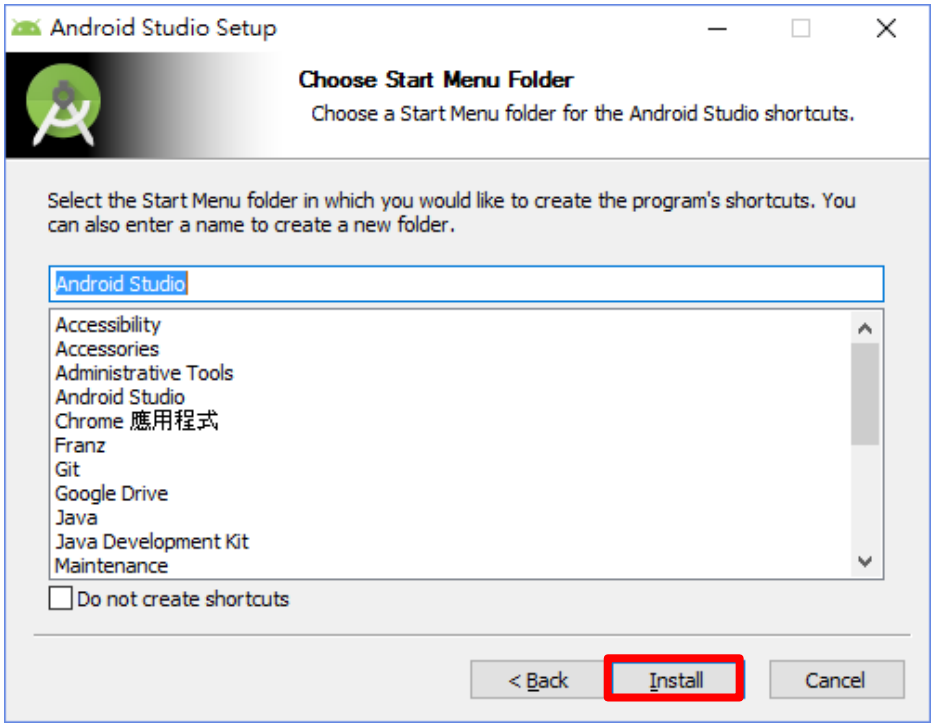

圖 1-12 設定目錄名稱

Step8 開始安裝 Android Studio，如圖 1-13 所示。

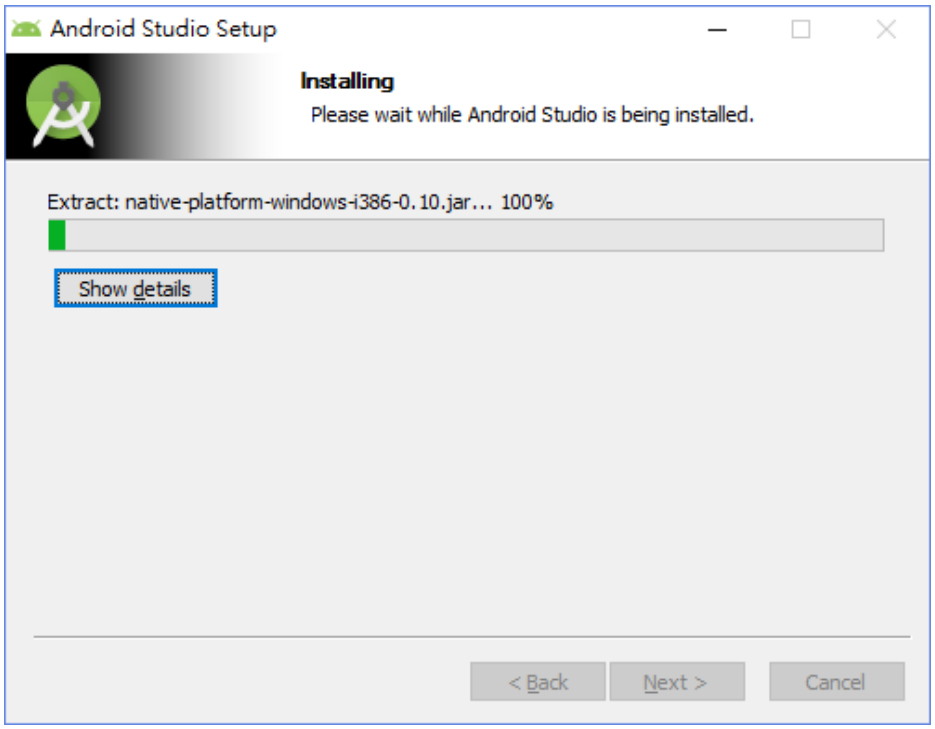

圖 1-13 開始安裝 Android Studio

1.1.3 建立 APP 專案

Step1 開啟位於開始目錄或桌面捷徑的 Android Studio 開發環境。

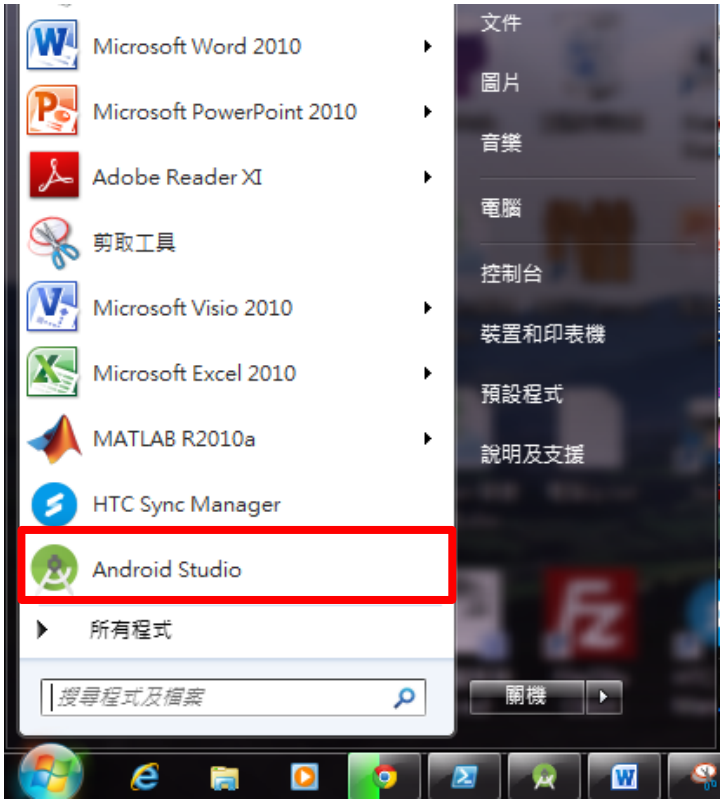

圖 1-14 開啟 Android Studio

Step2 第一次啟用會詢問是否匯入先前版本的設定，沒有的話選擇「I do not have a ...」然後點選「OK」，如圖 1-15 所示。

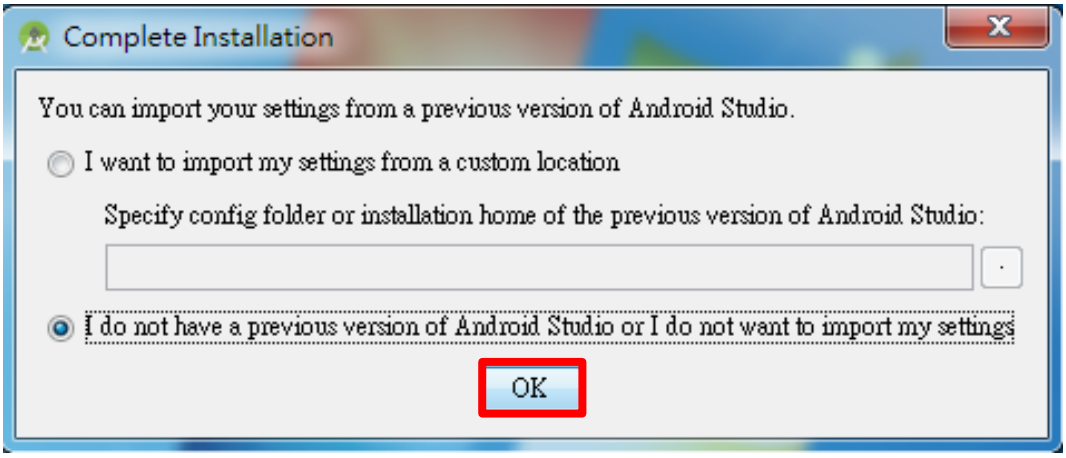

圖 1-15 是否匯入先前版本設定

Step3 等待 Android Studio 程式下載，並安裝 Android SDK，如圖 1-16 所示。

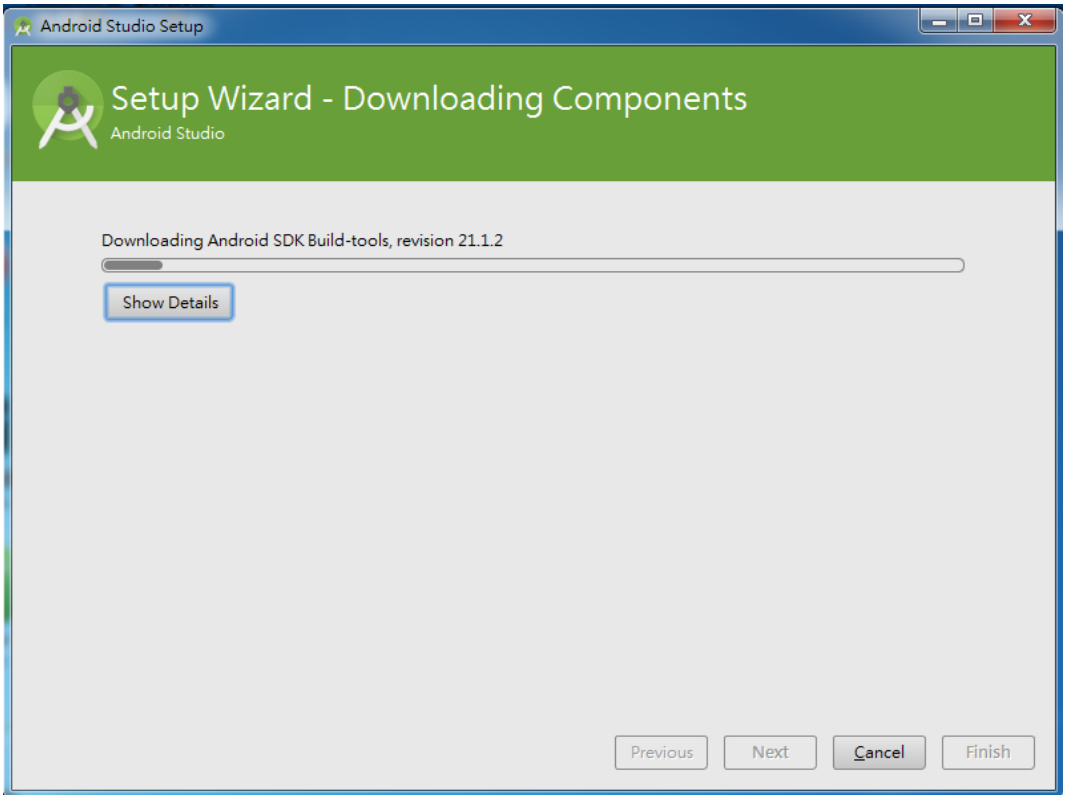

圖 1-16 安裝 Android SDK 套件

Step4 開始建立第一個 Android 專案，開啟 Android Studio 後，點選「Start a new Android Studio project」建立新專案，如圖 1-17 所示。

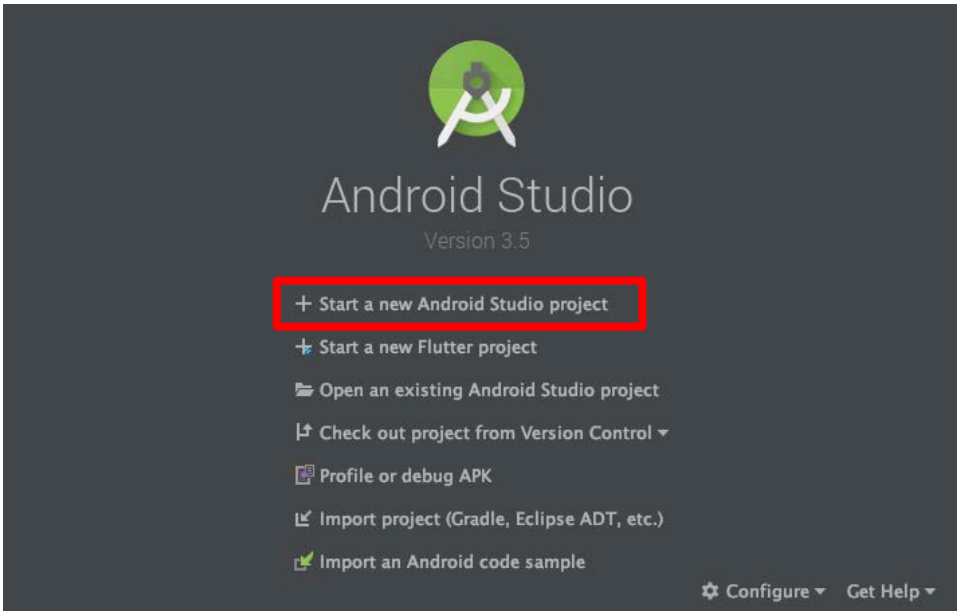

圖 1-17 開啟 Android Studio 建立新專案

Step5 根據功能需求可選擇功能模型進行開發，正常設計使用 Empty Activity 即可，設定後點選「NEXT」，如圖 1-18 所示。

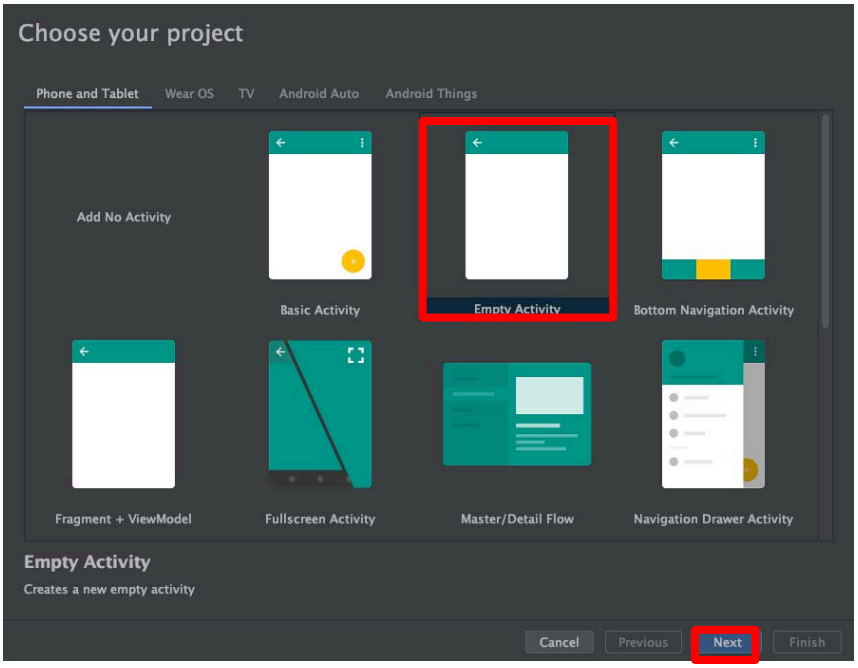

圖 1-18 選擇 Activity 樣式

Step6 設定專案名稱、專案路徑、語言與最低支援版本，點選「Finish」建立新專案，如圖 1-19 所示。

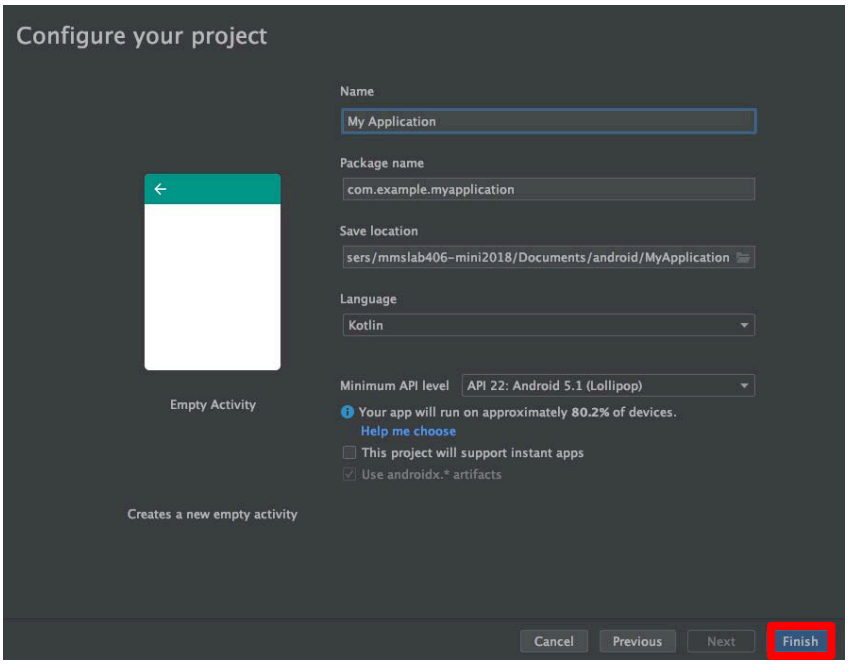

圖 1-19 設定專案屬性並建立新專案

Step7 專案建立成功後可以看到如圖 1-20 的畫面。

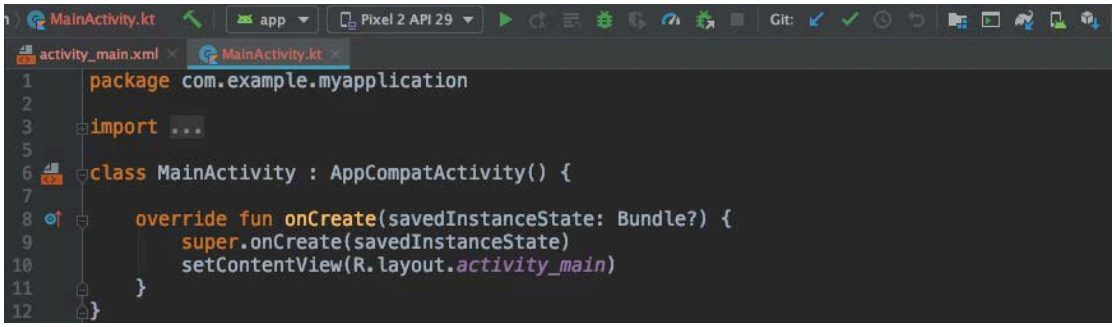

圖 1-20 專案建立成功

1.1.4 模擬器

Step1 按下圖 1-21 的 AVD Manager，AVD 位於 Android Studio 上方的工具列。

圖 1-21 按下 AVD Manager

Step2 點擊「Create a virtual device」建立一個新的模擬器，如圖 1-22 所示。

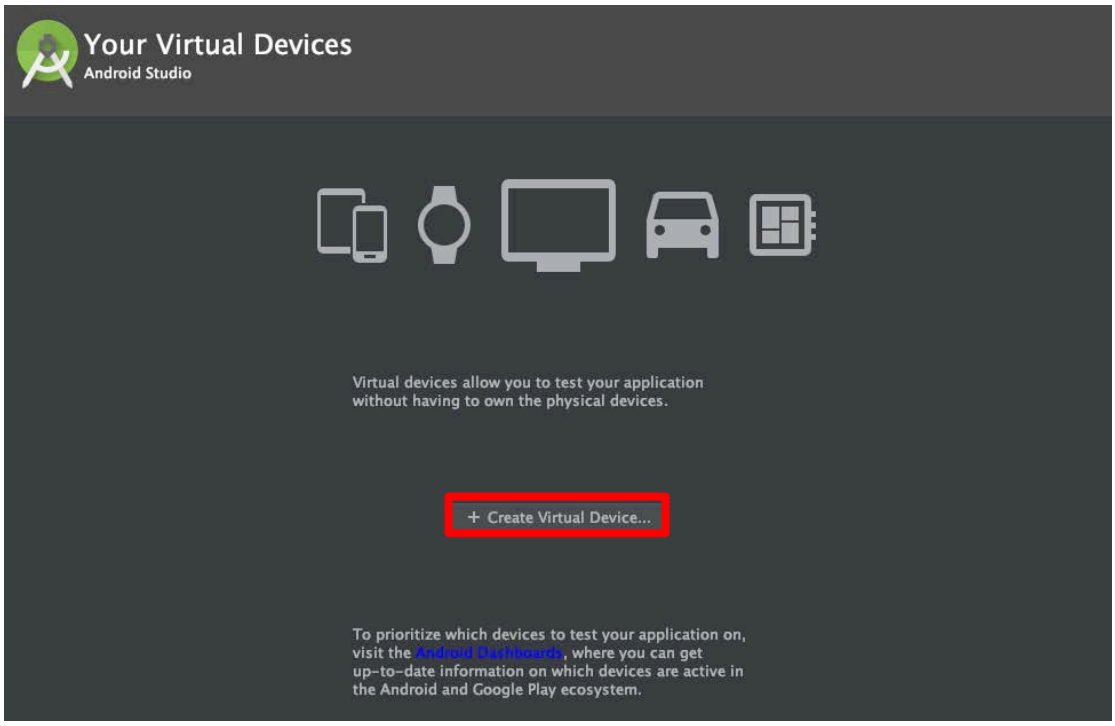

圖 1-22 建立新的 Android 模擬器

Step3 選擇模擬器的種類為 Phone，型號為 Pixel 2，接著後按「Next」，如圖 1-23 所示。

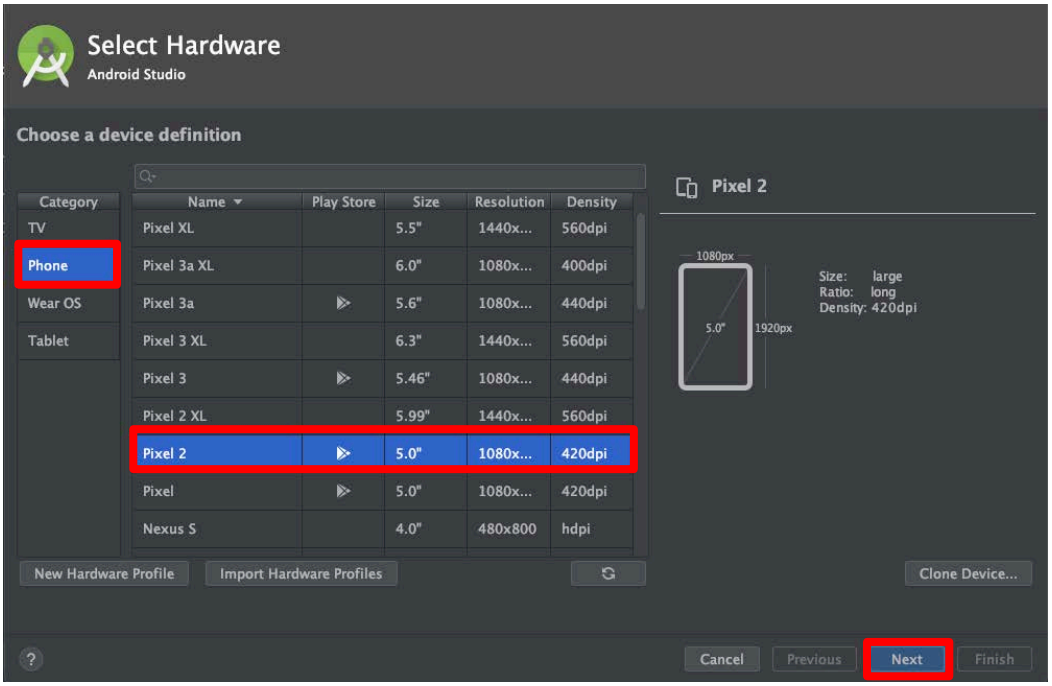

圖 1-23 選擇模擬器類型

Step4 點選在模擬器上運行的 Android 版本，如圖 1-24 所示。

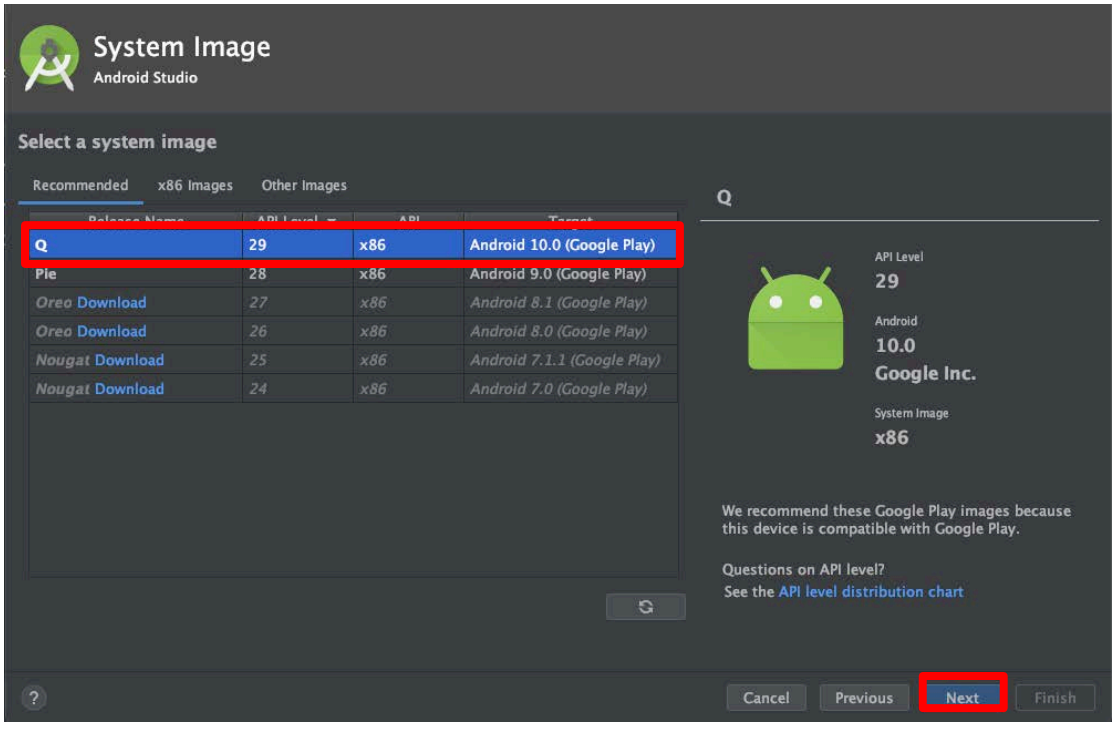

圖 1-24 選擇模擬器的 Android 版本

Step5 確認模擬器配置，點擊「Finish」完成模擬器的建立，如圖 1-25 所示。

圖 1-25 完成模擬器配置

Step6 完成後，模擬器列表會出現你剛建立的模擬器，點擊右側箭頭開啟模擬器，如圖 1-26 所示。

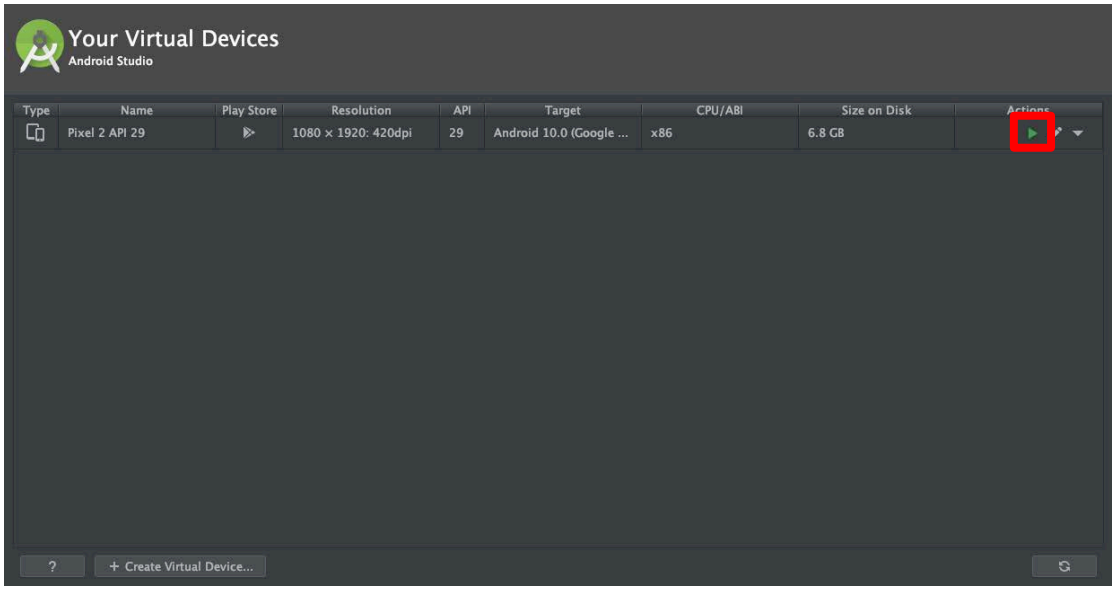

圖 1-26 模擬器列表

Step7 看到如圖 1-27 的畫面，就代表模擬器啟動完成。

圖 1-27 Android 模擬器畫面

1.1.5 執行 APP 專案

Step1 確認模擬器啟動且可運作之後，下一步開始編譯你的程式。點選位於 Android Studio 工具列中的綠色箭頭編譯程式，如圖 1-28 所示。

圖 1-28 編譯你的程式

Step2 安裝完成，預設的 Android 專案會顯示「Hello World!」，如圖 1-30 所示。

圖 1-30 專案安裝成功

1.2 Android 專案架構

建置完成 Android 應用程式專案後，Android Studio 左側表單內會顯示應用程式的配置目錄，如圖 1-31 所示，一般預設的配置模式下會是 Android 顯示配置。在 Android 顯示配置的 app 目錄之下會放置專案的主體，包含三個主要的子目錄：

- manifests－應用程式設定檔
- java－類別目錄
- res－資源目錄
- Gradle－自動化建構工具

以下分別對其介紹。

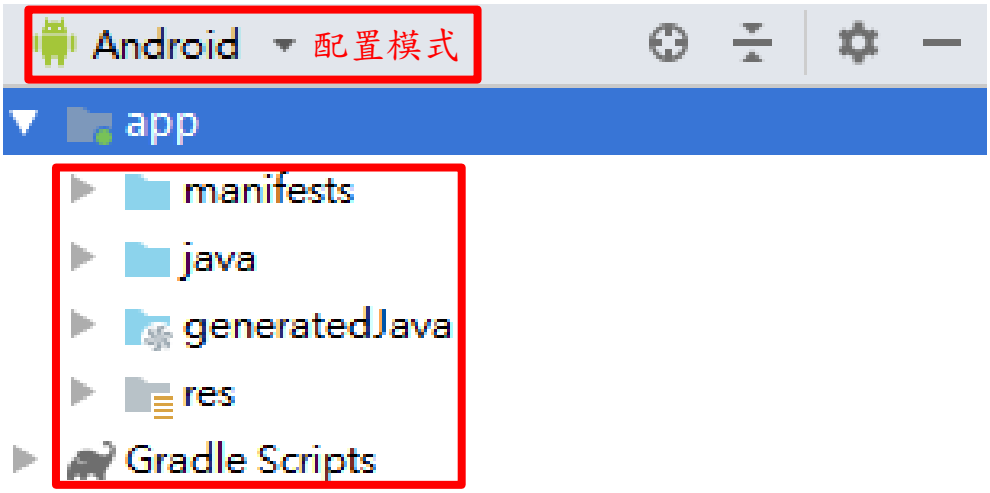

圖 1-31 Android 專案配置目錄

說明

Android 顯示配置的目錄不等於實際目錄。實際目錄需切換至 Project 顯示配置。

1.2.1 應用程式設定檔－AndroidManifest.xml

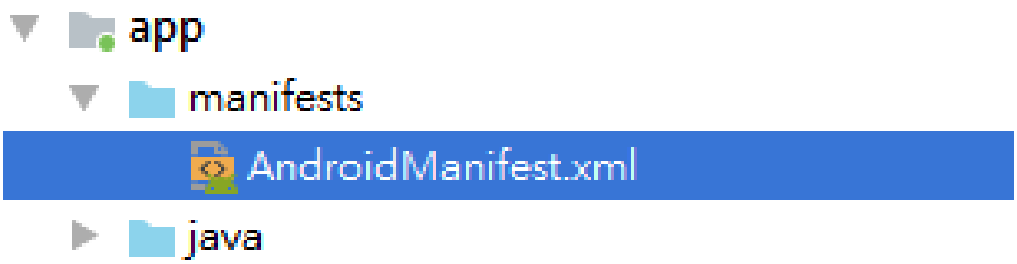

圖 1-32 應用程式設定檔 AndroidManifest

於 manifests 目錄下，展開後可見 AndroidManifest.xml，每一個應用程式的根目錄都必須包含圖 1-32 的 AndroidManifest.xml 檔案。系統在執行該應用程式的程式碼之前，需要向 Android 系統宣告應用程式的基本資訊，如應用程式的標題、圖示等，而這些將會被描述在 AndroidManifest.xml 之中。

詳細可見：https://developer.android.com/guide/topics/manifest/manifest-intro.html

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        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>
```

一個基本的 AndroidManifest.xml 中的描述的宣告內容如下：

- package：為應用程式的 Java 封裝命名。上架發佈的應用程式中 package 必須是唯一的，不能與其他應用程式有所重複，可以將 package 想像程式用於辨識或搜尋以發布的應用程式的識別名稱。

- application：用於定義應用程式相關的元件，設計中的屬性可預設應用程式的基本資訊，如下圖 1-33 所示。
  - 「android:icon」定義應用程式的圖示，預設為 Android 機器人圖示。
  - 「android:label」定義應用程式的標題，預設為專案建置時所設的名稱。
  - 「android:theme」定義應用程式的主題風格，也可說是應用程式的基本樣式，其設定於 application 中將會預設給所有子頁面。

圖 1-33 APP 的 Icon 與 Theme 樣式

- activity：「application」底下需要描述應用程式在執行中會使用到的所有類別方法，activity 為其中的 Activity 類別。
  - 包含使用者介面「activity」、後台服務「Service」、廣播「Broadcast」，這些類別在被執行前，系統會去查閱「application」是否有對應的描述，如果有個 activity 要使用，但是卻沒有描述在「application」之中，那系統將無法呼叫該 activity，這是初學者最常犯的錯誤之一。
  - 一個新創建的專案，一開始系統就會產生一個「MainActivity」的 activity 類別，需要更多的 activity 或是其他類別就需要手動增加。
  - 類別名稱的描述必須包含 package 名稱，如「demo.myapplication.MainActivity」，不過如果該類別屬於同個 package，則可省略為「.MainActivity」。

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    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>

    <reciver
        android:name=".MyReciver"
        android:enabled="true"
        android:exported="true"></reciver>
</application>
```

如果要使用 Service 或 receiver 元件就需要在此處加入對應的描述。
此章節中不需要加入。

1.2.2 java－類別目錄

android 的主要語言為 Java 與 Kotlin，所有的程式碼都會被描述成類別結構放置於「src」目錄下，在 Android 顯示配置下則會顯示於圖 1-34「java」目錄。

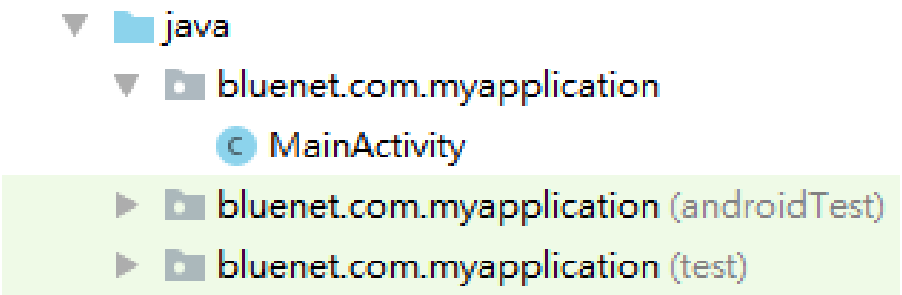

圖 1-34 專案的 Java 目錄

為了讓使用者能夠快速地對 android 進行開發，Google 提供了相關的 SDK 套件，SDK 套件提供使用者開發上的基礎框架，使用者可直接套用框架實作其功能。

如最常見使用的 activity 類別，使用者端可見的程式碼僅有短短數行，執行

後便可產生介面，實際上應用程式當然不可能如此簡單就能產生畫面，能實現此功能的主要原因是透過繼承名為「AppCompatActivity」的類別框架，該框架中本身就有編寫能產生畫面的完整程式，而透過繼承，使用者可以完全不需要對於產生畫面的部分作程式編寫，僅需要引用內建的程式資源並在其之上加入自己的客製化程式碼，就可以簡單的實作自己的應用程式。

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
                              繼承 AppCompatActivity 原生框架
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        可以客製化修改的程式部分
        setContentView(R.layout.activity_main);
    }
}
```

1.2.3 res－資源目錄

android 的專案中，除了程式碼之外，還包含其他的專案資源，如放置於應用程式內部的圖檔、版面配置、顏色、風格主題等等，都算是專案的資源之一，會被放置於圖 1-35「res」目錄下。

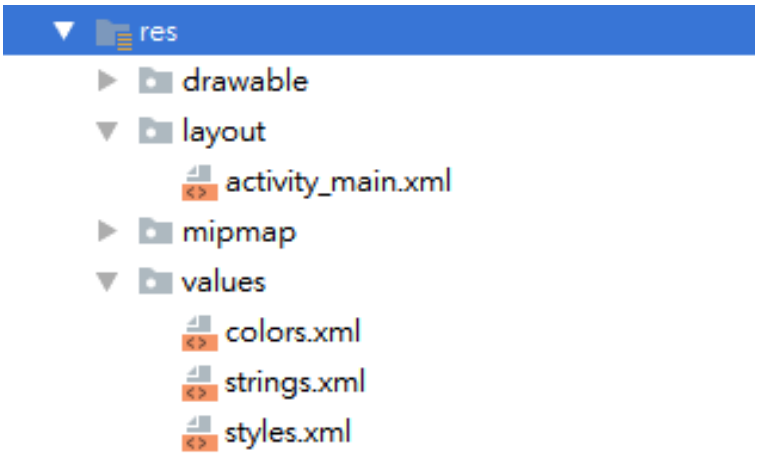

圖 1-35 專案的 res 資源目錄

依據用途的不同，最少又會區分成「drawable」、「layout」與「value」三個目錄，以下做簡單的說明：

- drawable：當應用程式需要使用圖檔時，會統一將需要的圖檔至於圖 1-36 目錄之下，圖檔資源可為「.png」或「.jpg」，或以 xml 格式設計的素材，也會放置於此目錄下。

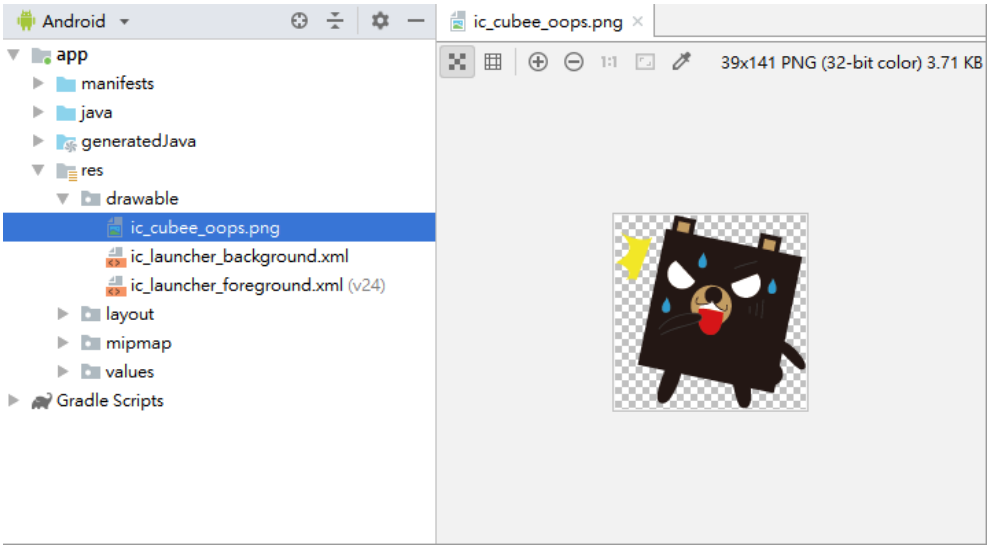

圖 1-36 圖檔 drawable

- layout：為使用者介面的版面配置檔，像是成形的畫面、按鈕畫面等等會放置於此目錄下。

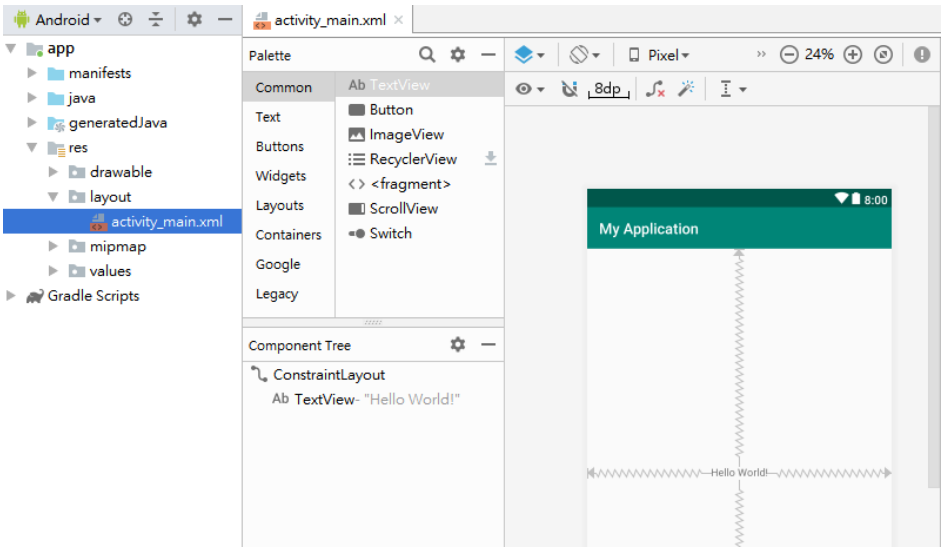

圖 1-37 布局 layout

- value：應用程式中可能使用的變數值，可放於此圖 1-38 目錄中，根據性質不同又可被分為字串（string）、顏色（color），區域大小（dimes）、

風格（styles）等。

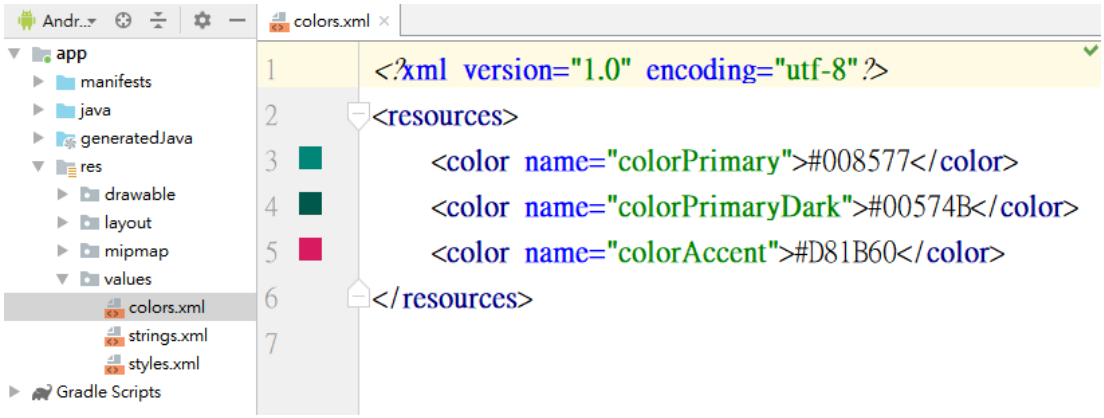

圖 1-38 數值 Values

資源目錄在有新檔案加入或修改後，使用該資源的方法有分成 XML 的形式與程式碼的形式。

在 XML 中，我們使用「@目錄/檔名」的命名描述指定特定資源，如我們要使用 value 中的字串（string），可以透過「@string/app_name」得到對應的字串。

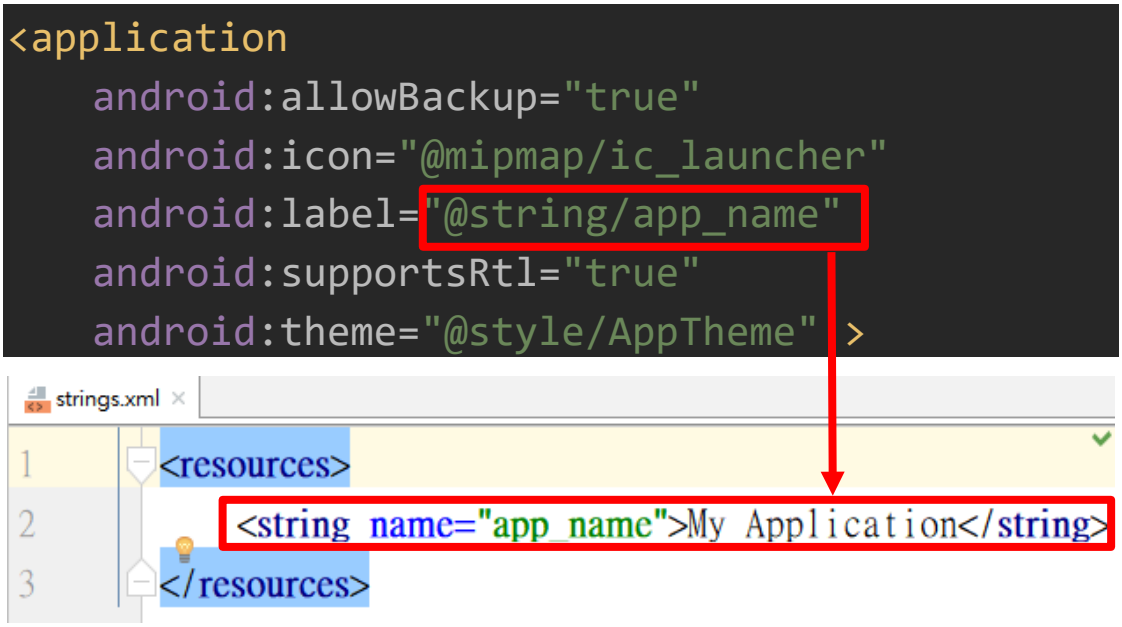

圖 1-39 字串資源 Strings

在 Android 中則需要透過 R 類別來連接資源。R 類別由系統自動產生，系統會分配資源目錄下的所有檔案一組路徑，並被描述在 R 類別之中，可以用 R.id.xxx 的命名描述去指定特定資源，如圖檔（R.drawable.xxx）、版面配置（R.layout.xxx）與字串（R.string.xxx），透過命名描述去查閱 R 類別的對照表去找到的實體路徑。

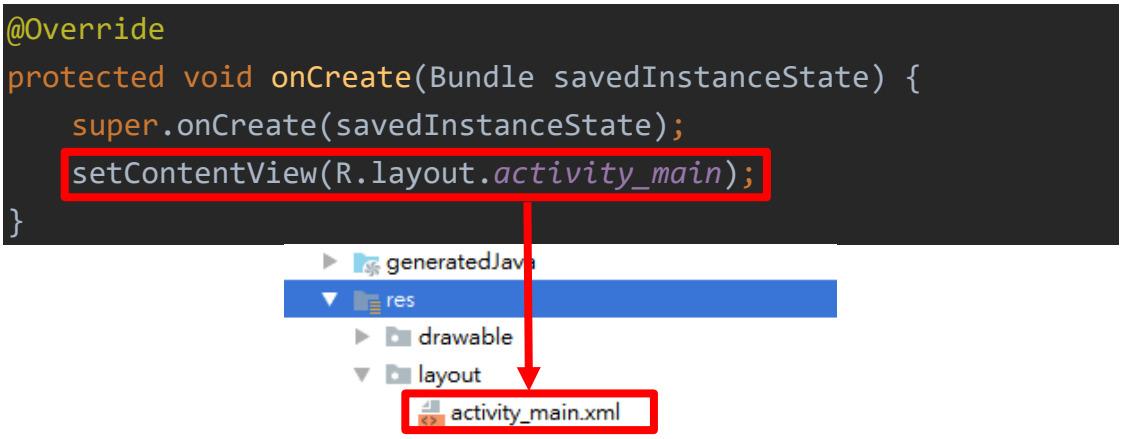

圖 1-40 透過 R 類別對照版面配置文件

1.2.4 Gradle－自動化建構工具

Gradle 是一個基於 Apache Ant 和 Maven 概念的專案自動化建構工具，在 Android Studio 中負責管理專案的設定如圖 1-41 所示，其中包含模組（Module）的設定檔，混淆碼規則與本地設定檔。

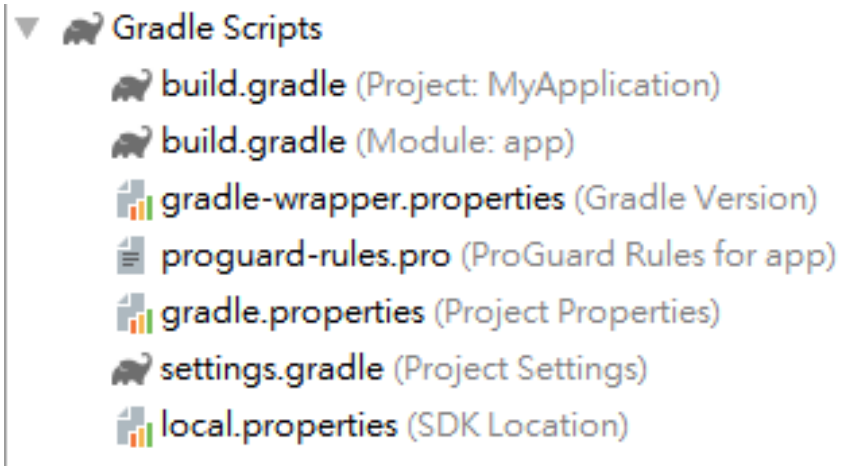

圖 1-41 自動化建置工具 Gradle

- build.gradle：程式建構文件。在 Android Studio 中，每項專案可擁有多個模組（Module），而每個模組（Module）都會有一個自己的設定檔，用來紀錄模組（Module）所需的屬性、簽署或是依賴項目。

```
apply plugin: 'com.android.application'  擴充

android {
    compileSdkVersion 28
    defaultConfig {
        ...
    }
    建置設定
    buildTypes {...}
}
依賴項目
dependencies {...}
```

專案的基本設定，包含專案的識別名稱、支援的最低與最高 Android 版本以及專案自身版本

- gradle-wrapper.properties：gradle-wrapper 配置文件。一般這個文件是自動產生，開發者無須更動，除非你想手動指定 gradle 的版本。

```
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
distributionUrl=https\://services.gradle.org/distributions/gradle-4.6-all.zip
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
```

- proguard-rule.pro：程式混淆規則配置文件。在 APP 發佈上架的過程中，保護程式碼是很重要的一步，透過 Proguard 將對類別、屬性和方法進行命名，增加反編譯後程式閱讀的難度，同時也可以降低 apk 檔案大小。
- gradle.properties：Gradle 設定文件。專門用來設定全域資料，將敏感信息存放在 gradle.properties 中，可以避免將其上傳到版本控制系統。
- settings.gradle：程式設定文件。管理專案中的模組（Module），當我們要使用其他的 Module 時，也必須在 settings.gradle 中加入該模組（Module）的路徑。
- local.properties：本地設定文件。在實作專案的時候，通常會有些屬性是僅限於個人或開發用，不需要或是被禁止上傳到 GitHub 的屬性，例如 SDK path 或 developer key 之類的，這邊就提供將這類屬性定義在 local.properties 以供使用。