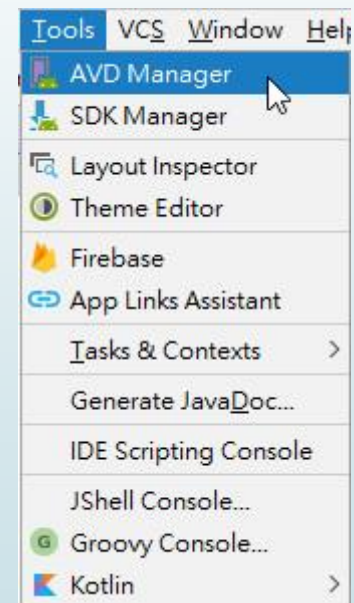


Android 開發環境

➤ Android Studio

➤ 模擬器設定：

➤ 功能表>Tools>AVD Manager→

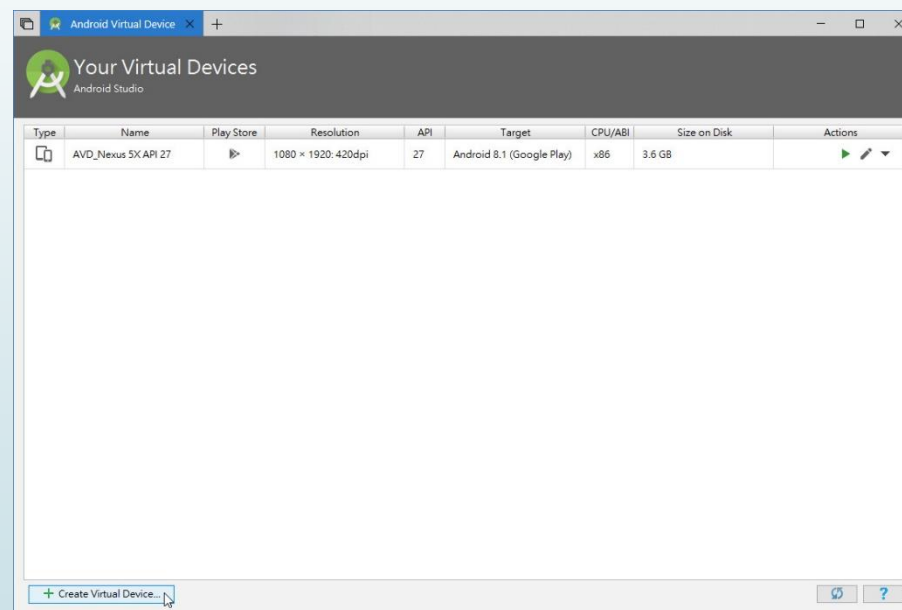


Android 開發環境

Android Studio

模擬器設定：

- 進入模擬器設定視窗 (Android Virtual Device)。
- 顯示目前已設定的模擬器列表 (Your Virtual Device)。
- 按下『Create Virtual Device...』鈕。

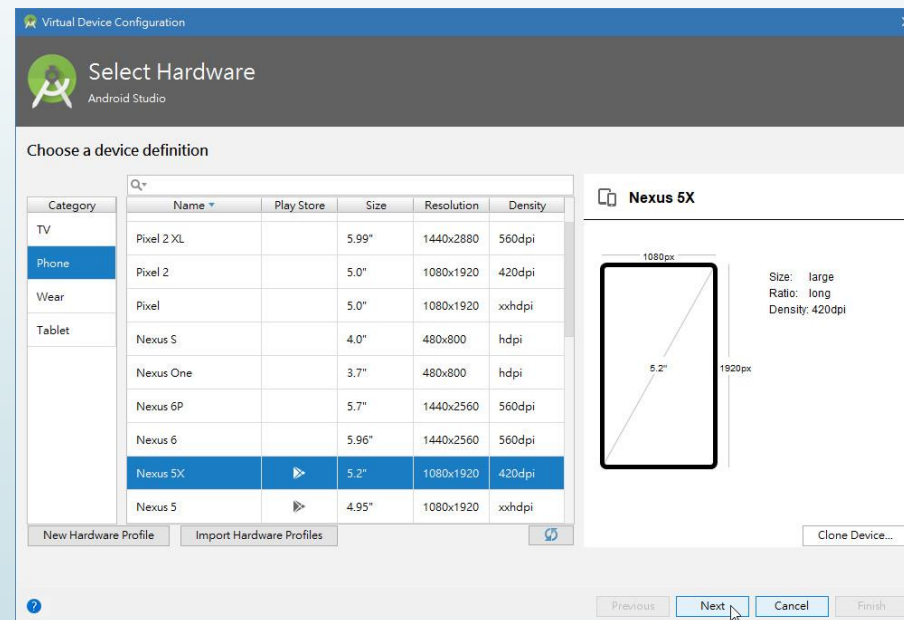


Android 開發環境

➤ Android Studio

➤ 模擬器設定：

- 選擇硬體 (Select Hardware)。
- 點選手機或平板的廠牌與尺寸→按下『Next』。

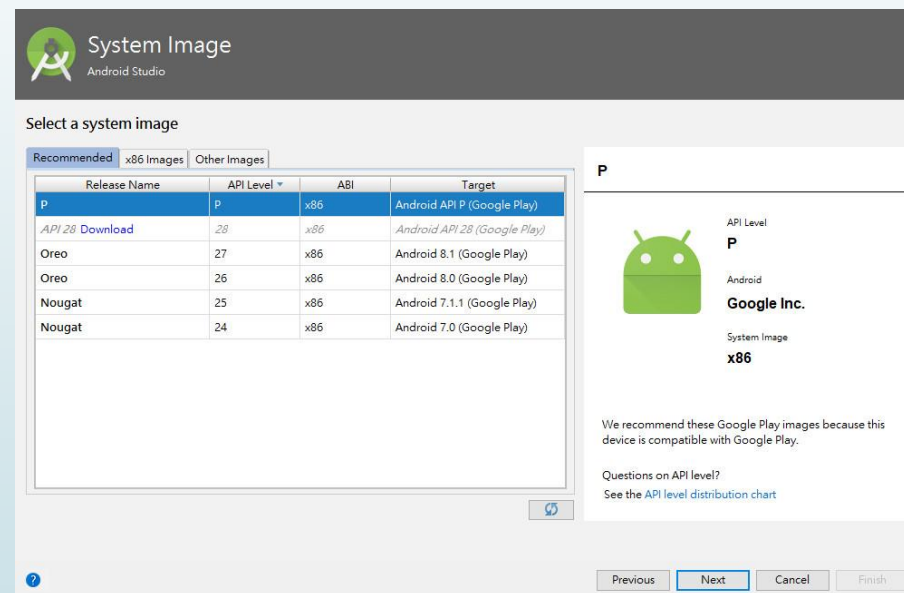


Android 開發環境

➤ Android Studio

➤ 模擬器設定：

- 選擇系統映射 (System Image)。
- 點選手機或平板要對應的 API 版本及 CPU→按下『Next』。
- 如果項目尚未安裝，按下『Download』。

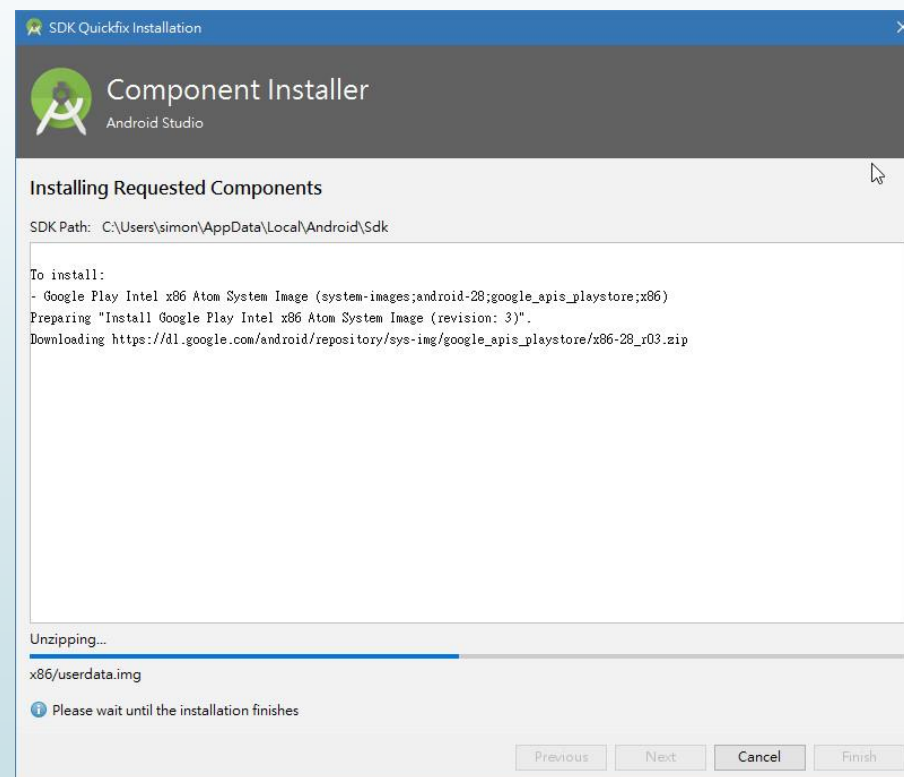


Android 開發環境

Android Studio

模擬器設定：

- 如果項目尚未安裝，按下『Download』進行系統安裝更新。

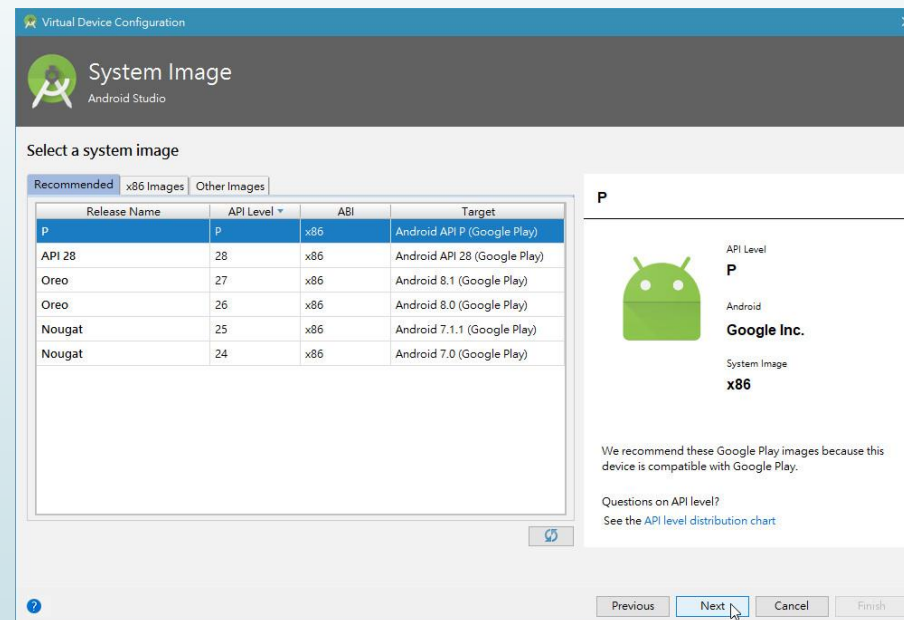


Android 開發環境

➤ Android Studio

➤ 模擬器設定：

➤ 系統安裝更新後→按下『Next』。

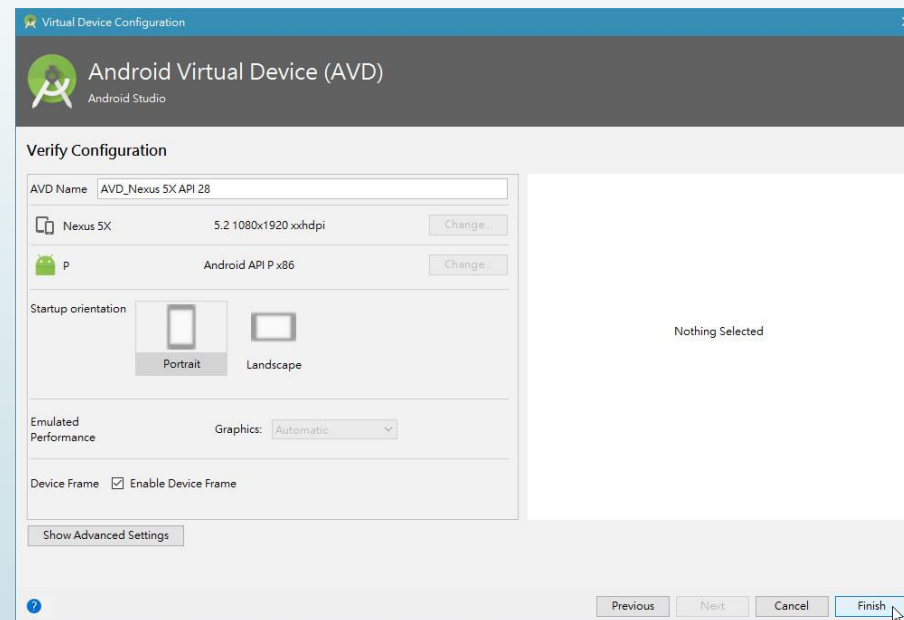


Android 開發環境

➤ Android Studio

➤ 模擬器設定：

- 確認設定的所有項目，並未設定的模擬器命名。
- 一切無誤後→按下『Finish』。

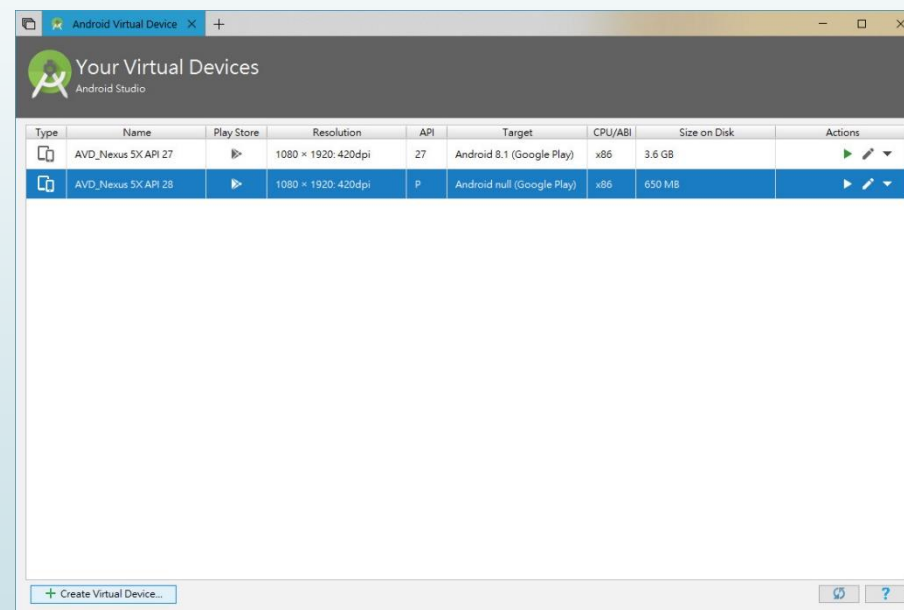


Android 開發環境

➤ Android Studio

➤ 模擬器設定：

- 模擬器設定完成後，顯示於模擬器選單中。

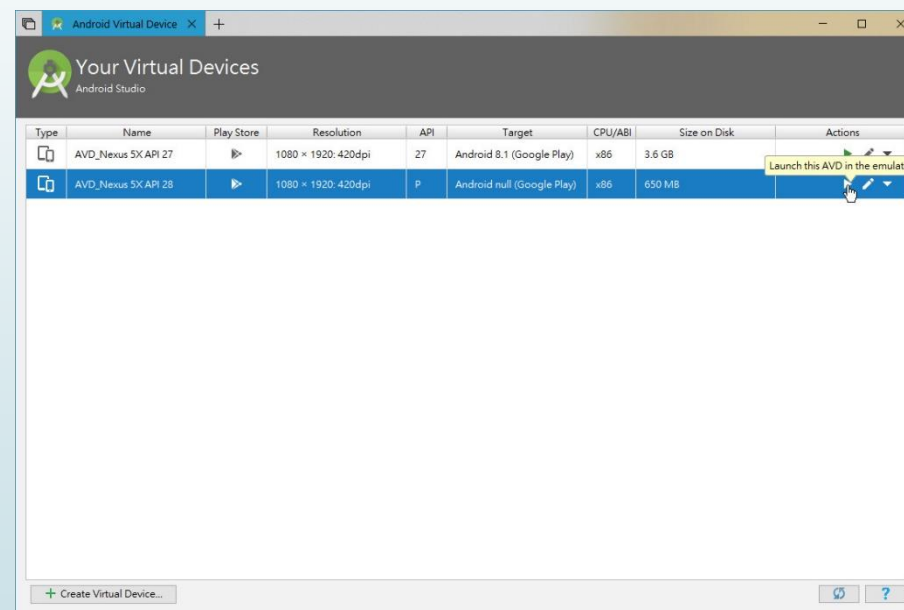


Android 開發環境

Android Studio

模擬器設定：

- 按下模擬器項目右方的『執行』鈕可執行開啟模擬器。

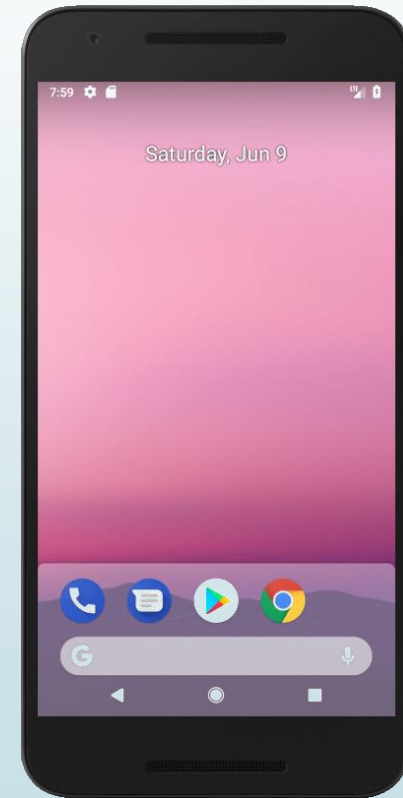


Android 開發環境

- Android Studio

- 模擬器設定：

- 開啟模擬器。



建立 Android App

■ 建立 Android App

■ UI 設計：

- 設計 App 在手機上顯示的畫面，即使用者介面 (User Interface ; UI)，在畫面中配置所需的元件。

■ 撰寫程式：

- 撰寫 Java 程式碼以控制元件的操作與行為。

■ UI 設計

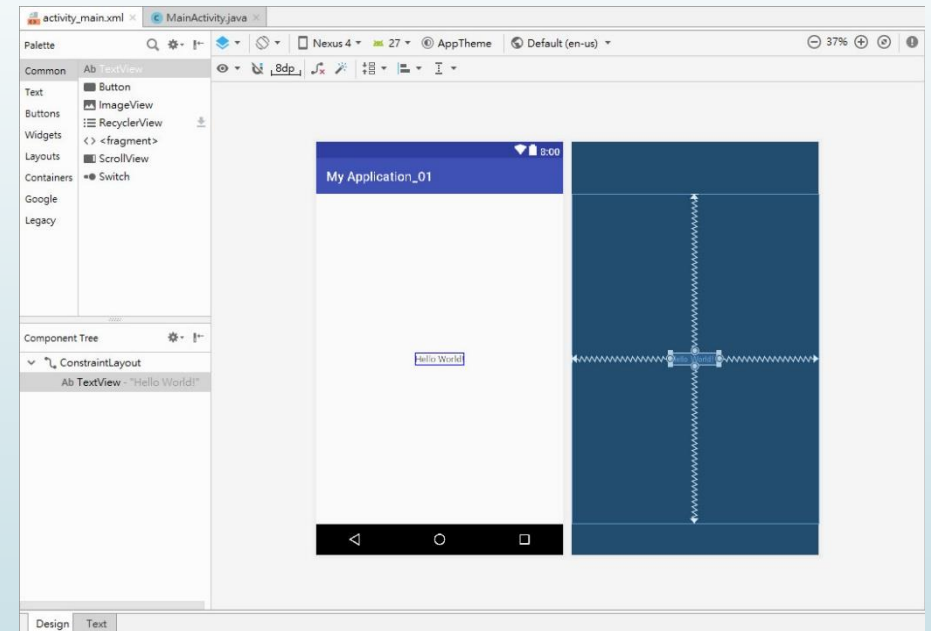
- 以『activity_mail.xml』檔案為主。

- 開啟新專案會自動建立一個預設的『activity_mail.xml』。(layout 資料夾中)

- 新專案預設內含一個『TextView』元件 (顯示文字)，並顯示『Hello World ! 』文字。

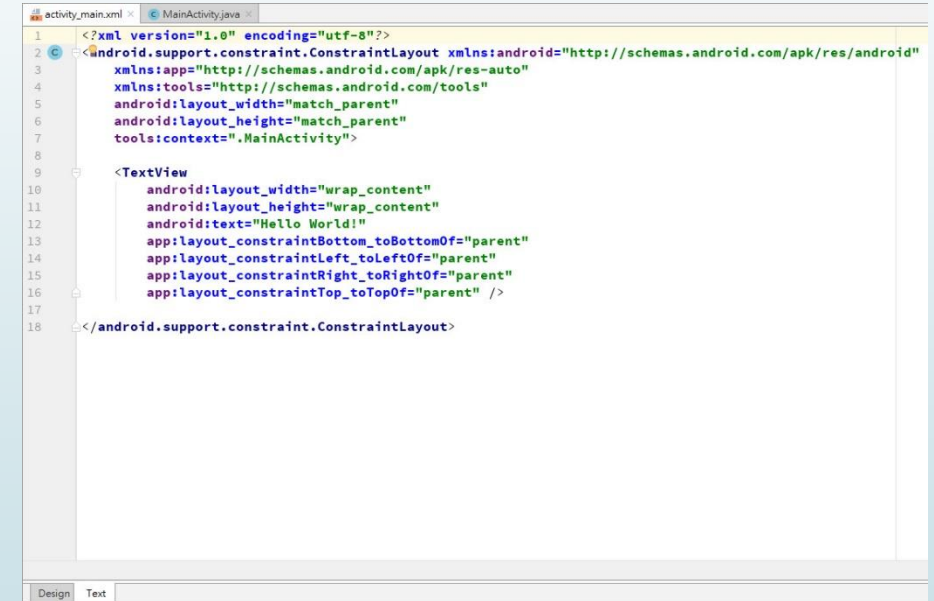
建立 Android App

- ▶ activity_main.xml
 - ▶ 有兩種操作介面：
 - ▶ Design：圖形介面，配置 App 要顯示的元件。



建立 Android App

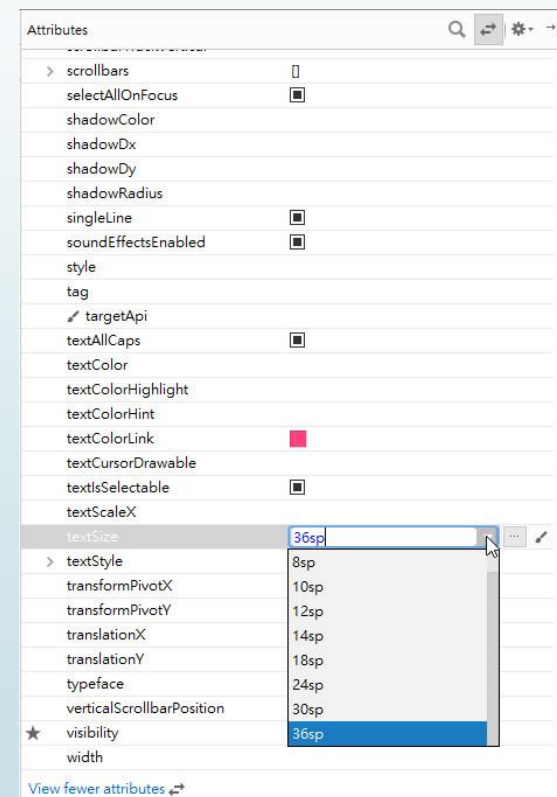
- ▶ activity_mail.xml
 - ▶ 有兩種操作介面：
 - ▶ Text：文字介面，撰寫元件的程式碼。



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     tools:context=".MainActivity">
8
9     <TextView
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:text="Hello World!"
13         app:layout_constraintBottom_toBottomOf="parent"
14         app:layout_constraintLeft_toLeftOf="parent"
15         app:layout_constraintRight_toRightOf="parent"
16         app:layout_constraintTop_toTopOf="parent" />
17
18 </android.support.constraint.ConstraintLayout>
```

建立 Android App

- ▶ activity_mail.xml
 - ▶ 編輯『TextView』：
 - ▶ 變更文字大小：
 - ▶ 切換至『Design』模式。
 - ▶ 元件『屬性面板』(右側)→設定『textSize』屬性。



建立 Android App

► activity_main.xml

► 編輯『TextView』：

► 變更文字內容：

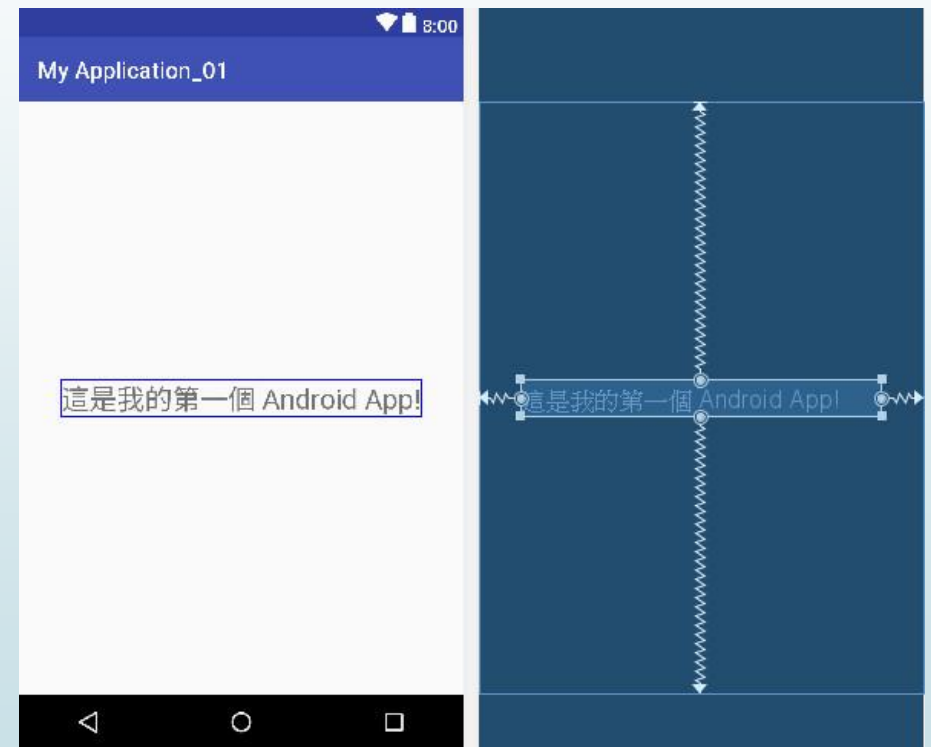
- 切換至『Text』模式。
- 於 TextView 元件標籤區域內更改『text』屬性內容，輸入要顯示的文字。



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.constraint.ConstraintLayout xmlns:andro
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     tools:context=".MainActivity">
8
9     <TextView
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:text="這是我的第一個 Android App!"
13         android:textSize="36sp"
14         app:layout_constraintBottom_toBottomOf="parent"
15         app:layout_constraintLeft_toLeftOf="parent"
16         app:layout_constraintRight_toRightOf="parent"
17         app:layout_constraintTop_toTopOf="parent" />
18
19 </android.support.constraint.ConstraintLayout>
```

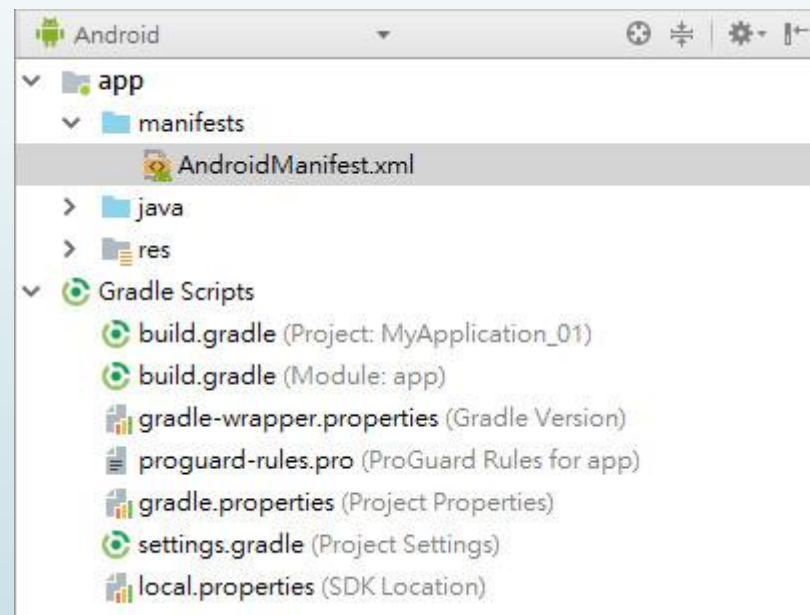
建立 Android App

- ▶ activity_mail.xml
 - ▶ 編輯『TextView』：
 - ▶ 變更文字內容：
 - ▶ 切換至『Design』模式。
 - ▶ 看到修改後的結果。



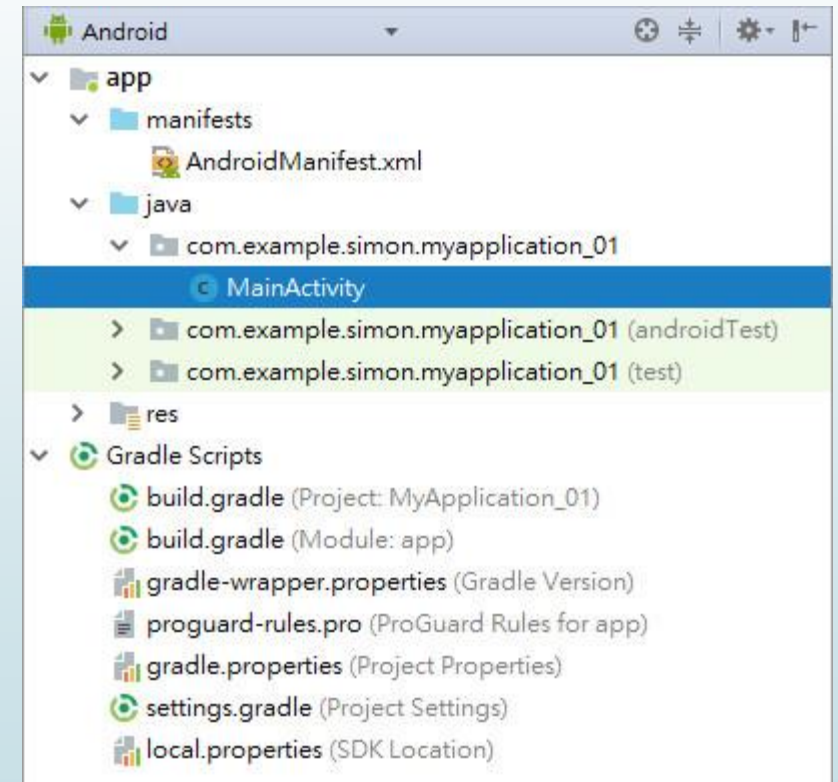
Android App 專案解析

- Android App 專案的組成架構
 - 『manifests』資料夾：
 - 存放『AndroidManifest.xml』檔案。(清單檔)
 - 宣告 App 中所有的元件與所有 App 的需求。



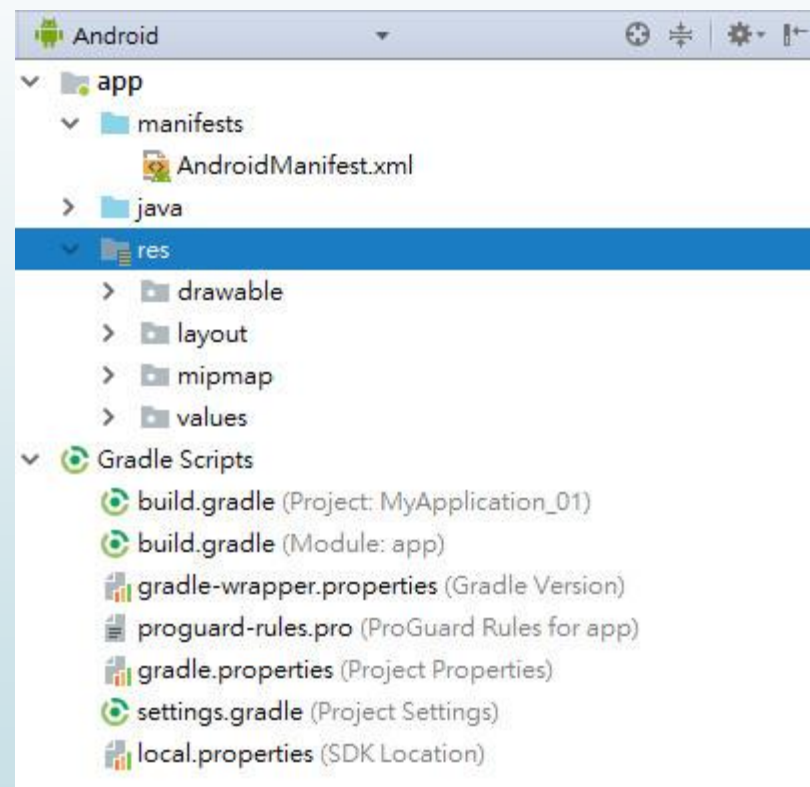
Android App 專案解析

- Android App 專案的組成架構
 - 『java』資料夾：
 - 存放 App 專案中的主程式 Java 檔。
 - 主要啟動 App 的是『MainActivity.java』檔。



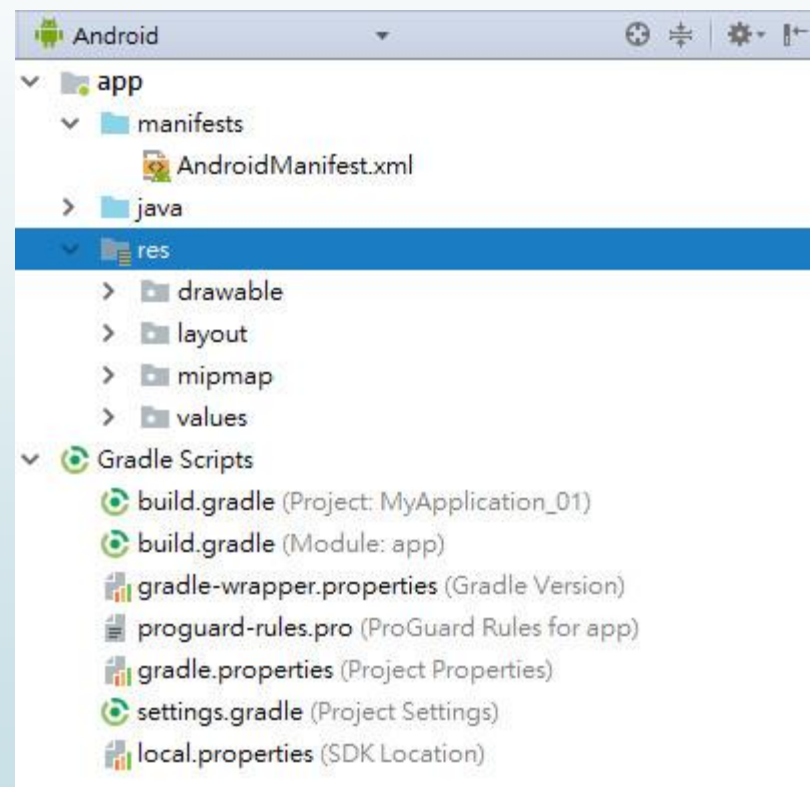
Android App 專案解析

- Android App 專案的組成架構
 - 『res』資料夾：
 - 存放 App 專案中的各項資源檔。
 - 『drawable』：
 - 存放 App 專案中所使用的圖檔與繪圖檔。
 - 『layout』：
 - 存放 App 專案中輸出的版面配置檔案，其中『activity_main.xml』為 App 專案主要的版面配置檔。



Android App 專案解析

- Android App 專案的組成架構
 - 『res』資料夾：
 - 存放 App 專案中的各項資源檔。
 - 『mipmap』：
 - 存放 App 專案中所使用的應用程式圖檔 (即 icon 檔)。
 - 『values』：
 - 存放 App 專案中的所使用的顏色、字串、樣式檔案，格式為 xml。



Android App 專案解析

- Android App 專案的組成架構
 - 『res』資料夾：
 - 存放 App 專案中的各項資源檔。
 - 『mipmap』：
 - 存放應用程式圖檔的建議。

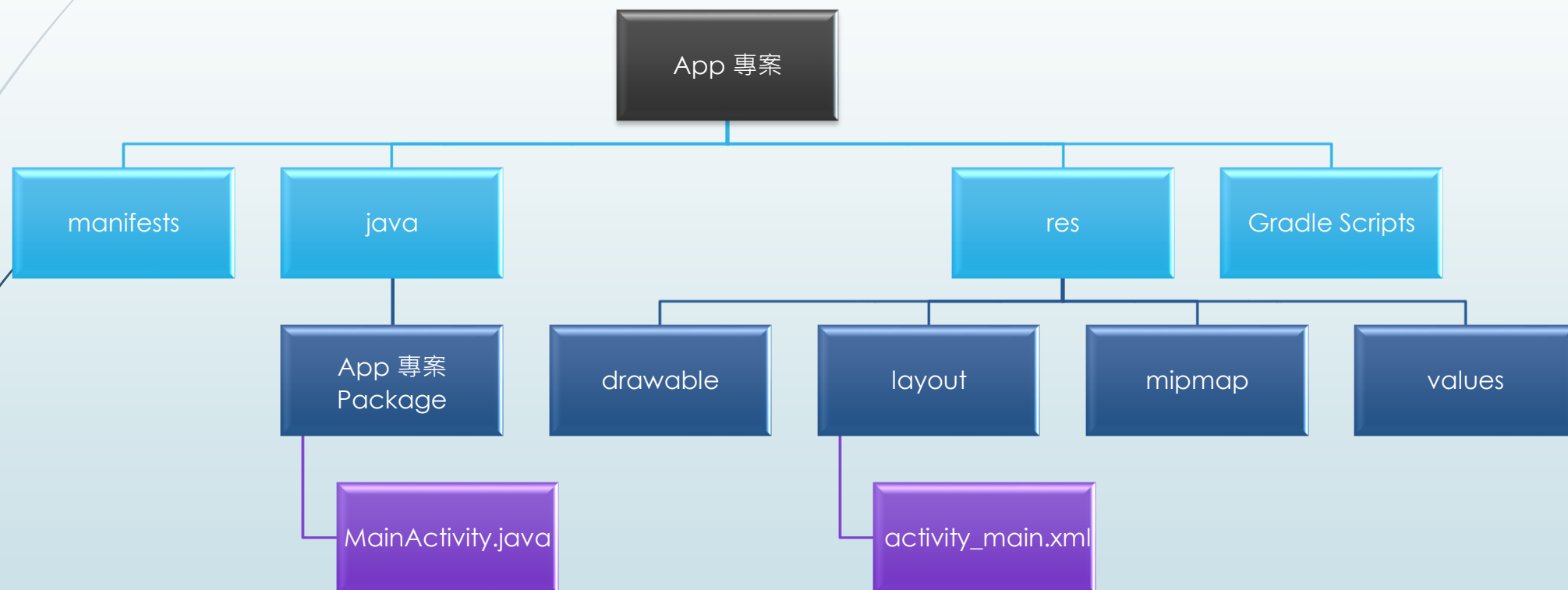
Icon 格式	dpi	建議尺寸
mdpi	120~160	48*48
hdpi	160~240	72*72
xhdpi	240~320	96*96
xxhdpi	320~480	144*144
xxxhdpi	480~640	192*192
anydpi	~	自適應任意大小

Android App 專案解析

► Android App 專案的資源檔

- Android 專案的每一個資源，在系統編譯專案時，SDK 建立工具會為每一資源指定一個整數 ID，並把資源彙整在『R.java』檔中，讓 App 或其他資源可以利用此 ID 存取特定的資源。
- 如 App 中含有一個影像圖檔名為『graph.png』(放在 /res/drawable/ 資料夾中)，則 SDK 工具會建立一個資源 ID，名稱為『R.drawable.graph』，就可以將此 ID 參考到該影像或是將該影像顯示到螢幕。

Android App 專案解析



Android App 專案解析

- Android 專案程式碼說明：MainActivity.java
 - package com.example.simon.myapplication_01 :
 - 指定 App 的套件名稱。
 - 套件名稱即對應到檔案系統的資料夾路徑。
 - 一個套件中可以包含多個程式檔。
 - import :
 - 要求編譯器載入指定的類別或是套件以供 App 使用。
 - 建立專案時都會自動套用『android-support-v7-appcompat』程式庫。主程式繼承的『AppCompatActivity』類別，即屬於此程式庫。
 - 使用程式庫的目的是為了讓 Android 系統的新功能可以在舊版的 Android 平台上使用。

Android App 專案解析

► Android 專案程式碼說明：MainActivity.java

► `public class MainActivity extends AppCompatActivity :`

- 定義程式的主類別，即專案的主類別，也是 App 程式開始執行之處。
- 是 App 主要的 Activity，為一個 Java 類別宣告，名稱為『MainActivity』。
- MainActivity 子類別是繼承自 Activity 類別或 AppCompatActivity 類別。

Android App 專案解析

➤ Android 專案程式碼說明：MainActivity.java

➤ onCreate 方法：

- 類別內含一個『onCreate』方法，MainActivity 子類別開始時必須改寫 (@override) 此方法，並傳入一個參數『savedInstanceState』。
- 是一個狀態處理方法，因在 MainActivity 物件執行過程中，會有不同的狀態轉換，每一個狀態都有對應的狀態處理方法。
- 當 MainActivity 物件產生時 (即程式啟動時)，會被 Android 系統呼叫執行的方法。
- super.onCreate(savedInstanceState)：
 - 呼叫父類別的『onCreate』方法，並將傳入參數『savedInstanceState』。
 - 『savedInstanceState』參數為 Android 系統傳入的一個『Bundle』物件，記錄一些專案中要傳送的資料，資料是以 key-values 方法存取，供『onCreate』方法處理。

Android App 專案解析

► Android 專案程式碼說明：MainActivity.java

► onCreate 方法：

► savedInstanceState 參數值：

- App 被使用者啟動，則其值為『null』。
- App 被 Android 系統強迫結束，則 Android 系統會利用 onSaveInstanceState() 狀態處理方法將 App 的狀態儲存，接著在 onCreate() 方法中，利用 savedInstanceState 引數傳入 App 之前的執行狀態。

► setContentView(R.layout.activity_main)：

- 以『setContentView』方法將定義在『activity_main.xml』主介面佈局檔案之中的畫面配置顯示在設備的螢幕上。

Android App 專案解析

■ manifests 資料夾

■ 儲存『AndroidManifests.xml』檔案

■ 『AndroidManifests.xml』檔案：

- 記錄程式專案的架構，以及執行時必須取得那些系統功能，如網路、SD卡、相機...等。
- 宣告了 App 的所有活動、所有 App 的需求、所需的設備組態、非程式碼的 App 資源 (圖像、字串...)...
- 因有各種搭載 Android 的設備，這些設備並不提供相同的特性與功能，為避免 App 無法安裝在該設備中，所以必須清楚的在『AndroidManifests.xml』檔案中宣告 App 所需的軟硬體需求。
- 這些需求對系統來說只是一個聲明，並不會去讀取它，但是『Google Play』會讀取它，以便判斷該 App 是否適合下載到該設備中。

Android App 專案解析

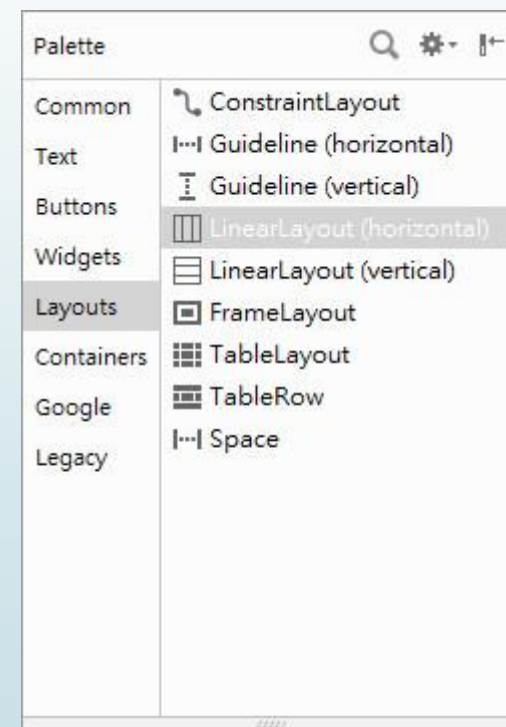
► manifests 資料夾

► 『AndroidManifests.xml』檔案所包含的資訊如下：

- 宣告 App 的套件 (package) 名稱，該名稱是 App 的唯一名稱。
- 宣告 App 包含哪些元件與實作元件的類別：
 - 活動 (activity)、服務 (services)、廣播接收器 (broadcast receiver)、內容提供器 (content provider)。
- 發佈元件所具備的功能。
- 指出哪一個元件是 App 的出入口。
- 指示要執行此 App，使用者必須獲得甚麼授權。
- 宣告要執行此 App 所需的軟硬體 (如相機、藍芽服務、多觸點螢幕)。
- 宣告 App 可以被執行的最低 API 層級需求。
- App 所需的其他 API 函數庫。

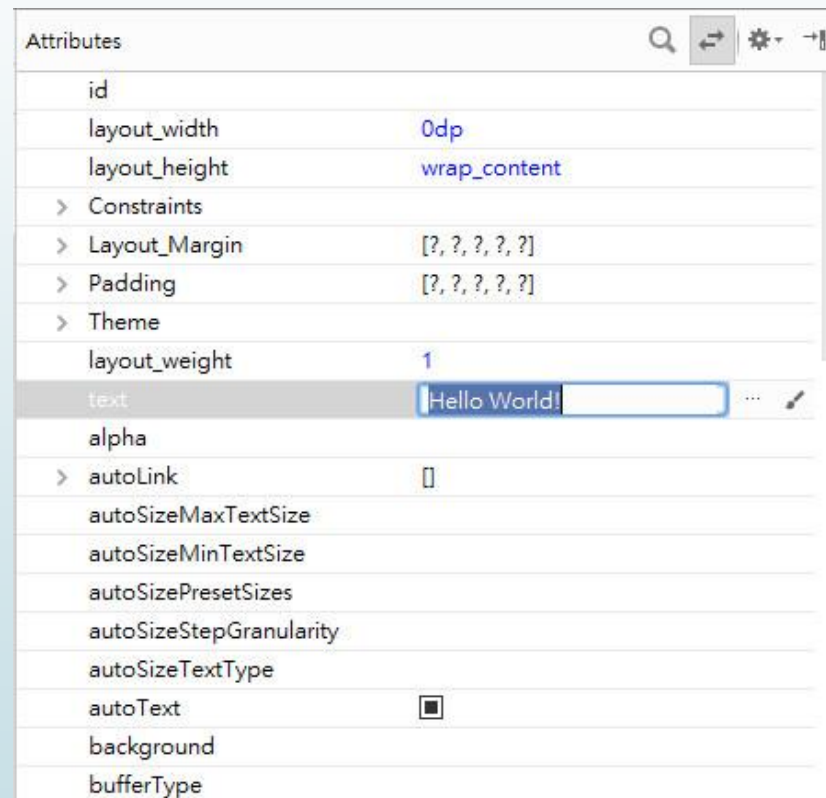
Android App 設計

- 視覺設計：
 - activity_main.xml；介面佈局檔。
 - 即使用者介面；User Interface (UI)，以 XML 程式撰寫。
 - 使用『Design』模式，於『Palette』(面板區) 中將 UI 元件拖曳至設計預覽區中。



Android App 設計

- ➡ 視覺設計：
 - ➡ 點選元件，於右側屬性區設定該元件相關屬性。



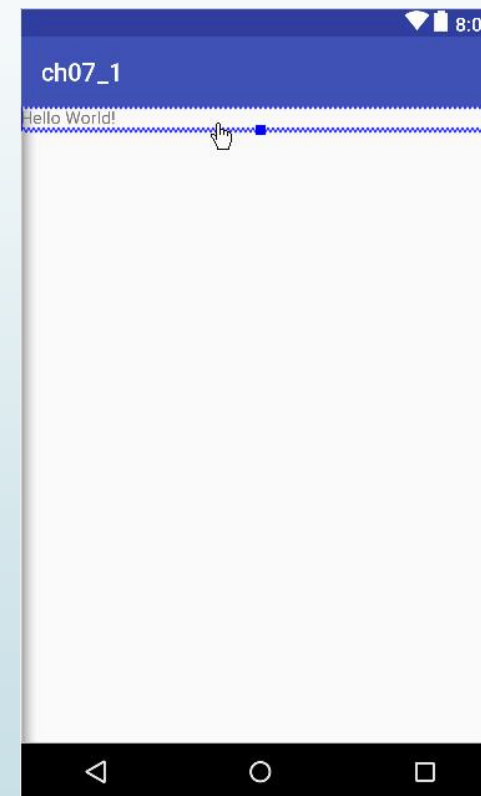
Android App 設計

- ➡ 視覺設計：
 - ➡ 使用『Text』模式，撰寫 xml 程式。

```
activity_main.xml x MainActivity.java x
1  <?xml version="1.0" encoding="utf-8"?>
2  <android.support.constraint.ConstraintLayout xmlns
3      xmlns:app="http://schemas.android.com/apk/res-
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".MainActivity">
8
9      <LinearLayout
10         android:layout_width="match_parent"
11         android:layout_height="match_parent"
12         android:orientation="horizontal">
13
14         <TextView
15             android:layout_width="0dp"
16             android:layout_height="wrap_content"
17             android:layout_weight="1"
18             android:text="Hello World!" />
19         </LinearLayout>
20
21 </android.support.constraint.ConstraintLayout>
```

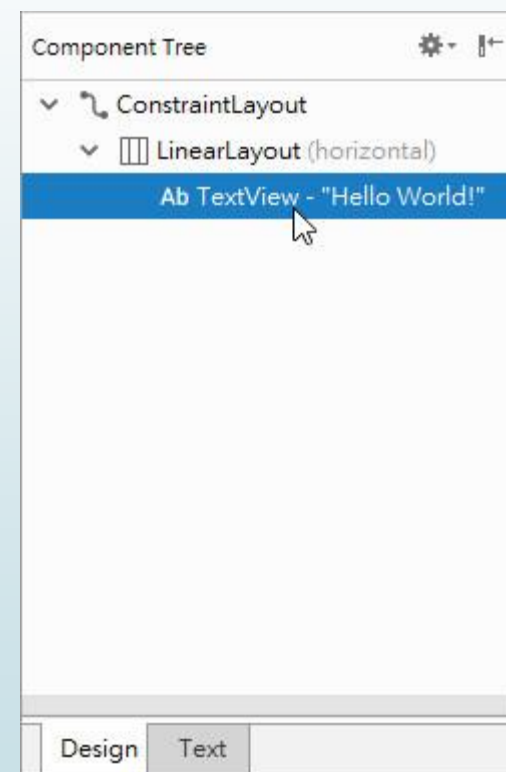

Android App 設計

- 視覺設計：
 - 選取元件的方式：
 - 『Design』模式：
 - 於設計預覽區中點選。



Android App 設計

- 視覺設計：
 - 選取元件的方式：
 - 『Design』模式：
 - 於『Component Tree』面板中點選。



Android App 設計

- 介面元件都是佈置在【res / layout】資料夾的版面配置檔案 (XML 檔) 中。
- 主要功用：顯示程式的版面。
- 新增元件：
 - Design 標籤>介面元件區→拖曳元件至設計預覽區。
 - Text 標籤→以『xml』語法新增元件。
 - 屬性視窗→設定元件屬性。

Android App 設計

■ 介面元件

■ 【android:id】屬性：

- 指定唯一的元件辨識代碼。

■ 【android:layout_width】屬性：

- 指定元件的寬度。
- 若設為『match_constraint』，則寬度為填滿整個容器寬度。
- 若設為『wrap_content』，則寬度為隨元件內容大小而定，可設定『width』屬性。

■ 【android:layout_height】屬性：

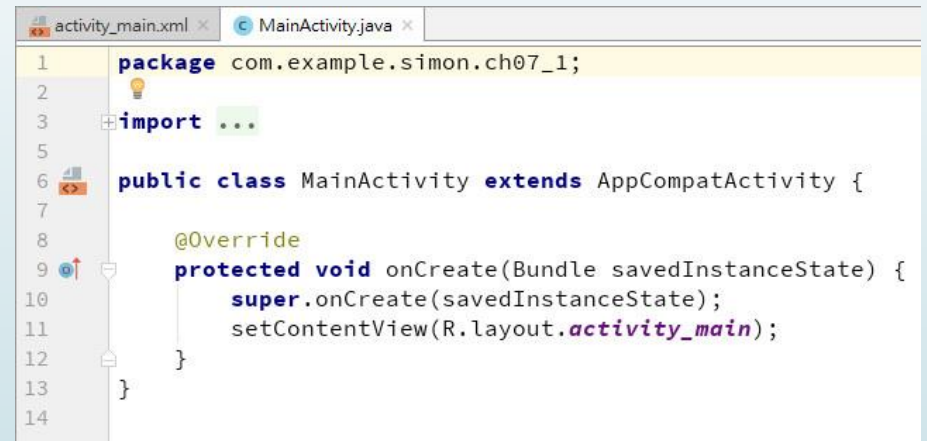
- 指定元件的高度。
- 若設為『match_constraint』，則高度為填滿整個容器高度。
- 若設為『wrap_content』，則高度為隨元件內容大小而定，可設定『height』屬性。

■ 【android:text】屬性：

- 指定元件的顯示文字。

Android App 設計

- 程式邏輯：
 - 即程式碼撰寫，以 JAVA 程式撰寫。
 - 設計主要的 Java 檔案 (MainActivity.java)。

A screenshot of an IDE window showing the MainActivity.java file. The code is as follows:

```
1 package com.example.simon.ch07_1;
2
3 import ...
4
5
6 public class MainActivity extends AppCompatActivity {
7
8     @Override
9     protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_main);
12     }
13 }
14
```

Android App 設計

➤ id 屬性

- 即元件的識別碼，在版面上所有的元件 id 不可以重複。
- 當元件設定 id 後，就會在『R.java』檔中產生對應的資源 ID。
- 語法：
 - **Android:id= "@+id/元件名稱"**
- Android 將所有可以放到圖形化佈局設計預覽區的元件都歸類在一個資源類別，即 id 類別。
- 所以 id 類別中的元件，其資源 ID 就是『R.id.資源名稱』。
 - 如 TextView 元件的 id 屬性命名為 tv，則可以使用『R.id.tv』來存取該 TextView 元件。