Lab1
Android 環境建置與專案架構

本節目的：

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

1.1 Android 環境建置

1.1.1 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 1-1 所示。

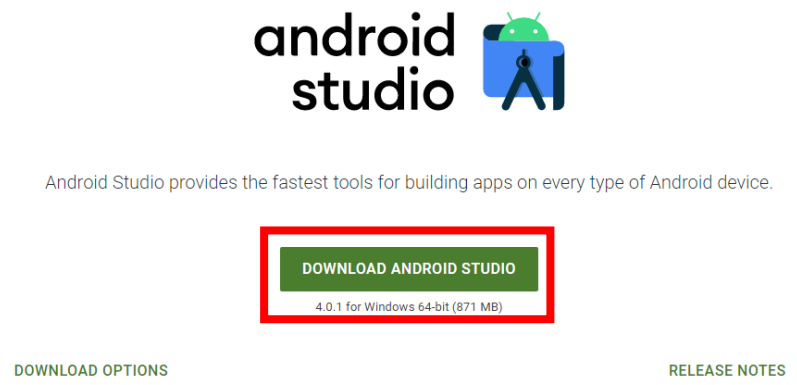


圖 1-1 下載 Android Studio

Step2 閱讀並同意 Android Studio 下載聲明，如圖 1-2 所示。

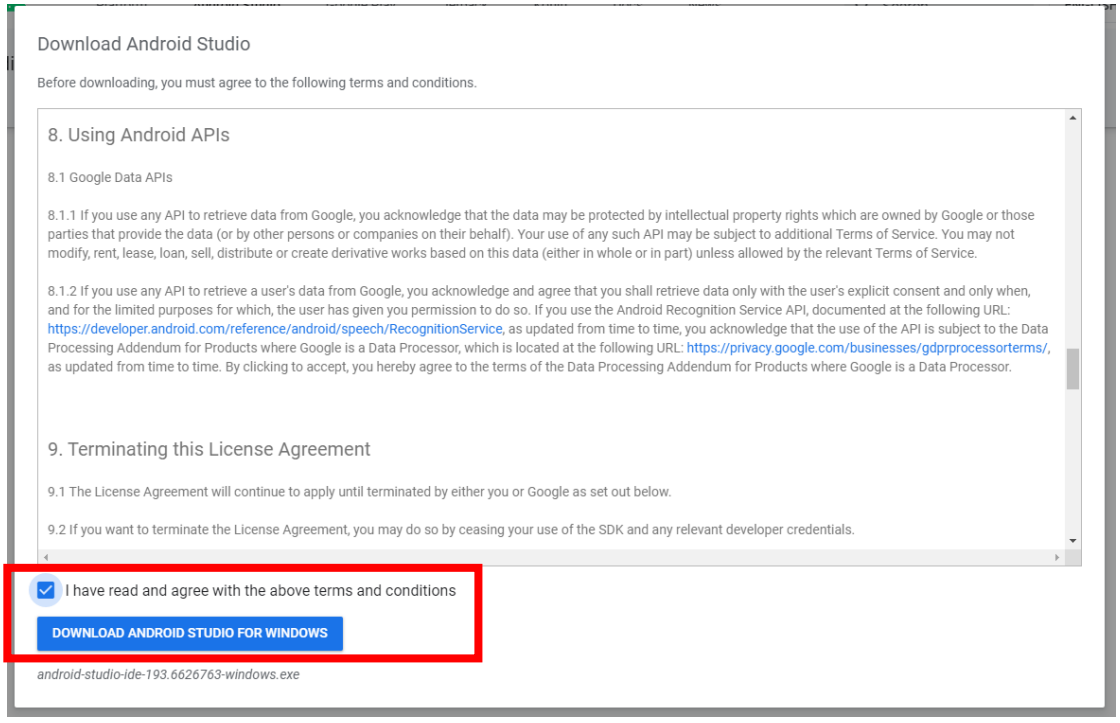


圖 1-2 閱讀並同意下載聲明

Step3 開啟 Android Studio 安裝檔，然後點擊「NEXT」，如圖 1-3 所示。

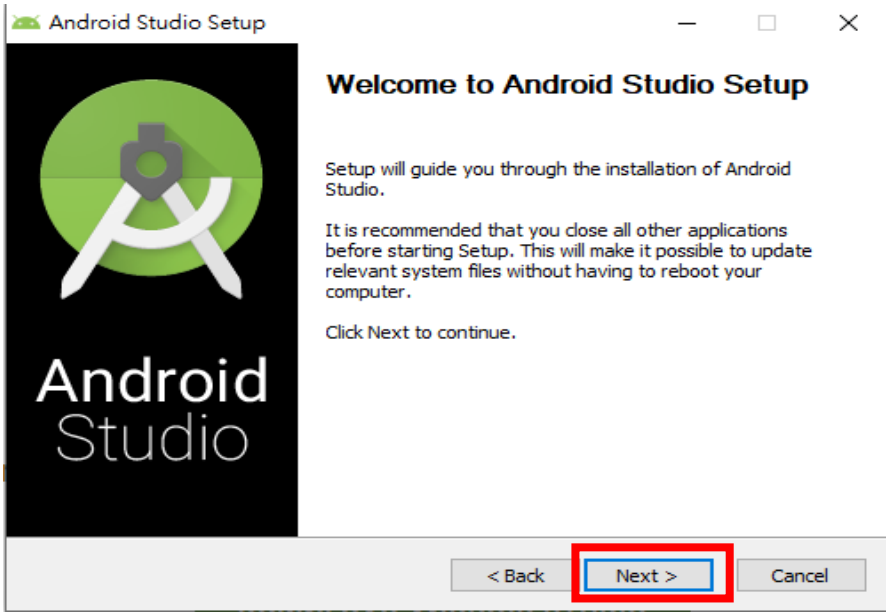


圖 1-3 開始安裝 Android Studio

Step4 選擇安裝套件，此處勾選「Android Virtual Device」，如圖 1-4 所示。

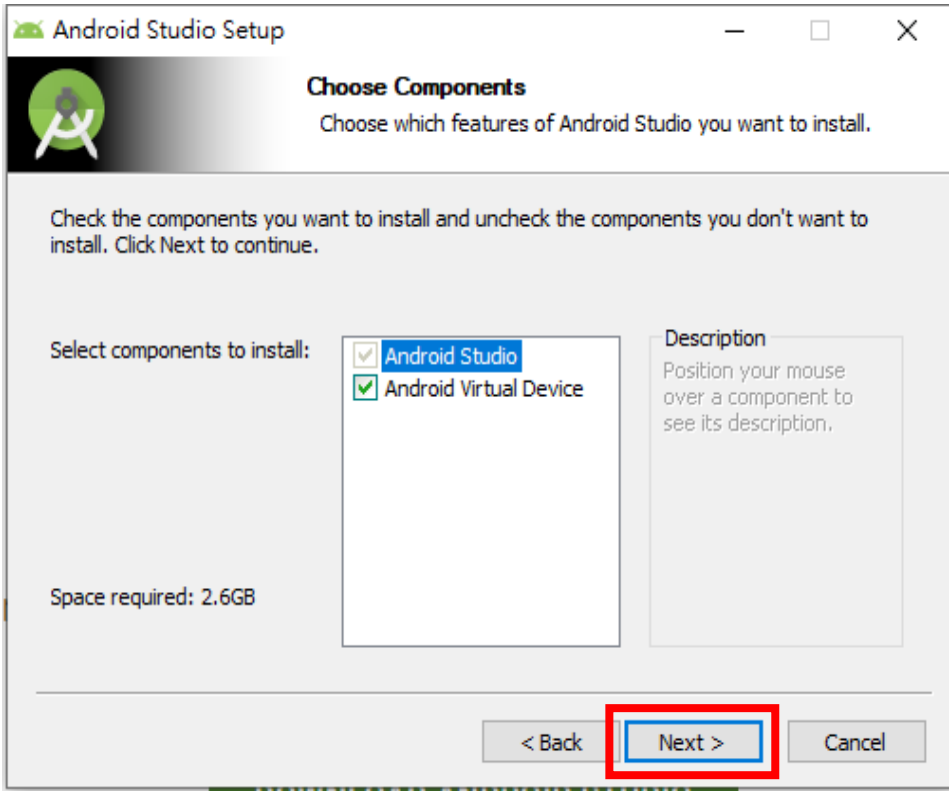


圖 1-4 選擇安裝 AVD 套件

Step5 選擇檔案位置下一步，如圖 1-5 所示。

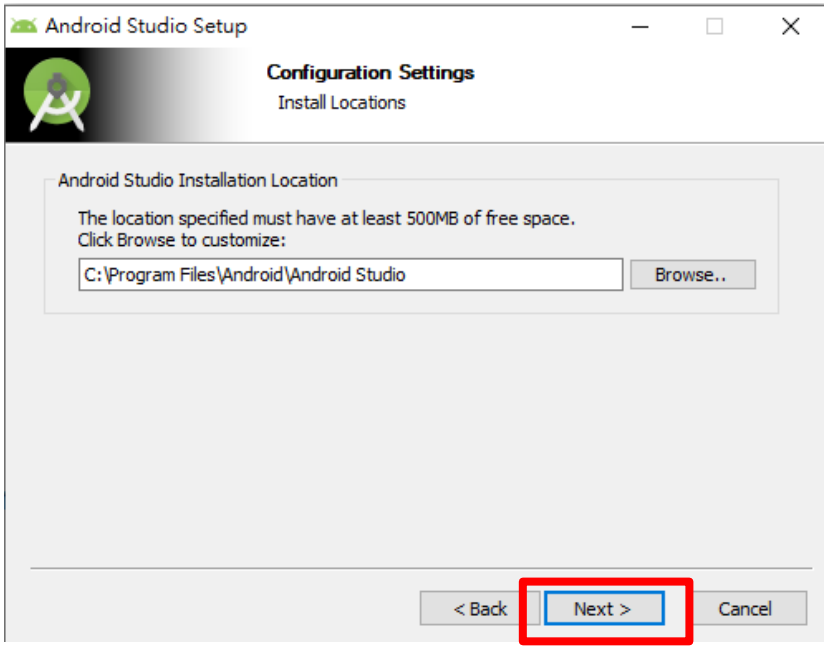


圖 1-5 設定安裝路徑

Step6 開始安裝 Android Studio，如圖 1-6 所示。

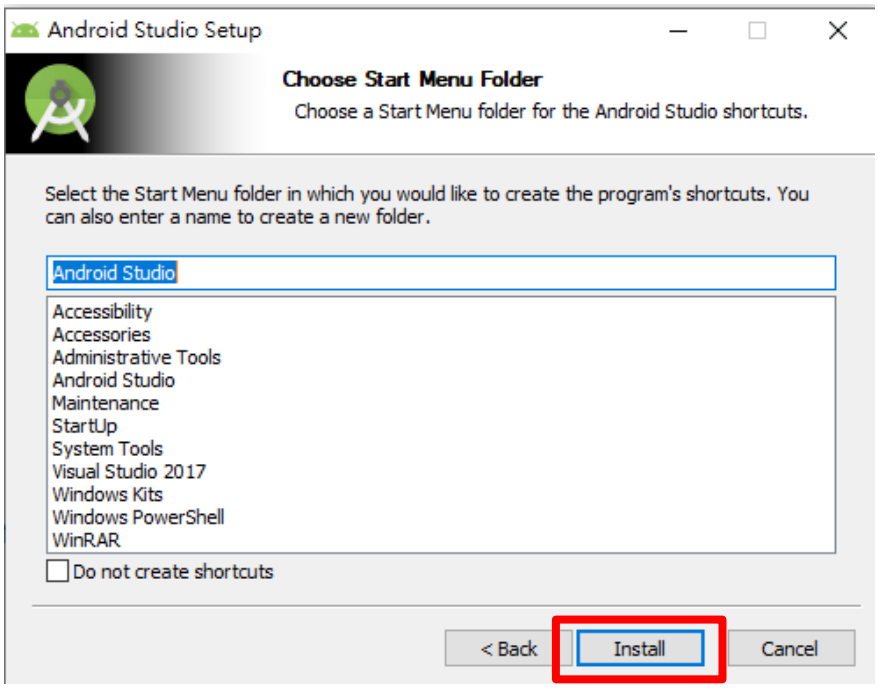


圖 1-6 安裝

1.1.2 建立模擬器

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

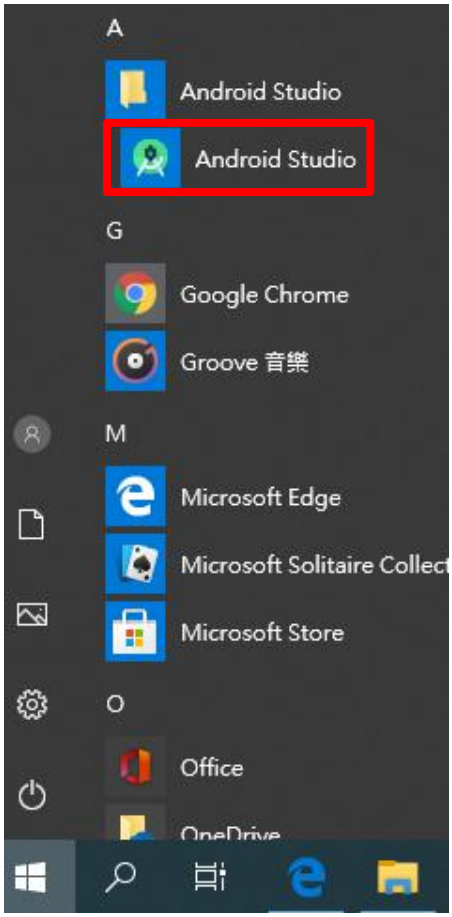


圖 1-7 開啟 Android Studio

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「Do not import ...」然後點選「OK」，如圖 1-8 所示。

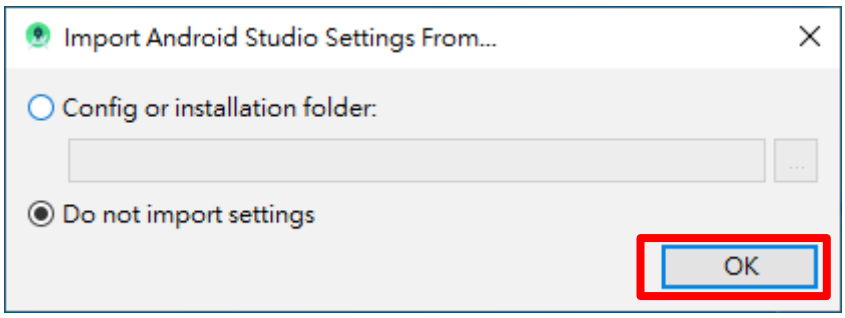


圖 1-8 是否匯入先前版本設定

Step3 Android Studio 程式下載完成，基本環境設定，如圖 1-9 所示。

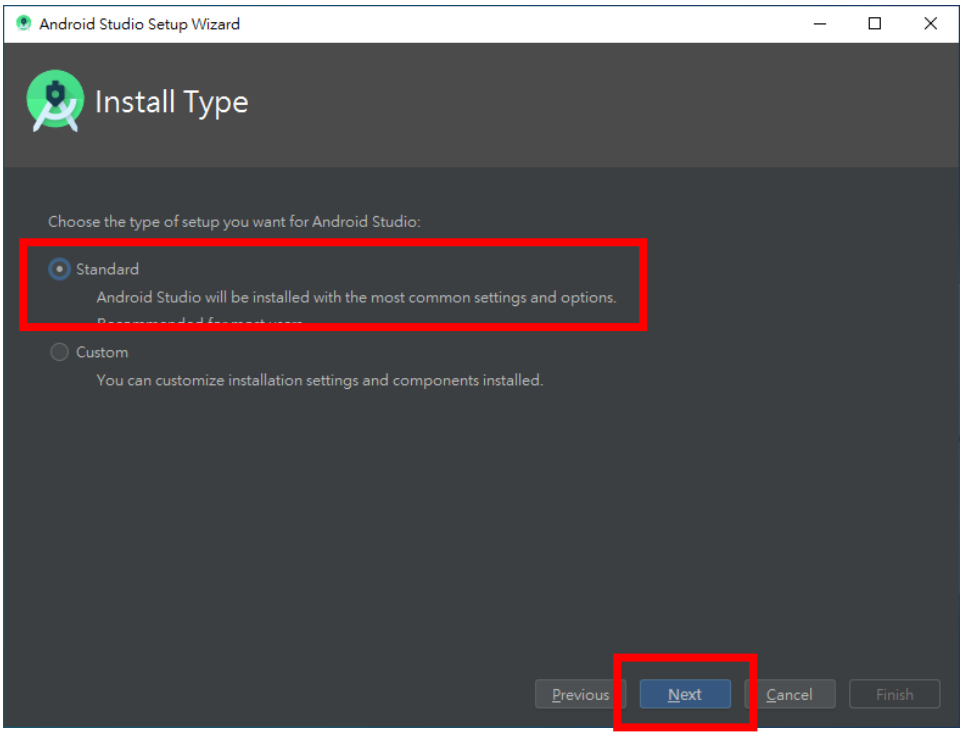


圖 1-9 選擇標準模式

Step4 基本環境設定，設定背景色，如圖 1-10 所示。

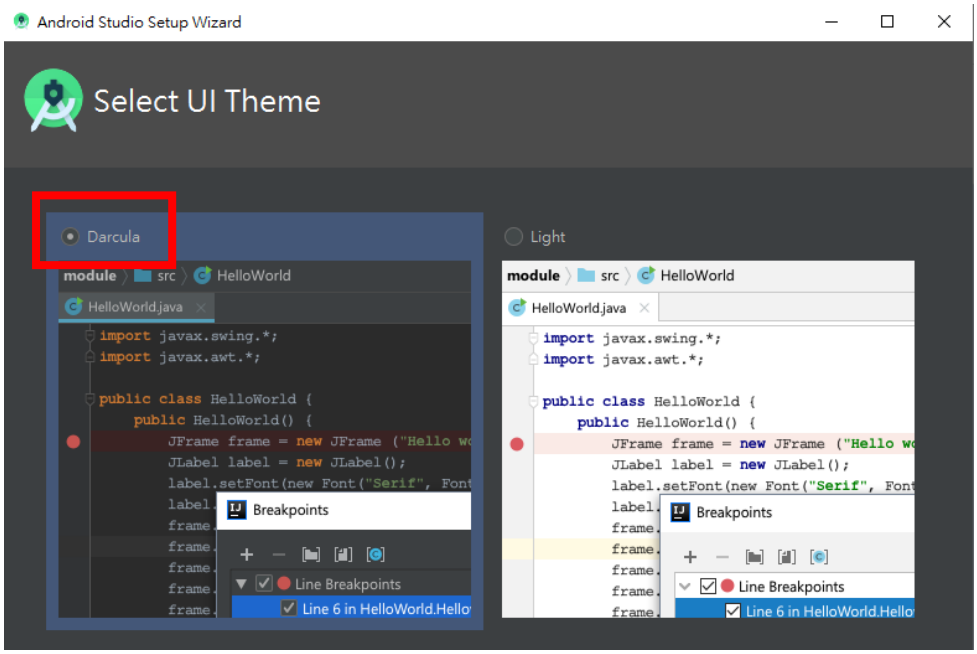


圖 1-10 選擇深色畫面

Step5 基本環境設定完成，下一步，如圖 1-11 所示。

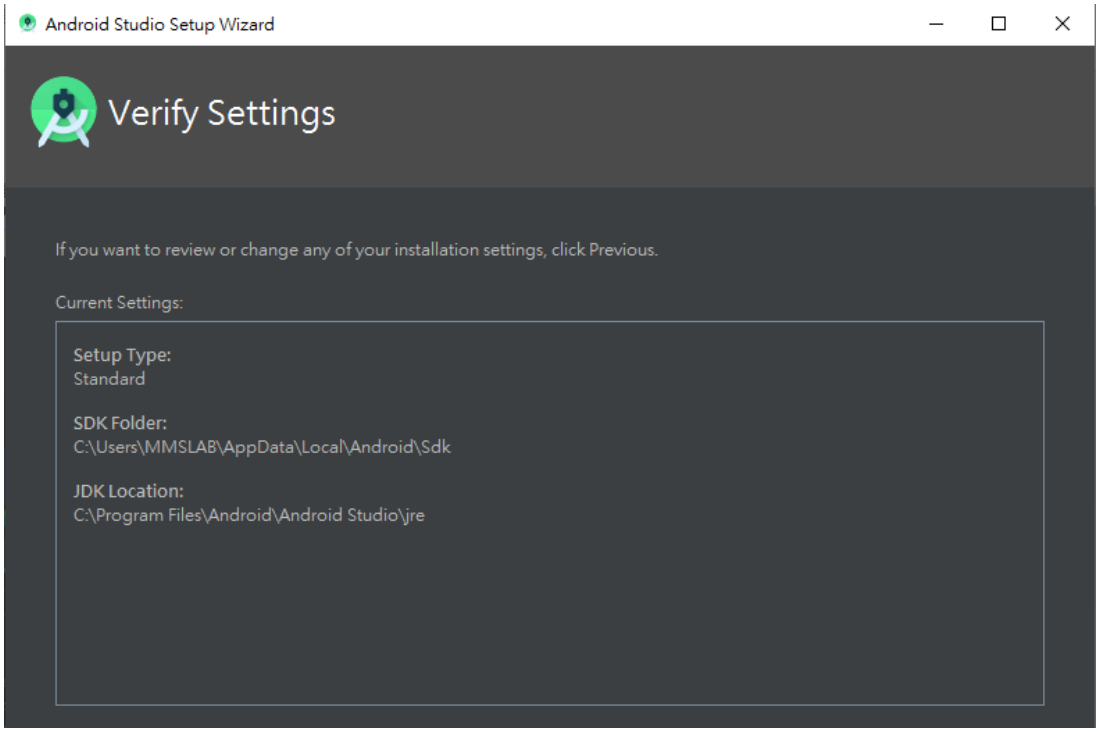


圖 1-11 基本環境安裝完成

Step6 安裝模擬器

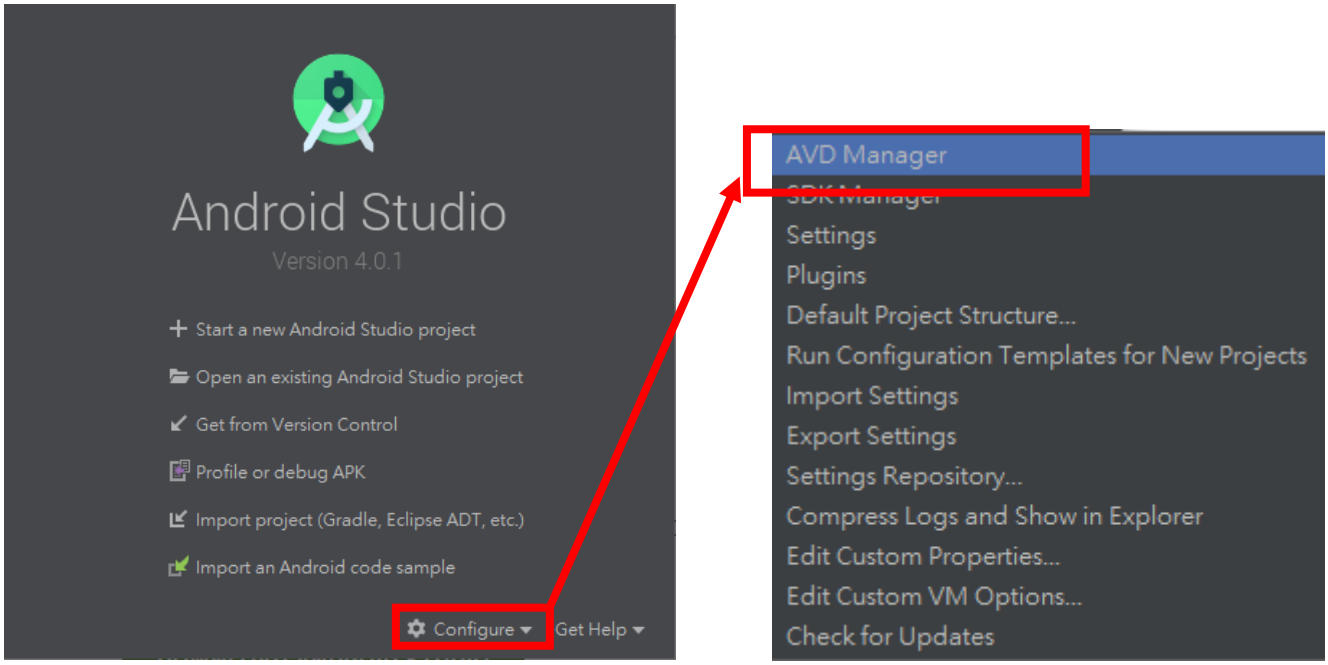
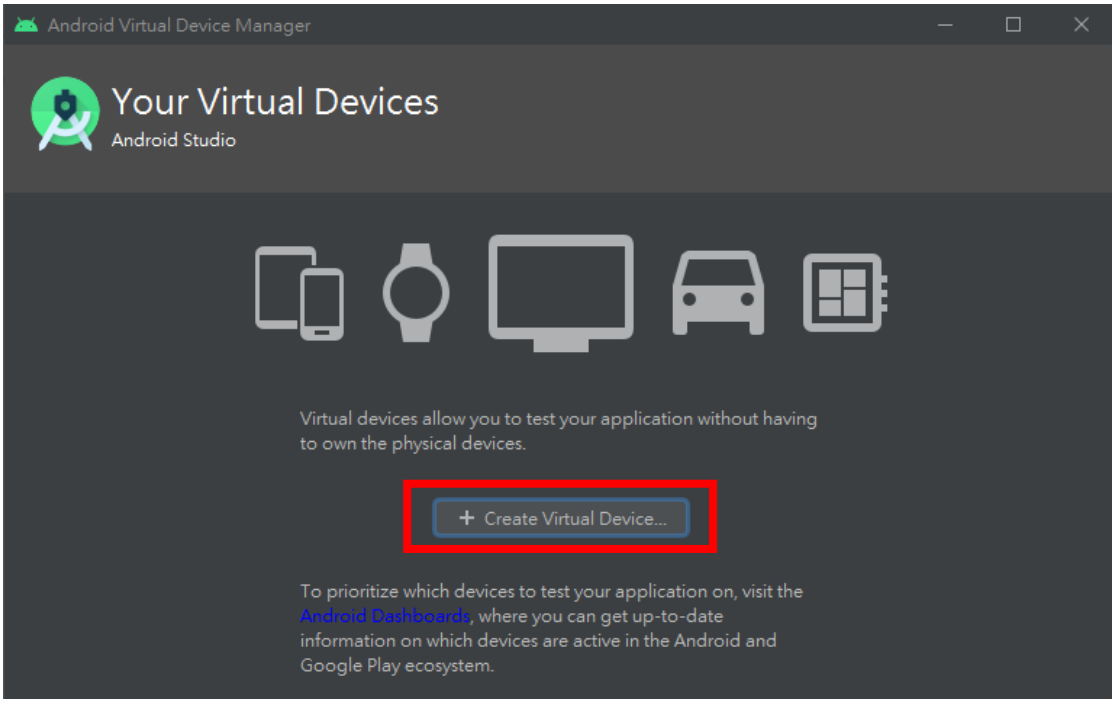


圖 1-12 安裝模擬器

Step7 建立模擬器，選擇手機類型。



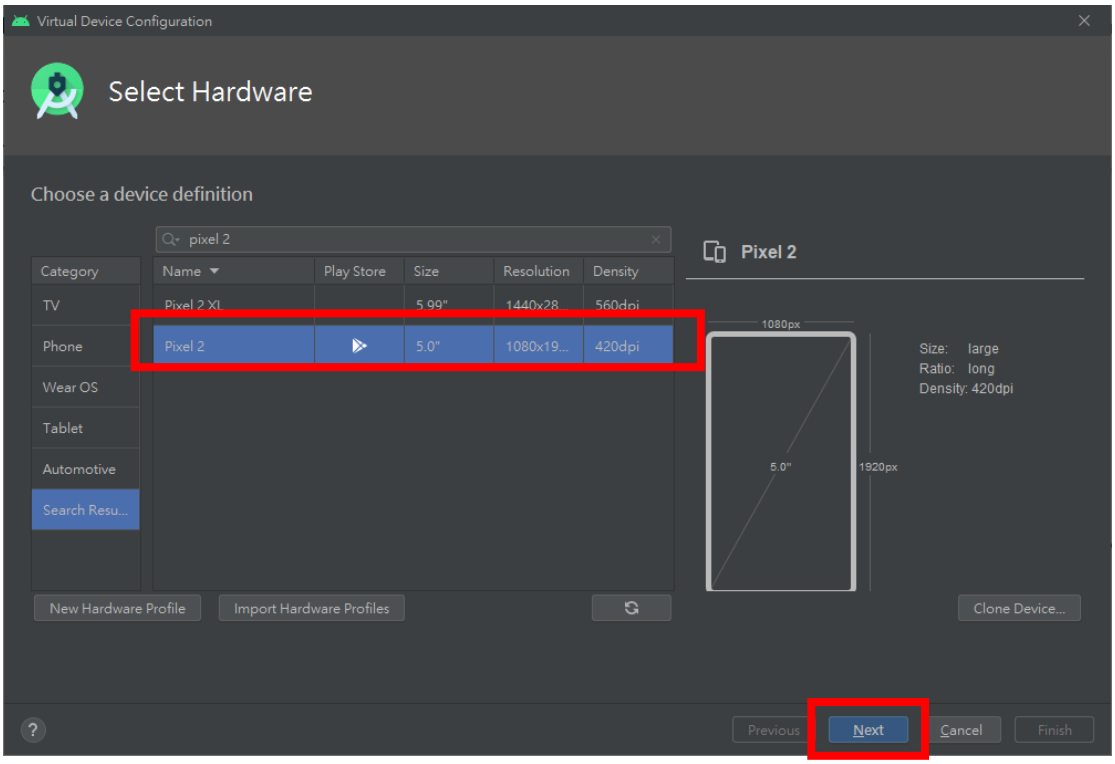


圖 1-13 建立模擬器

Step8 下載作業系統版本。

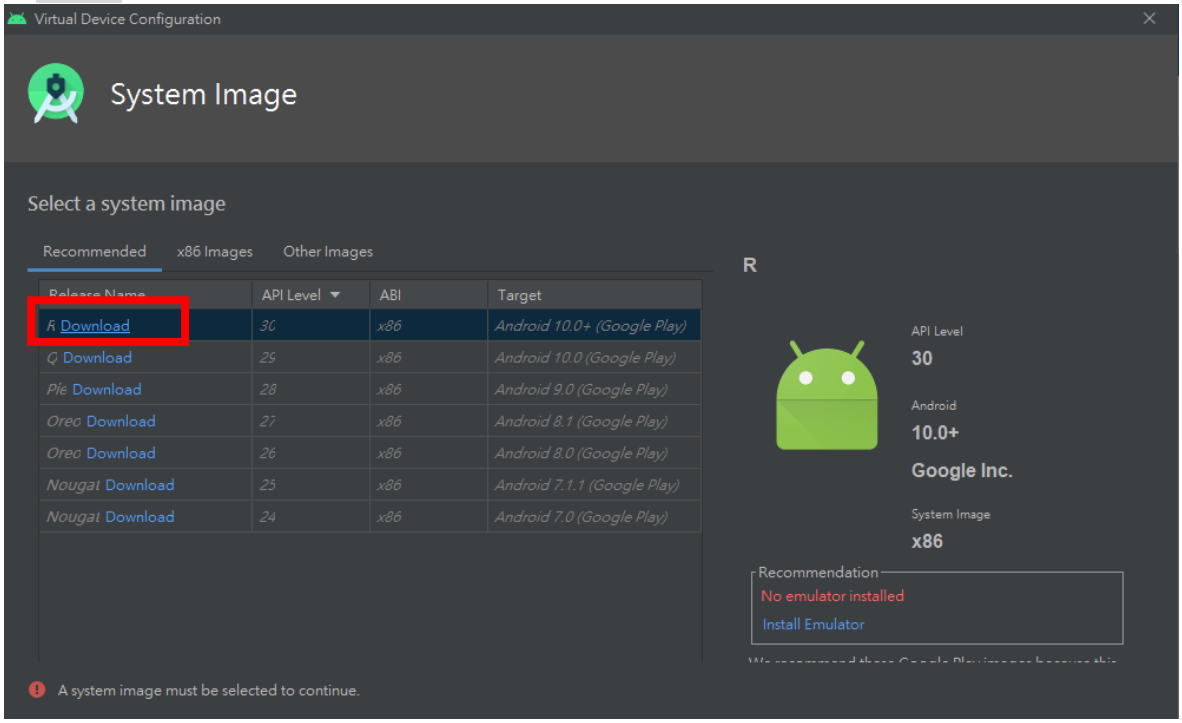


圖 1-14 下載作業系統版本

Step9 模擬器下載，如圖 1-15 所示。

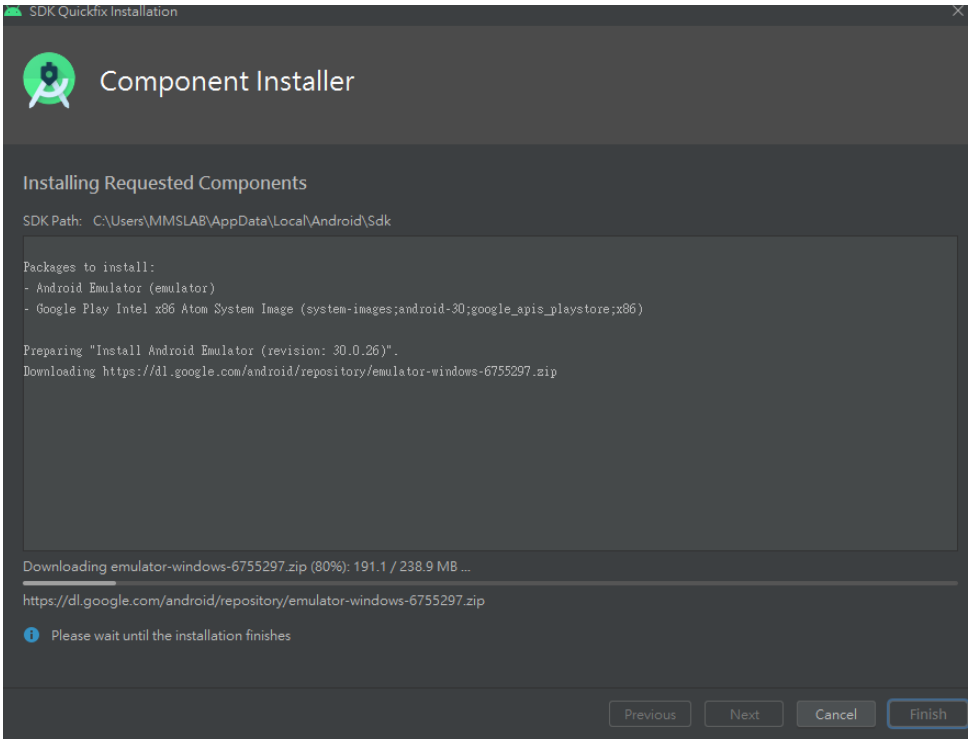


圖 1-15 安裝

Step10 選擇作業系統版本，如圖 1-16 所示。

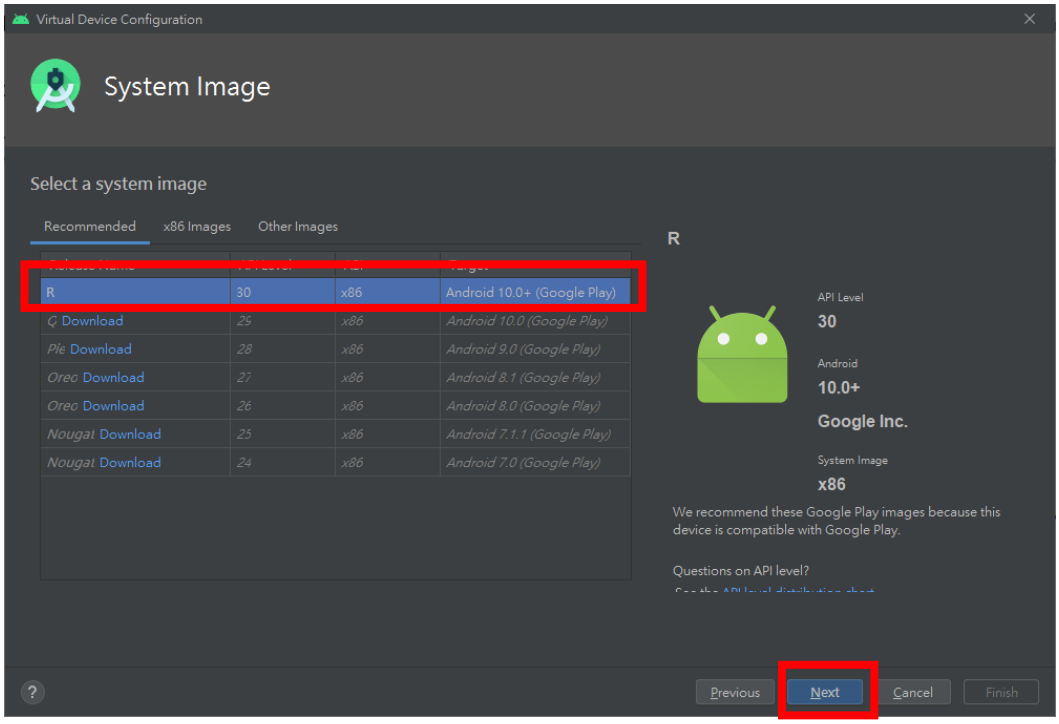


圖 1-16 選擇作業系統版本

Step11 下載硬體加速器(Haxm)，如圖 1-17 所示。

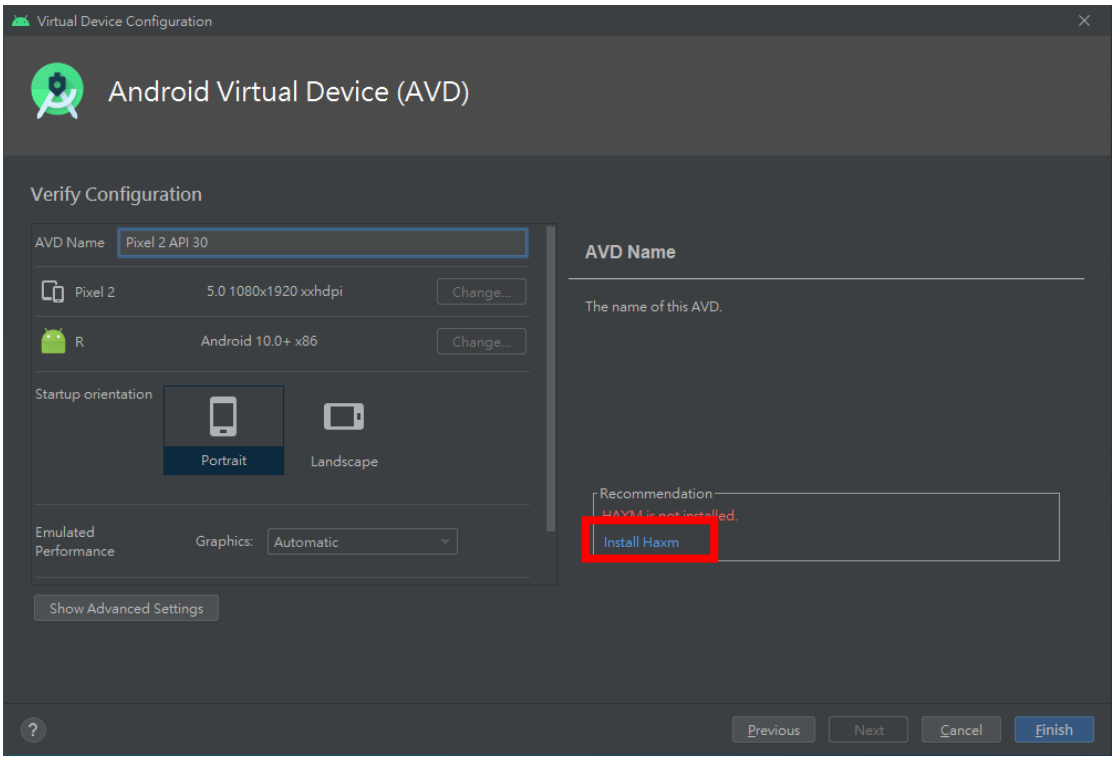


圖 1-17 選擇 Install Haxm

Step12 記憶體選擇 2GB，如圖 1-18 所示。

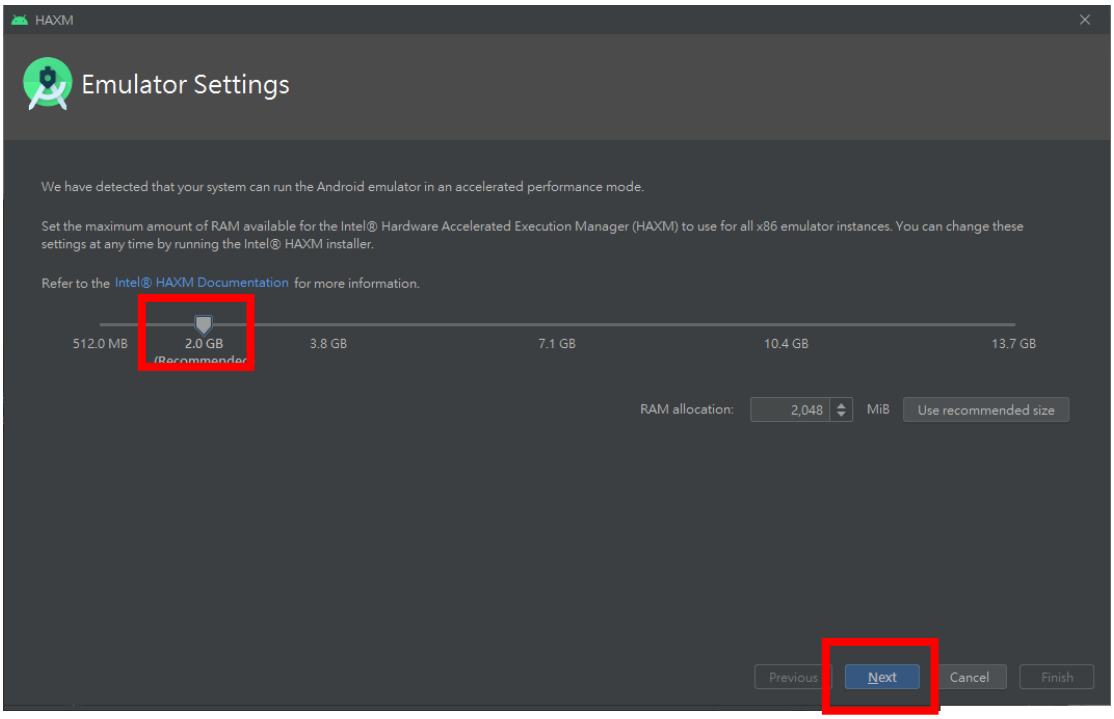


圖 1-18 記憶體選擇 2GB

Step13 下載完成，如圖 1-19 所示

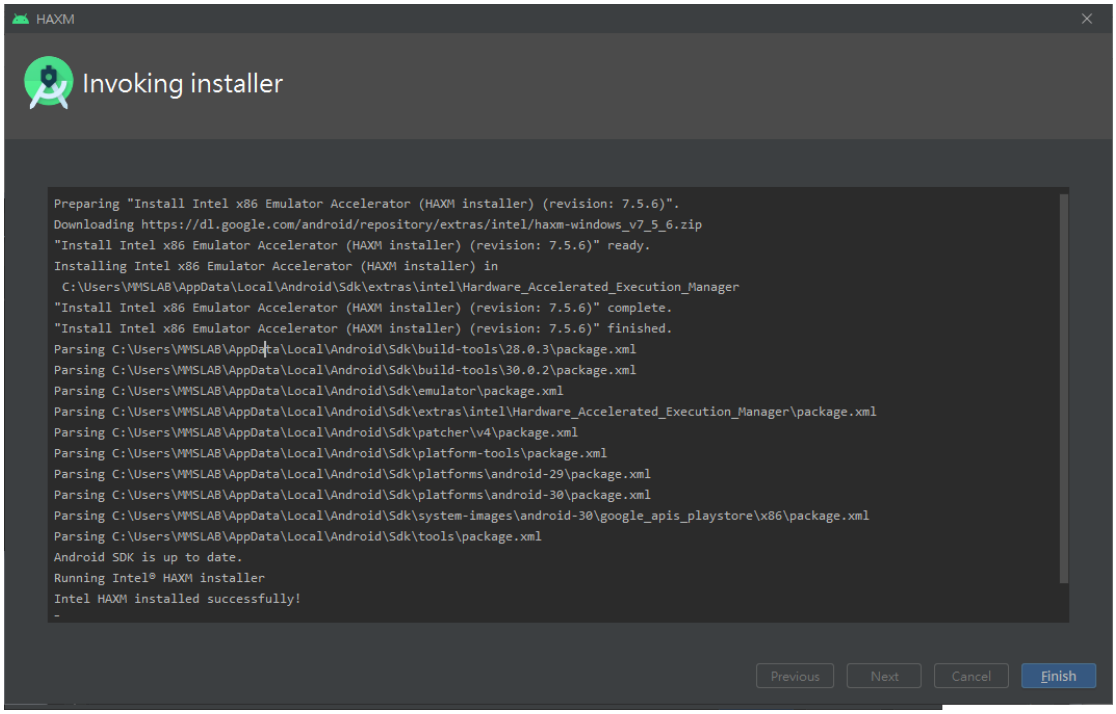


圖 1-19 硬體加速器安裝完成

Step14 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 1-20 所示。

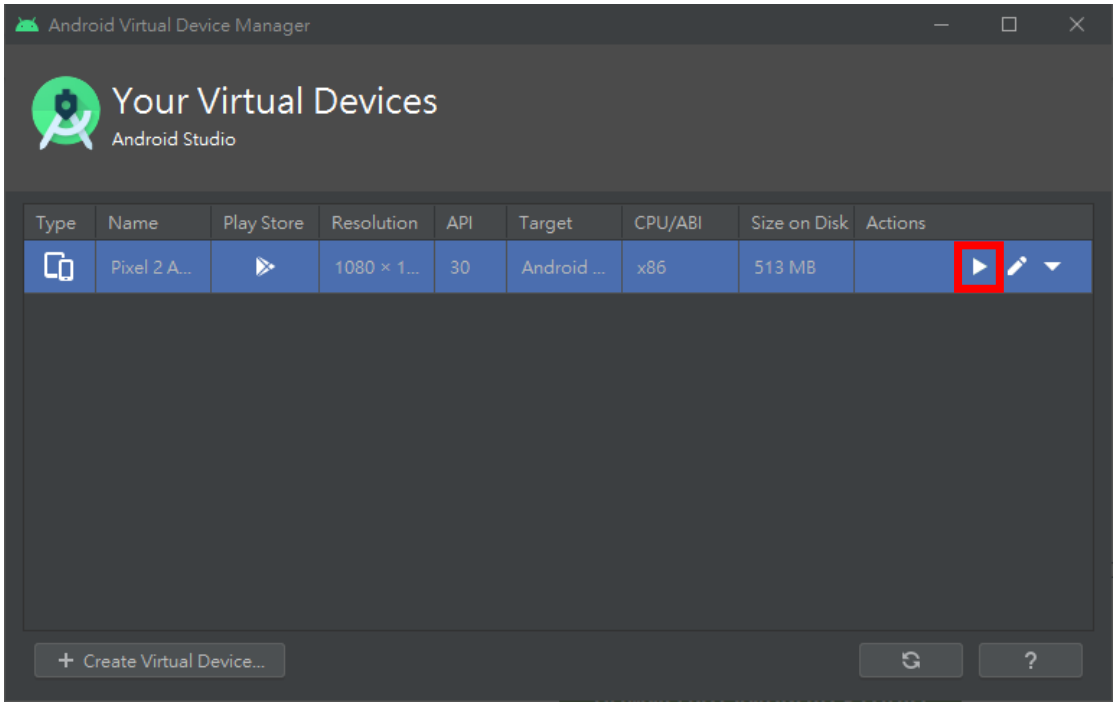


圖 1-20 模擬器列表

Step15 看到如圖 1-21 的畫面，就代表模擬器啟動完成。

圖 1-21 Android 模擬器畫面

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。

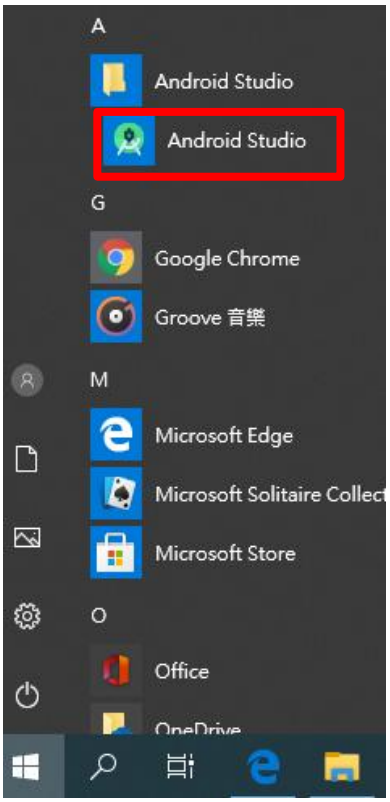


圖 1-22 開啟 Android Studio

Step2 建立新專案。

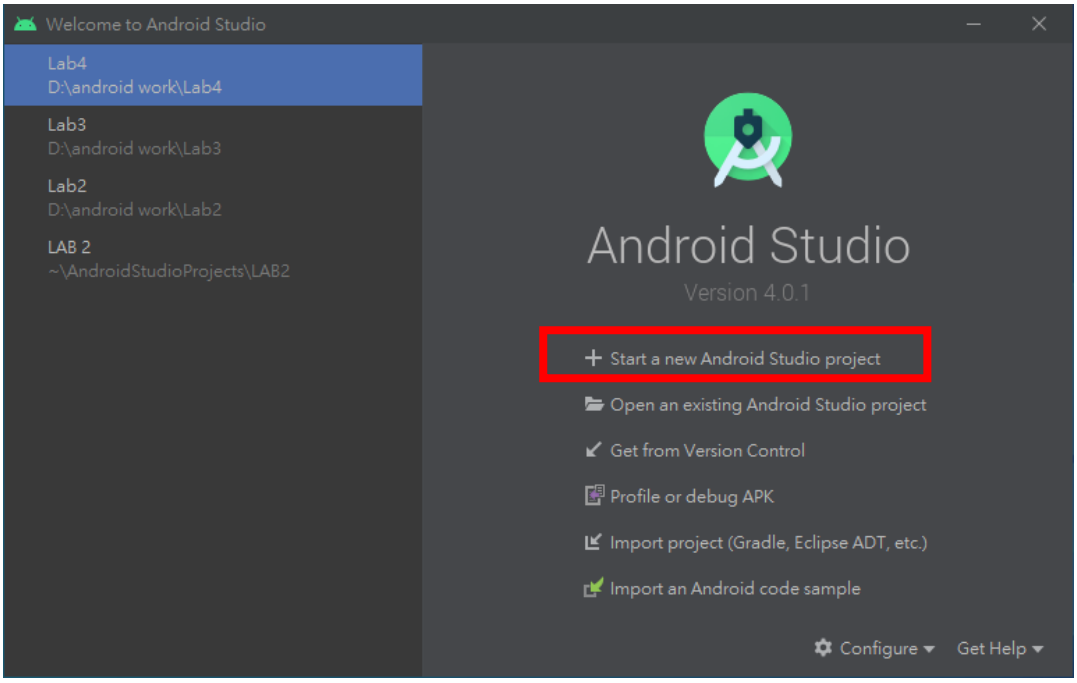


圖 1-23 建立專案

Step3 選擇 Empty Activity，下一步。

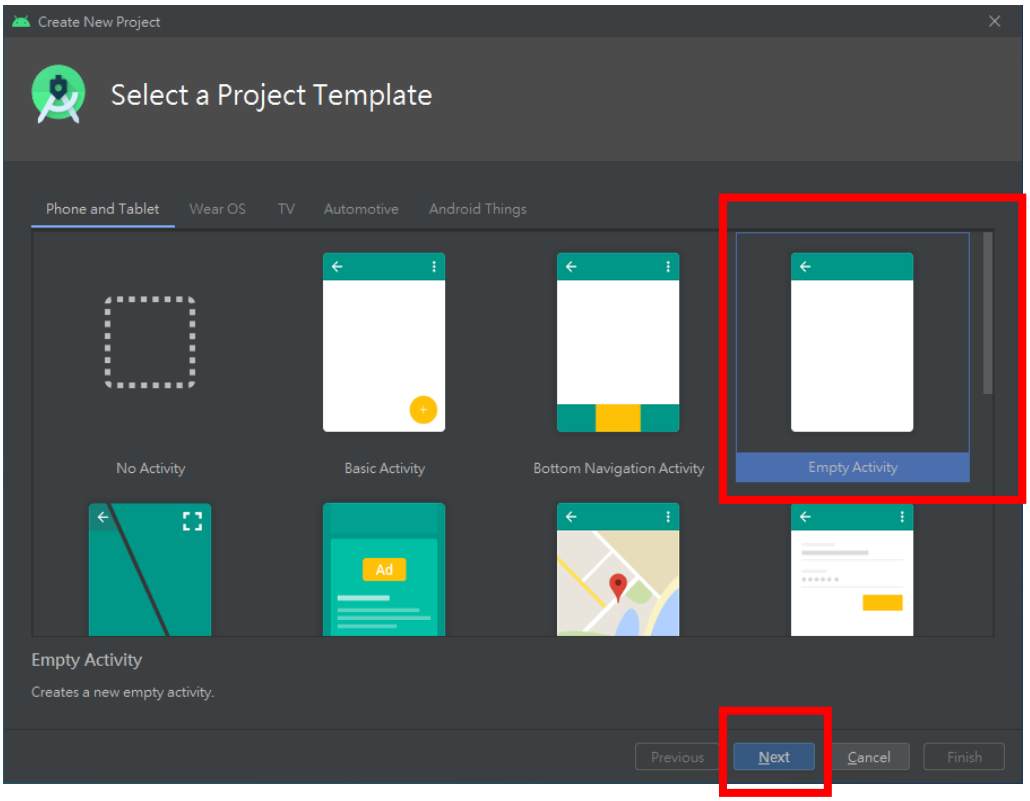


圖 1-24　選擇開發模板

Step4 輸入專案名稱，完成。

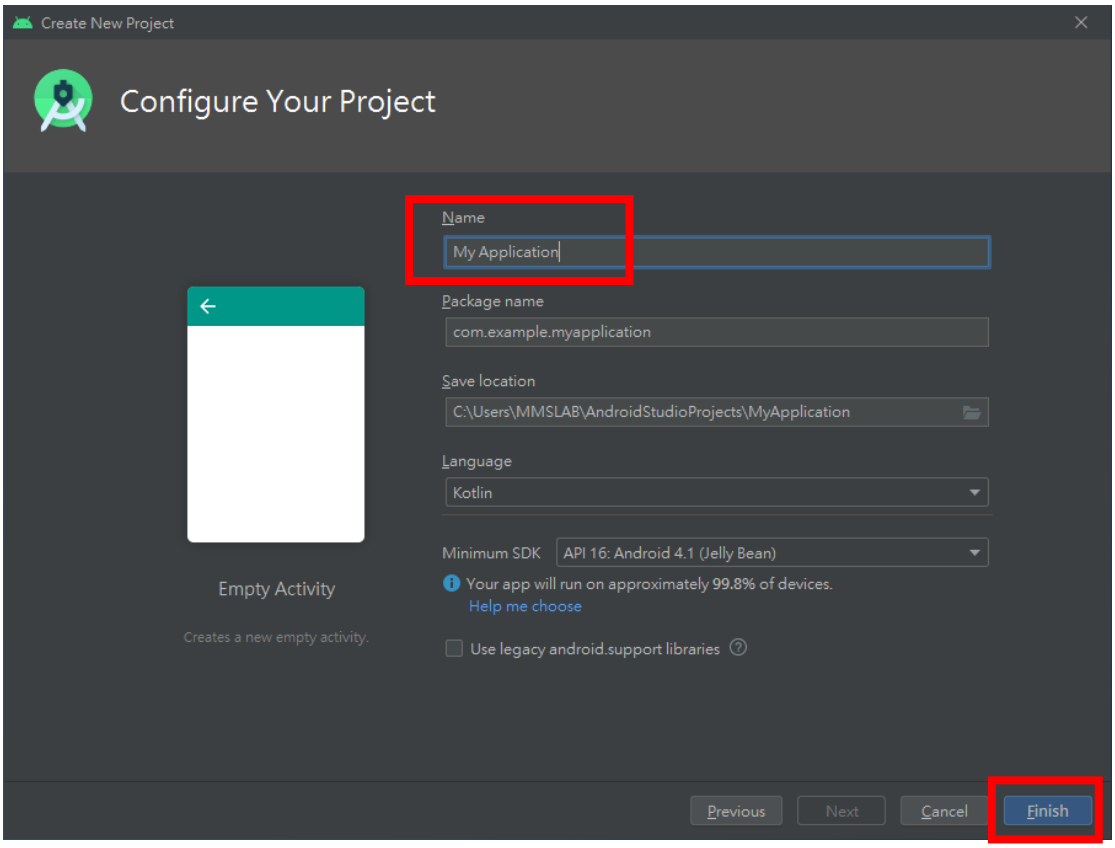


圖 1-25　建立完成

Step5 等待編譯完成。

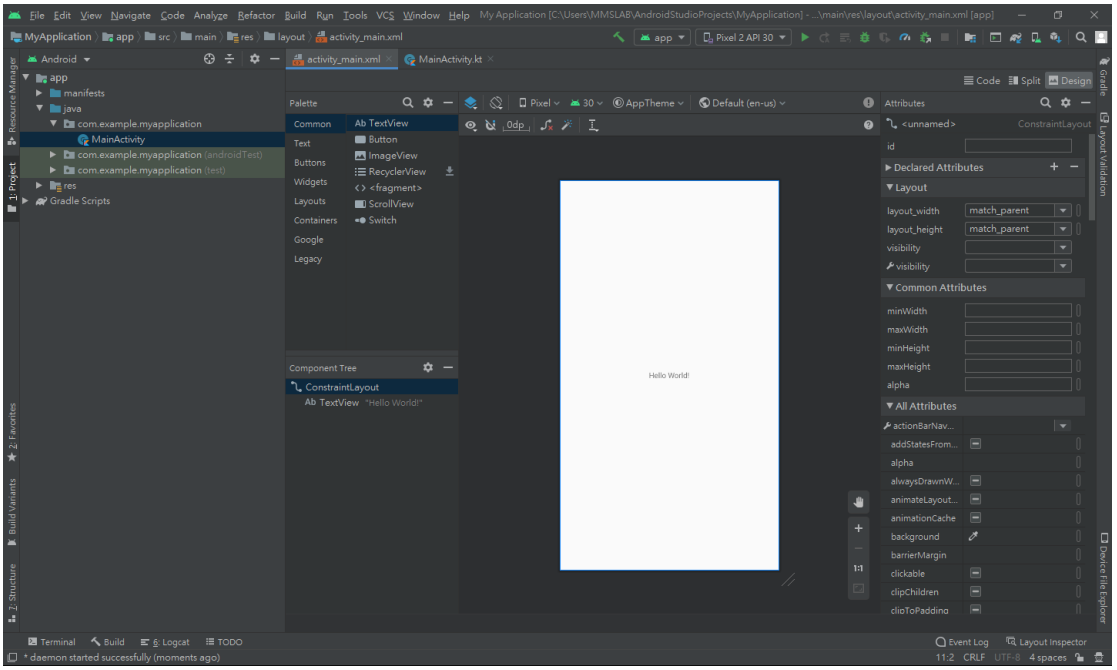


圖 1-26　建置完成

1.1.4 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 1-27 所示。

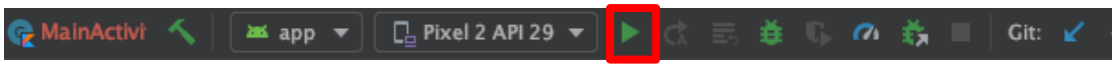


圖 1-27 編譯你的程式

Step2 安裝完成，預設的 Android 專案會顯示「Hello World!」，如圖 1-28 所示。

圖 1-28 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 1-29 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。

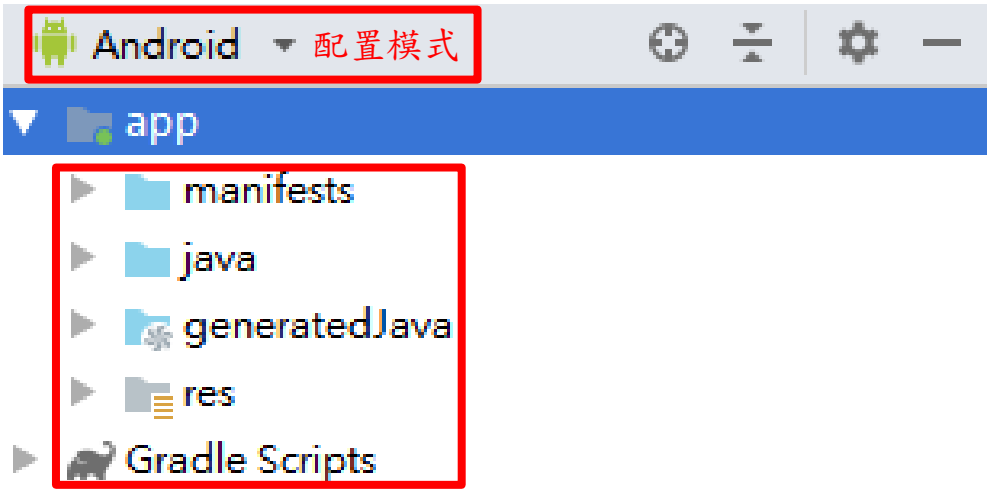


圖 1-29Android 專案配置目錄

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

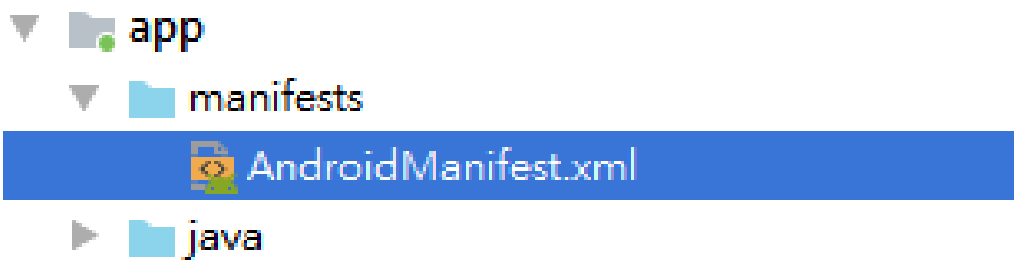


圖 1-30 應用程式設定檔 AndroidManifest

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 1-30 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"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.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"
        tools:ignore="AllowBackup,GoogleAppIndexingWarning,RtlEnabled"
        tools:targetApi="donut">
        <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 1-31 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 1-31 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:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme"
    tools:ignore="AllowBackup,GoogleAppIndexingWarning,RtlEnabled"
    tools:targetApi="donut">
    <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"
        tools:ignore="WrongManifestParent" />

    <reciver
        android:name=".MyReciver"
        android:enabled="true"
        android:exported="true"/>
</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 1-32「java」目錄。

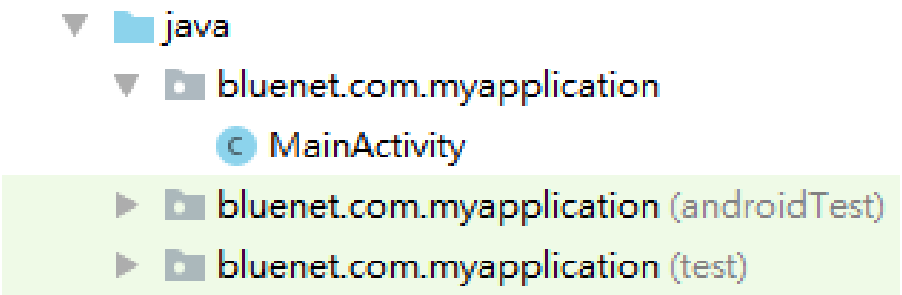


圖 1-32 專案的 Java 目錄

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 androidx.appcompat.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 1-33「res」目錄下。

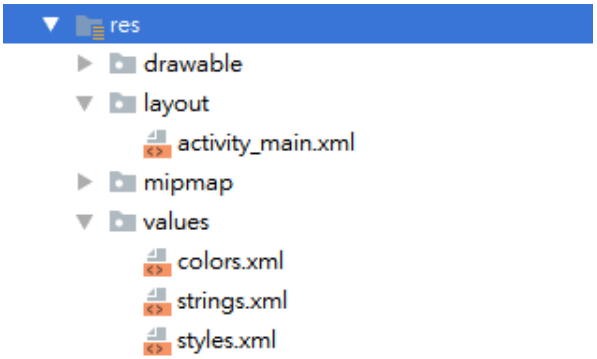


圖 1-33 專案的 res 資源目錄

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 1-34 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。

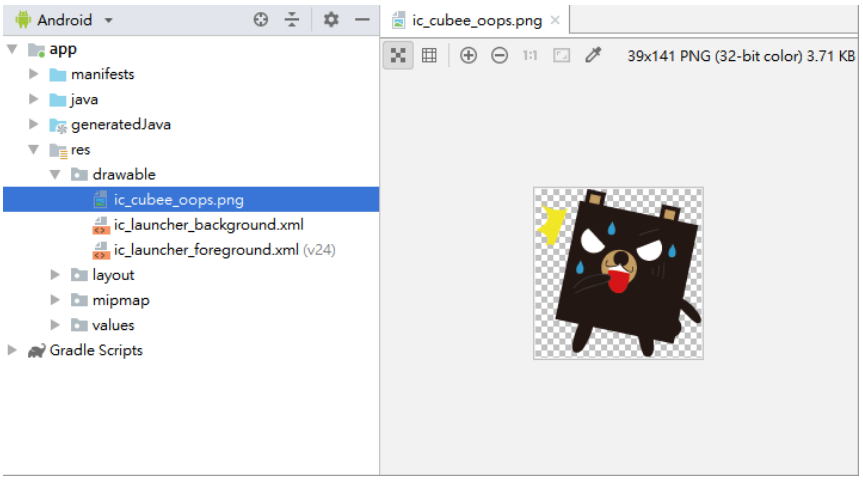


圖 1-34 圖檔 drawable

- 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。

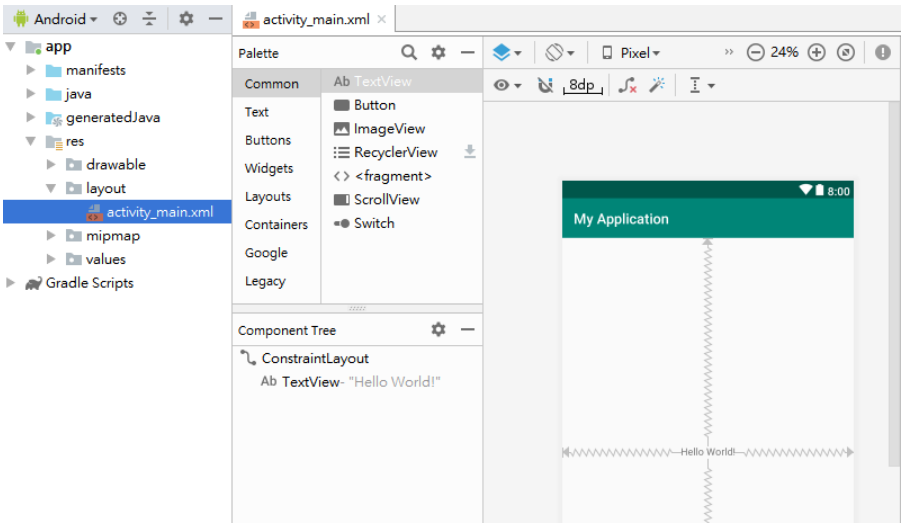


圖 1-35 布局 layout

- value：應用程式中可能使用的變數值，可放於此圖 1-36 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。

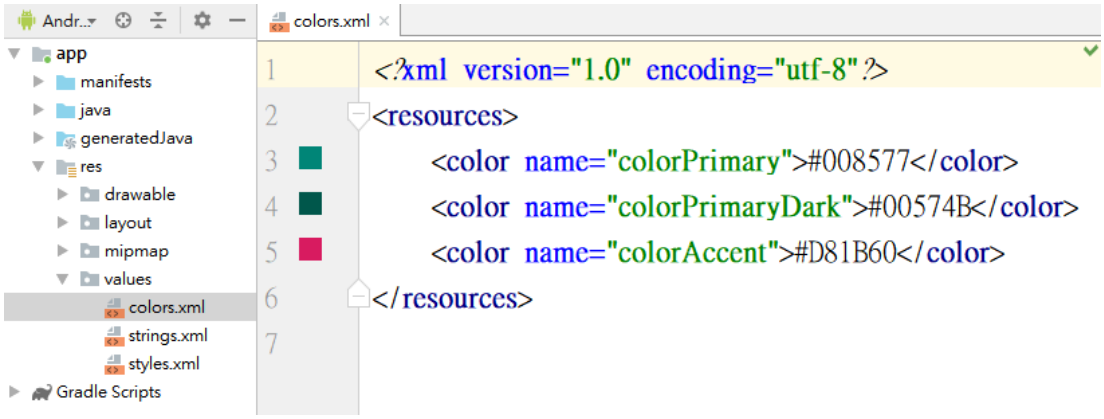


圖 1-36 數值 Values

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。

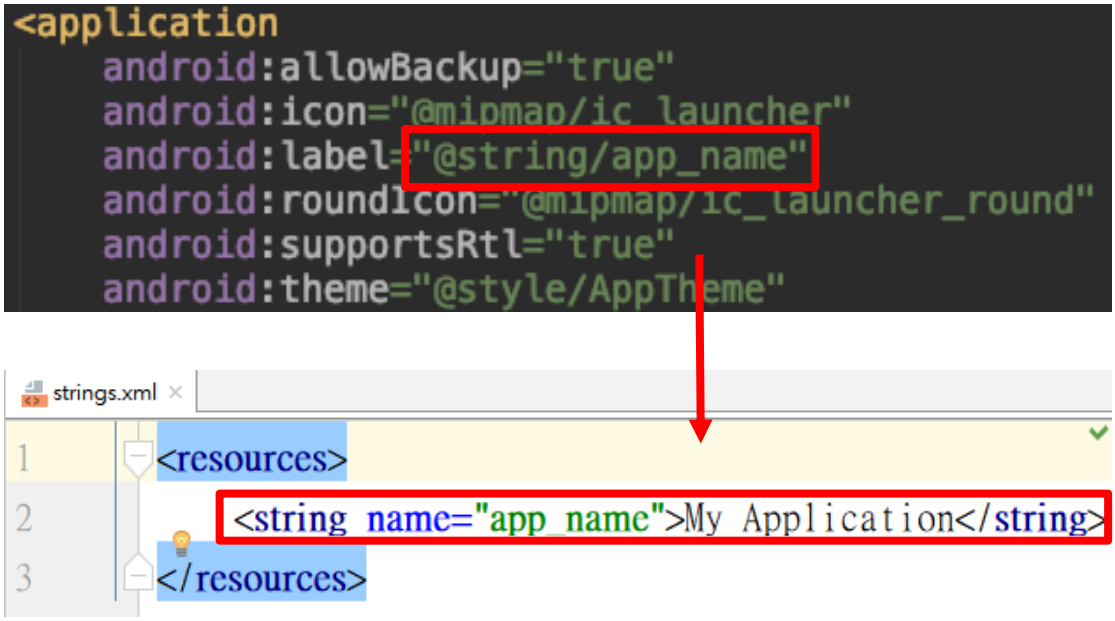


圖 1-37 字串資源 Strings

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。

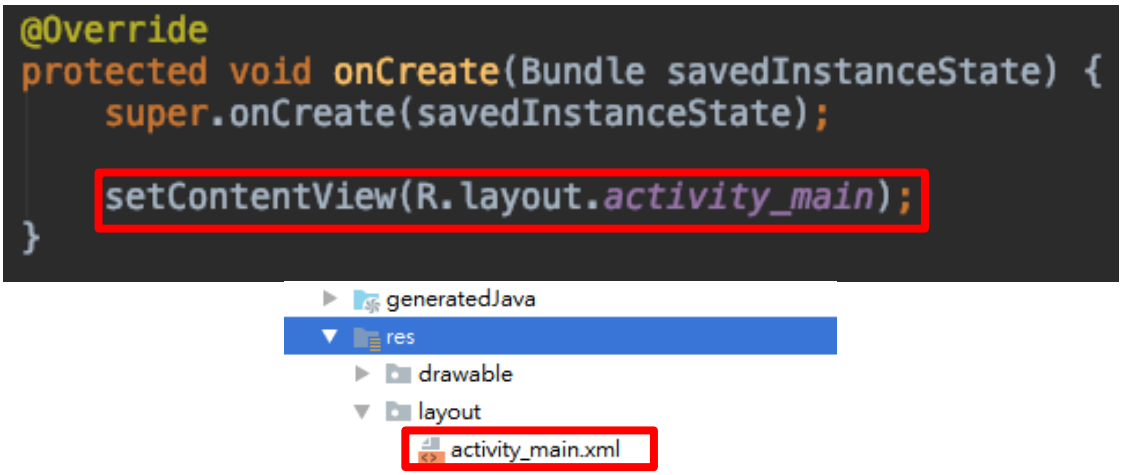


圖 1-38 透過 R 類別對照版面配置文件

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 1-39 所示，其中包含模組（Module）的設定檔，混淆碼規則與本地設定檔。

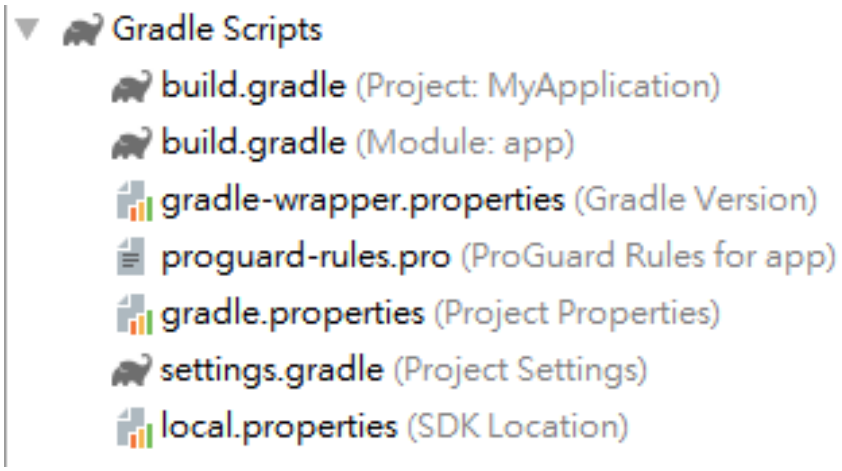


圖 1-39 自動化建置工具 Gradle

- 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。

- 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
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
distributionUrl=https\://services.gradle.org/distributions/gradle-5.4.1-all.zip
```

- proguard-rule.pro：程式混淆規則配置文件。保護好自己的程式碼是 APP 發佈上架的過程中很重要的一環，透過 Proguard 對類別、屬性和方法進行重新命名，增加有心人士反編譯後閱讀程式碼的難度，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。