「Android 開發技術每隔幾年就徹底改變一次，這種書很難寫，Griffiths 幾乎完全改寫了這本經典的第三版，用很厲害的方法來展示現代的 Android app 究竟是怎麼開發的。總之，他們再次寫出這個領域最棒的書籍。如果你想要用正確的方式來建構 Android app，買這本書就對了！」

— Ken Kousen，Kousen IT 公司總裁

「寫一本準確易懂的書很難，但 Griffiths 伉儷做到了，他們用清楚、準確的方式講解複雜的 Android，即使是新手也能理解，給我留下非常深刻的印象。」

— G. Blake Meike，Android 開發者
暨《Programming Android 第二版》的共同作者

「Dawn 與 David 很用心地將 Java 範例修改成 Kotlin，把第三版變成優秀的現代 Android 開發學習資源。」

— Duncan McGregor 與 Nat Pryce，
《Java to Kotlin: A Refactoring Guidebook》的作者

「很感謝你們出版這本書，我是 Android 開發的新手，這本書很清楚地解釋了每一件事情。」

— Ambreen Khan，Android developer

「Android 開發已經不一樣了，而且差異很大，讓這本書成為你的好友，準確且熟練地引導你在正確的場合以正確的方式安心地學會游泳，在你享受許多樂趣的同時，避開電力食人魚、非同步毒蛇，以及性能水蛭出沒的水域。」

— Ingo Krotzky，Android 學員

在全新的開始

機器散發魅力

心中諸多疑問

本書一一指明

— Susan B. Brenner，機器詩人

《深入淺出 Kotlin》推薦文

「本書清楚、直覺、易懂，如果你是 Kotlin 新手，這是很棒的入門書籍。」

— Ken Kousen，Kotlin 官方訓練師，JetBrains 認證

「深入淺出 Kotlin 絕對可以幫助你快速掌握這個語言，為你打下深厚的基礎，並幫你找到寫程式的樂趣。」

— Ingo Krotzky，Kotlin 學員

「終於不需要為了學 Kotlin 而學 Java 了，我一直在等待這本簡明、有趣的書籍。」

— Matt Wenham 博士，資料科學家與 Python 程式設計師

「Kotlin 是炙手可熱的新興語言，在科技產業中，它的人氣還在不斷上昇。《深入淺出 Kotlin》用有趣、平易近人的方式，從頭開始教你如何撰寫 Kotlin。本書使用 IntelliJ IDEA 的截圖、在程式碼周圍加上注釋，以及各種視覺效果，貼心地幫助想學程式的讀者。

— Amanda Hinchman-Dominguez，Android 工程師，
Google Groupon 與 Kotlin GDE

1　千里之行，始於足下

一頭栽進 Android 世界

Android 是世上最受歡迎的行動作業系統。全球有數十億 Android 用戶等著下載你的下一個好點子。在這一章，你將藉著**建構基本的 Android app** 並修改它，來了解如何將腦海中的想法化為現實。你將知道如何讓它在實體與虛擬設備上運行。在過程中，你將認識所有 Android app 的兩項核心組件：**activity** 與 **layout**。讓我們開始潛水吧！

歡迎光臨 Android 小鎮

Android 是世上最受歡迎的行動作業系統。根據最新的統計，全球有超過 *30* 億台活躍的 Android 設備，這個數量仍在成長。

Android 是以 Linux 為基礎且開放原始碼的綜合平台，由 Google 支援。它是一個強大的開發框架，涵蓋讓你建立偉大 app 的所有事物。此外，它可讓你將這些 app 部署到各種設備上，包括手機、平板…等。

那麼，Android app 的主要元素有哪些？

我們將使用 Kotlin 與 XML 來建構 Android app。我們會在過程中解釋所有事情，但你也要稍微了解 Kotlin，才能充分理解這本書。

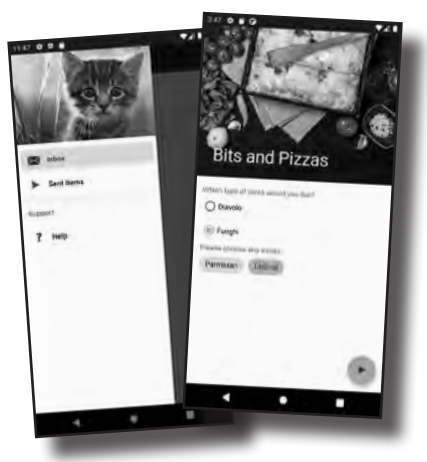
Activity 負責定義 app 可做什麼事

每一個 Android app 都有一或多個 **activity**。activity 是一種特殊的類別，通常是用 Kotlin 寫成的，它負責控制 app 的行為，以及決定如何回應用戶。例如，如果 app 有按鈕，你要在 activity 裡面用程式來說明當按鈕被用戶按下時該如何動作。

用 activity 來定義 app 可做哪些事情。

layout 負責定義每個畫面的外觀

典型的 Android app 有一或多個畫面。你要用 **layout** 檔案或其他的 activity 程式來定義每一個畫面的外觀。layout 通常是用 XML 來定義的，每一個畫面都可以放入按鈕、文字、圖像…等組件。



用 layout 告訴 Android 你的 app 的畫面長怎樣。

還有其他檔案

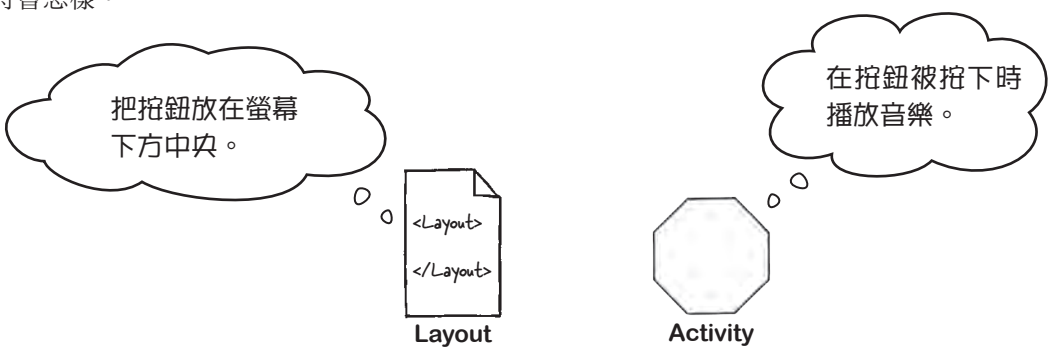
除了 activity 與 layout 之外，Android app 通常還需要一些額外的資源，例如圖像檔與應用程式資料。你可以將額外的檔案加入 app。

app 可能有其他的檔案，例如圖像檔。

Android app 其實只是被放在特定目錄裡面的一群檔案。當你建構 app 時，這些檔案會被包在一起，產生一個可在設備上運行的 app。

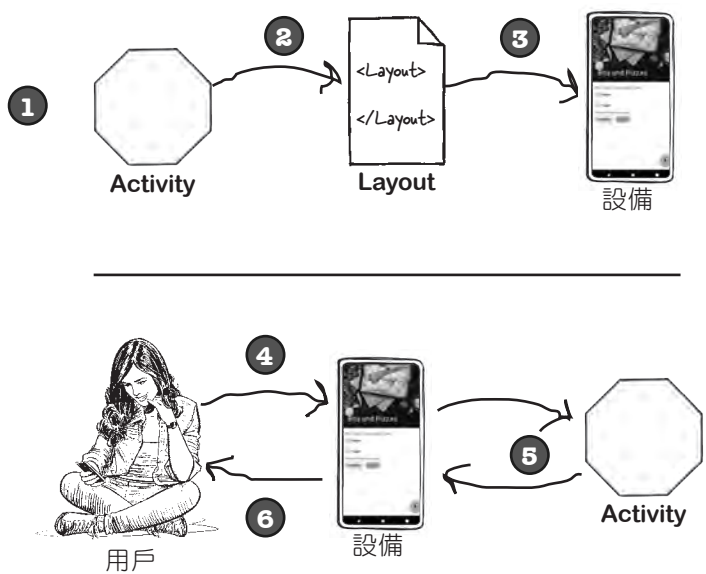
activity 與 layout 是 app 的骨幹

典型的 app 一起使用 activity 與 layout 來定義 app 的用戶介面。layout 可讓 Android 知道如何排列各個螢幕元素，activity 可控制 app 的行為。例如，如果 app 有按鈕，你要用 layout 來指定它的位置，用 activity 來控制當它被用戶按下時會怎樣。



這就是當你在設備執行 app 時，activity 與 layout 的合作情況：

1. **Android 啟動 app 的主 activity。**
2. **activity 叫 Android 使用特定的 layout。**
3. **layout 在設備上顯示出來。**
4. **用戶與 layout 互動。**
5. **activity 回應這些互動，並更新畫面…**
6. **…用戶在設備上看到那些畫面。**

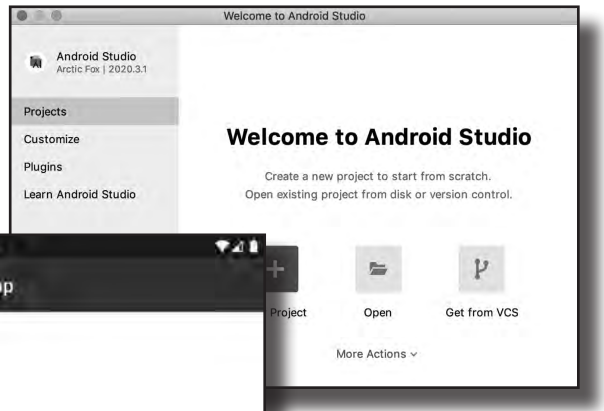


知道 Android app 如何組裝之後，我們來製作一個基本的 Android app 吧！

我們接下來要做這些事情

讓我們來建立一個 Android app，我們只要做幾件事：

本書將使用 Android Studio 來製作所有的 app。



1. **設定開發環境。**
   我們要安裝 Android Studio，它有用來開發 Android app 的所有工具。

2. **製作基本的 app。**
   用 Android Studio 來製作一個簡單的 app，讓它在螢幕上顯示一些文字。

3. **執行 app。**
   我們會在實體設備與虛擬設備上執行 app，來觀察它啟動與運行的情況。

4. **修改 app。**
   最後，我們會稍微修改 app，並再次執行它。

問：Android app 都是用 Kotlin 來開發的嗎？

答：你也可以用其他語言來開發 Android app，例如 Java，但目前使用 Kotlin 是最好的做法，因為有些功能只有 Kotlin 提供。

問：我該怎麼學習 Kotlin ？

答：雖然這有點偏心，但我們認為學 Kotlin 最好的途徑是買一本我們的《*深入淺出 Kotlin*》。它會教你需要知道的所有事情，讓你可以從這本書學到最多知識。

問：你剛才說，我們可以用 activity 程式取代 layout 檔案，來定義 app 的外觀，是嗎？

答：沒錯，你要使用一種稱為 Jetpack Compose 的工具組。

本書會教你如何使用它。現在我們先專心用 layout 檔來定義 app UI。

你在這裡

設定環境
建構 app
執行 app
修改 app

Android Studio：你的開發環境

開發 Android app 的最佳手段是使用 **Android Studio**，它是官方的 Android app 開發 IDE。

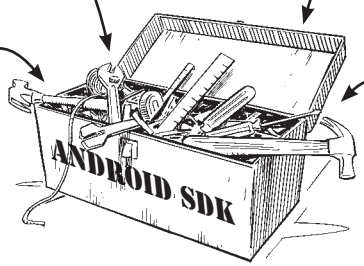
Android Studio 是建立在 IntelliJ IDEA 的基礎之上的，你可能已經熟悉 IntelliJ IDEA 了。Android Studio 有一組程式編輯器、UI 工具與模板，它們都是為了讓你在 Android 小鎮裡面過得更愜意而設計的。

它也有 **Android SDK**（Android Software Development Kit），所有的 Android app 開發工作都需要使用它。Android SDK 包括 Android 原始檔，以及一個編譯器，用來將你的程式編譯成 Android 格式。

Android SDK 的主要組件包括：

SDK 工具
包括用來開發與偵錯 Android app 的完整工具組。

SDK 組建工具
包括建構 Android app 所需的工具。



SDK Platform
每一個 Android 版本都有它自己的 SDK Platform 程式包（package）。它可讓你為該版本的 Android 編譯 app，它裡面也有該版本的原始檔。

SDK Platform 工具
包括與 Android 平台對接的工具，這些工具是回溯相容的，但新功能可能只能在新版的 Android 使用。

你要安裝 Android Studio

Android Studio 包含 Android app 的所有開發工具與功能，我們將使用它來建構本書的所有 app。

在我們進一步討論之前，**你要先在電腦上安裝 Android Studio**。下一頁會詳細告訴你該怎麼做。

- [ ] 設定環境
- [ ] 建構 app
- [ ] 執行 app
- [ ] 修改 app

安裝 Android Studio

為了從這本書學到最多東西，你必須安裝 Android Studio，在此不說明完整的安裝流程，因為這些流程很快就會過時，但你可以按照網路上的說明順利地完成安裝。

首先，到這裡確認 Android Studio 需求：

https://developer.android.com/studio#Requirements

在這裡下載 Android Studio：

https://developer.android.com/studio

這些 URL 可能會變，如果它們失效了，搜尋 Android Studio 可以找到正確的網頁。

然後按照安裝說明來進行安裝。

安裝好 Android Studio 之後，打開它，並按照說明加入最新的 SDK 工具與 Support Libraries。

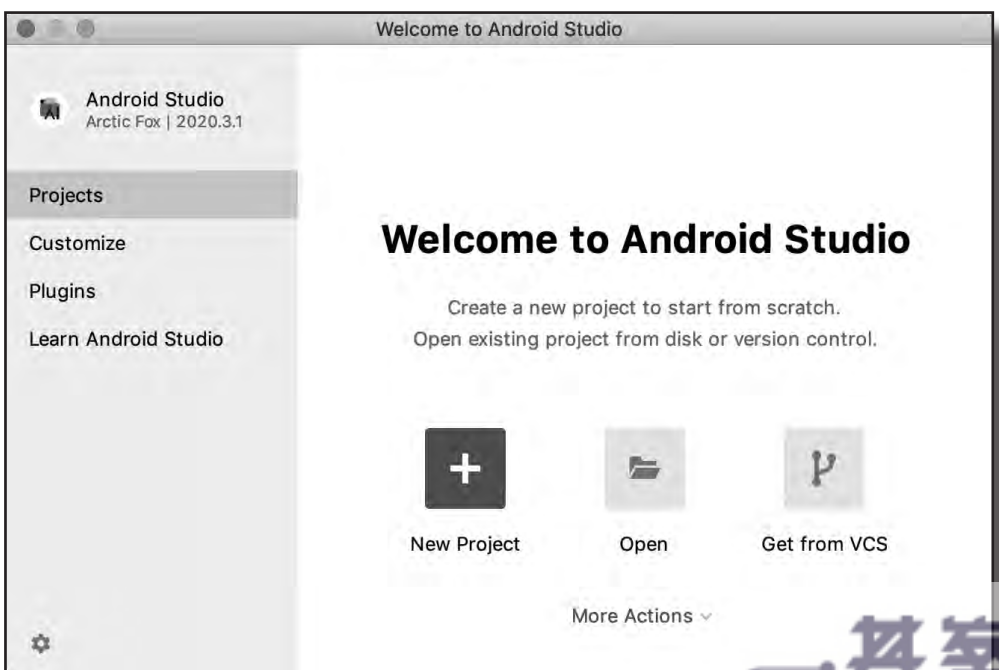
如果你曾經安裝較舊的 Android Studio 版本，建議你恢復 IDE 的預設設定，做法是前往 File 選單，選擇 Manage IDE Settings，然後選擇 Restore Default Settings 選項。

這會重設 Android Studio 保存的所有舊設定，那些設定可能讓你的程式無法運行。

完成之後，你應該可以看到 Android Studio 歡迎畫面：

在這本書裡，我們使用 Android Studio 2020.3.1（別名 Arctic Fox）。務必安裝它以上的版本。

這是 Android Studio 的歡迎畫面，裡面有一組選項可讓你選擇。



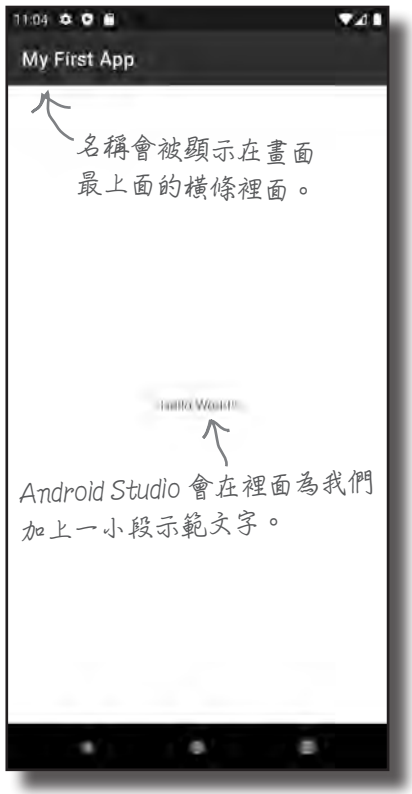
製作基本的 app

這一步完成了，所以將它打勾。

- [x] 設定環境
- [ ] **建構 app**
- [ ] 執行 app
- [ ] 修改 app

設好開發環境之後，你可以開始製作第一個 Android app 了。它的外觀將是：

這是你將製作的 app，它非常簡單，你的第一個 Android app 只需要做到這樣。



讓我們開始製作 app 吧！

問：**我非得使用 Android Studio 來建構 Android app 不可嗎？能不能使用其他的 IDE ？**

答：嚴格來說，你只需要編寫與編譯 Kotlin 程式的工具，以及轉換編譯好的程式，讓程式在 Android 設備上運行的工具。話雖如此，Android Studio 是 Android 官方 IDE，而且 Android 團隊推薦使用它。我們認為它非常適合用來開發 Android app，這就是我們決定使用它的原因。

問：**不使用 IDE 可以寫出 Android app 嗎？**

答：可以，但你會辛苦很多。

使用 IDE 更快速且更方便，所以我們建議你使用 Android Studio。

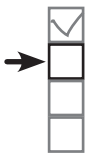
如何製作 app

當你製作新 app 時，你必須為它建立新專案。打開 Android Studio，然後跟我們一起做。

- [x] 設定環境
- [ ] 建構 app
- [ ] 執行 app
- [ ] 修改 app

1. 建立新專案

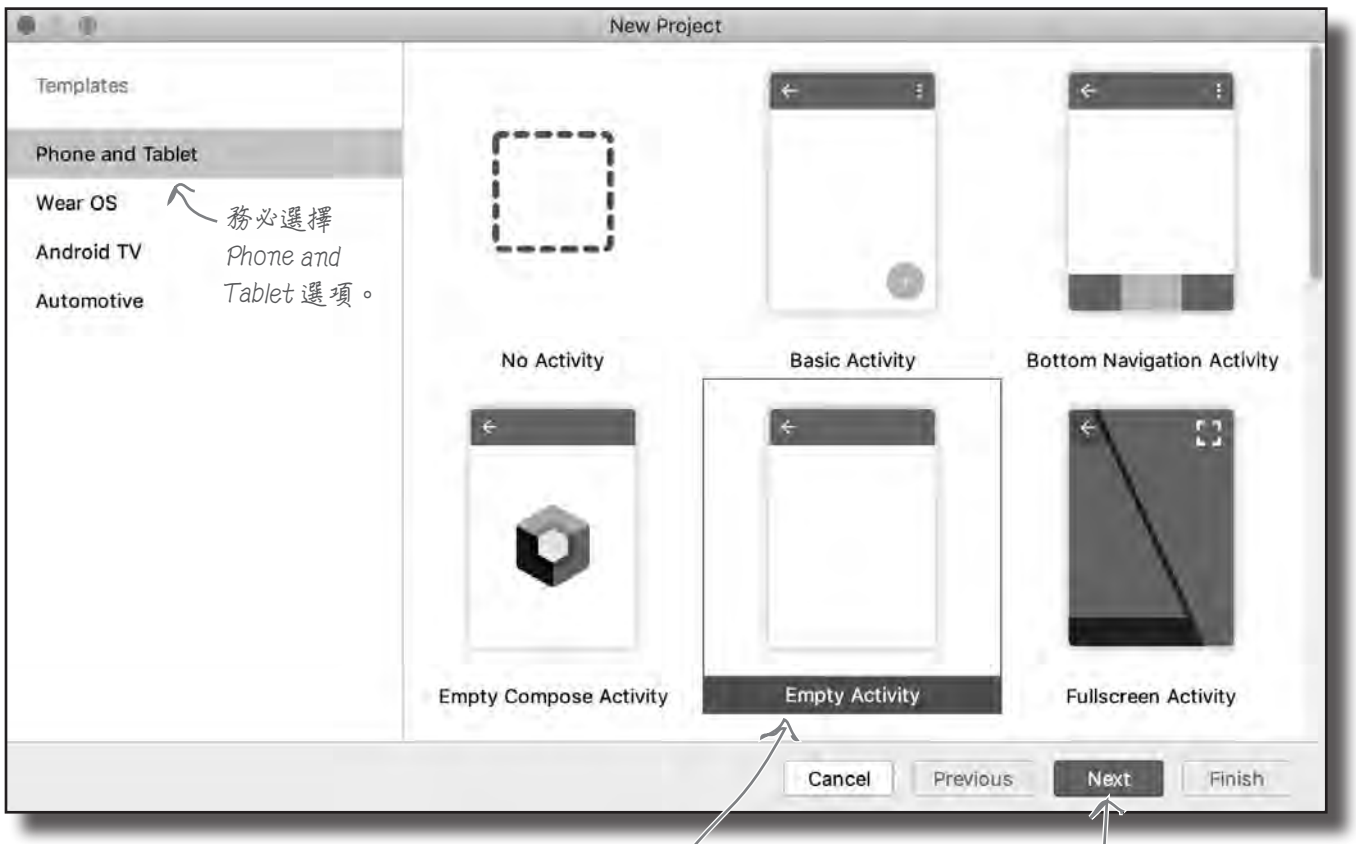
Android Studio 歡迎畫面提供一些選項，你要建立新專案，所以選擇 Projects 選項，然後按下「New Project」。

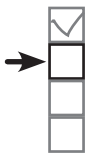


設定環境
建構 app
執行 app
修改 app

2. 選擇專案模板

接下來，你要指定你想製作哪一種 Android Studio 專案。我們要製作一個在手機或平板上運行的 app，它有一個空 activity，所以你要選取 Phone and Tablet，並選擇 Empty Activity 選項。除了 Empty Activity 選項之外還有其他選項，現在只要按下 Next 按鈕進入下一步即可。



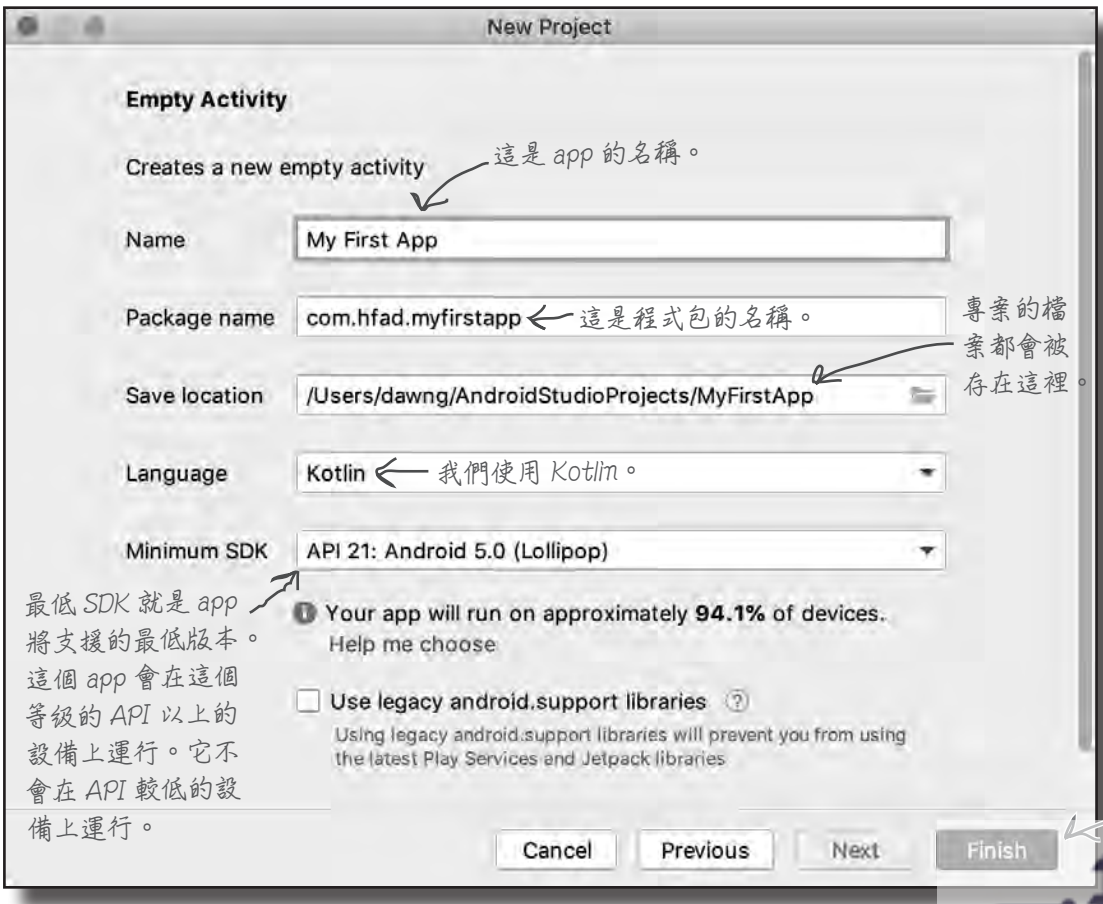


3. 設定專案

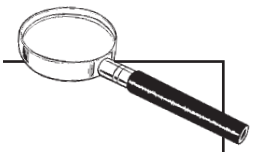
現在你要設定專案，指定 app 名稱、程式包名稱，以及儲存位置。在 Name 輸入「My First App」，在 Package name 輸入「com.hfad.myfirstapp」，並接受預設的儲存位置。

你也要讓 Android Studio 知道你想使用哪一種程式語言，並指定 SDK 的最低版本，它就是 app 將支援的最低版本 Android，下一頁會說明 SDK 等級。

在 Language 選擇 Kotlin，在 Mininum SDK 選擇 API 21，讓 app 可在大多數的設備上運行。按下 Finish 按鈕之後，Android Studio 就會建立專案了。接下來幾頁會介紹它們是什麼意思。



Android 版本探究



你應該聽過有人用可口的零食來稱呼 Android 版本吧？例如 Nougat、Oreo 與 Pie。這些食物代表什麼？

Android 版本有版本號碼與代號，版本號碼就是 Android 的準確版本（例如 9.0），而代號是比較通俗的名稱，可能涵蓋幾個 Android 版本（例如 Pie）。API 等級就是 app 使用的 API 的版本。例如，Android 9.0 對應的 API 等級是 28。

在 Pie 之後，Android 的代號就不使用可口的食物了，例如，Android 10.0 直接稱為 Android 10。

版本	代號	API 等級
1.0		1
1.1	Petit Four	2
1.5	Cupcake	3
1.6	Donut	4
2.0–2.1	Eclair	5–7
2.2–2.2.3	Froyo	8
2.3–2.3.7	Gingerbread	9–10
3.0–3.2.6	Honeycomb	11–13
4.0–4.0.4	Ice Cream Sandwich	14–15
4.1–4.3.1	Jelly Bean	16–18
4.4–4.4.4	KitKat	19–20
5.0–5.1.1	Lollipop	21–22
6.0–6.0.1	Marshmallow	23
7.0–7.1.2	Nougat	24–25
8.0–8.1	Oreo	26–27
9.0	Pie	28
10.0	10	29
11.0	11	30
12.0	12	31

這些版本幾乎沒有人使用了。

大部分的設備都使用這些 API 之一。

當你開發 Android app 時，你必須考慮 app 將要與哪些 Android 版本相容。如果你指定 app 只與最新版的 SDK 相容，你應該會發現它無法在許多設備上運行。當你建立新專案時，你可以按下「Help me choose」選項，來了解使用特定版本的設備的百分比。

你已經建立第一個 Android 專案了

設定環境
建構 app
執行 app
修改 app

完成 New Project wizard 引導的步驟之後，Android Studio 會用一分鐘左右的時間建立專案。在這段時間，它做了這些事情：

根據你的規格來設定專案。

Android Studio 會檢查你希望 app 支援的最低 SDK，並將基本且有效的 app 需要的所有檔案與資料夾加進來。它也會設定程式包結構，以及 app 的名稱。

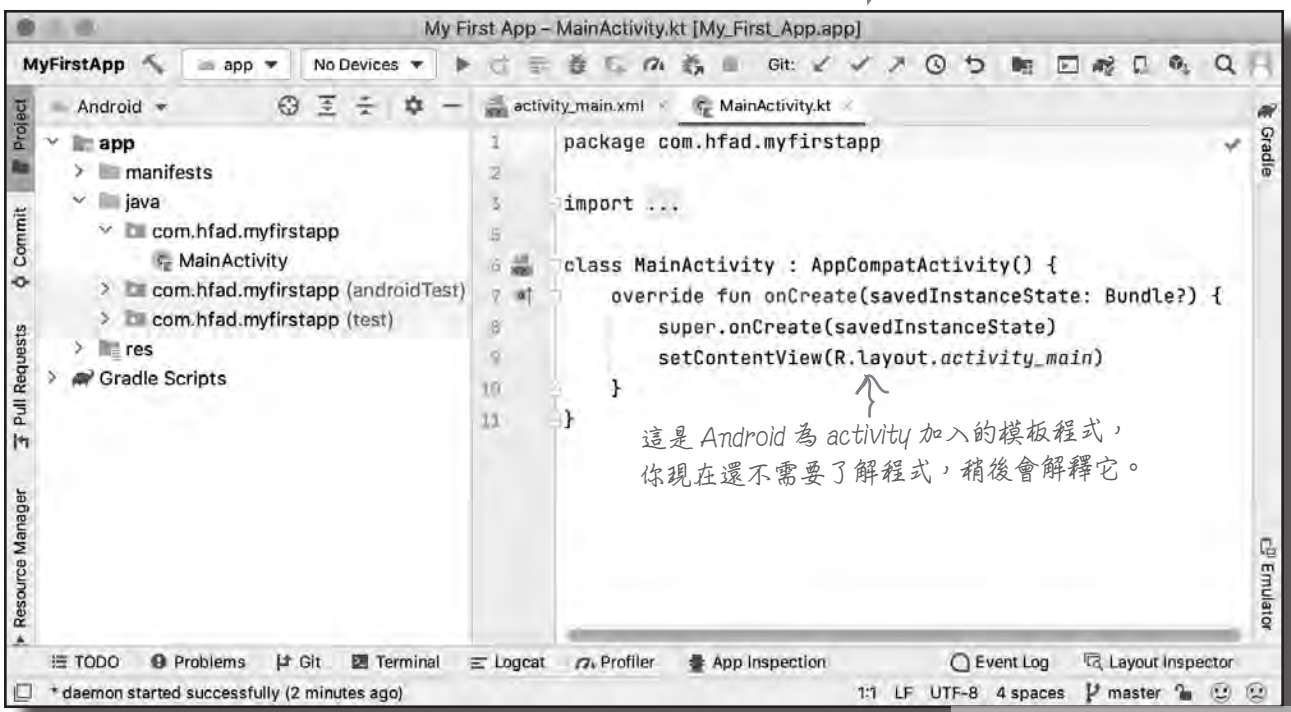
加入一些模板程式。

模板程式是由一個以 XML 寫成的 layout 與一個以 Kotlin 寫成的 activity 組成的。本章稍後會詳細介紹它們。

Android Studio 將專案建立完成之後會自動打開它。

這是我們的專案的樣子（看起來有點複雜，但別怕，接下來幾頁會好好講解它）：

這是在 Android Studio 裡面的專案。



剖析你的新專案

Android app 其實是被放在特定的資料夾結構裡面的一堆檔案，Android Studio 會在你建立新 app 時為你安排好一切。若要查看這個資料夾結構，最簡單的方法是使用 Android Studio 最左邊的 explorer。

資料夾結構包含不同類型的檔案

explorer 裡面有你目前打開的所有專案。在這裡有一個名為 MyFirstApp 的專案，它就是我們剛才建立的那一個。

當你在 explorer 裡面瀏覽各個資料夾時，你將發現 wizard 已經為你建立了各種類型的檔案與資料夾，例如：

Kotlin 與 XML 原始檔

Android Studio 自動建立一個名為 *MainActivity.kt* 的 action 檔，以及一個名為 *activity_main.xml* 的 layout。

資源檔

包括圖像檔、app 使用的主題，以及 app 使用的任何常見的 `String`（字串）值。

Android 程式庫

你曾經在 wizard 裡面指定 app 可支援的最低 SDK。Android Studio 會確保 app 加入與該版本有關的 Android 程式庫。

設定檔

設定檔可讓 Android 知道有哪些東西被加入 app，以及 app 應該如何運行。

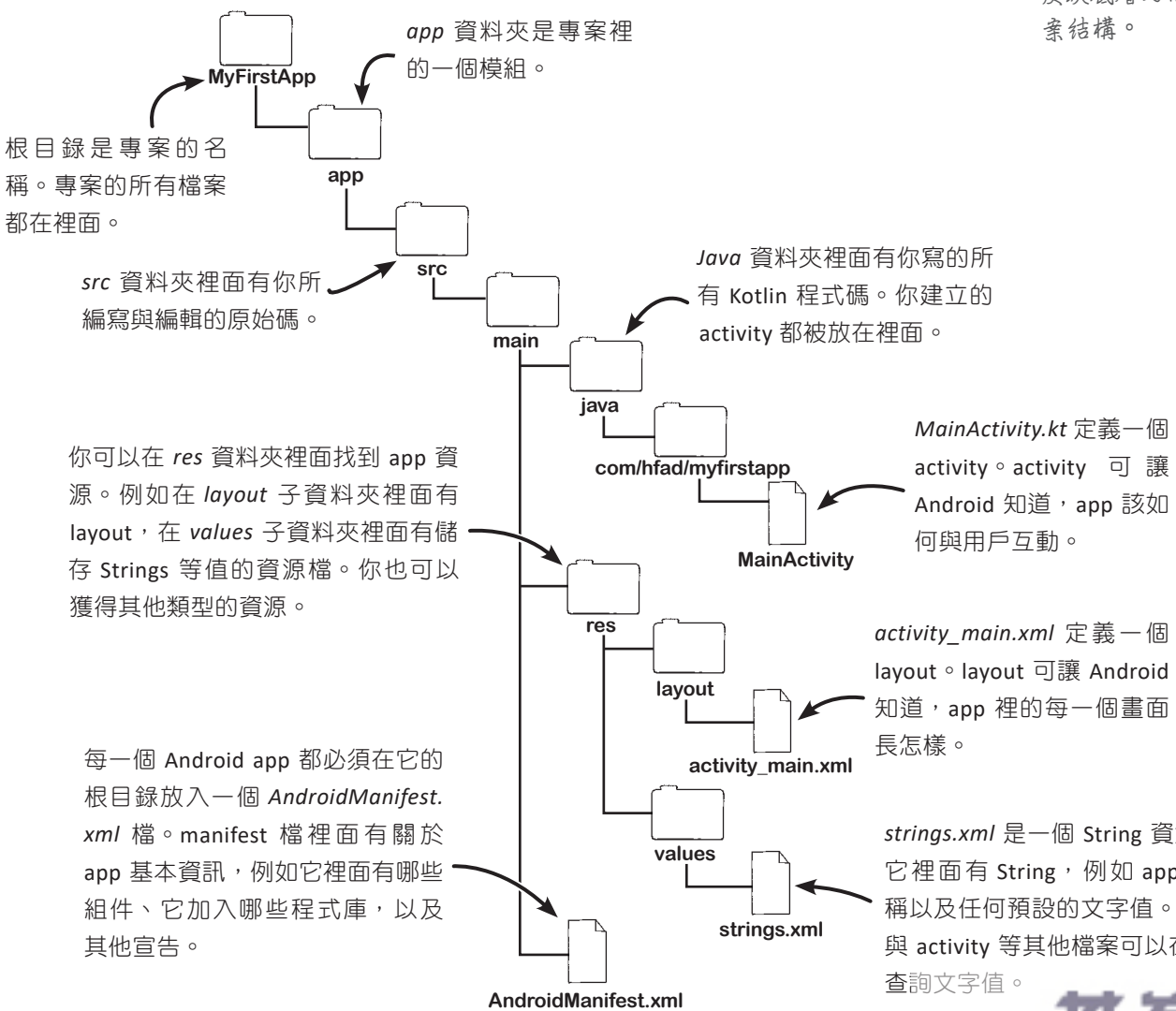
我們來仔細研究專案中的一些重要檔案與資料夾。

在專案中加入關鍵檔案

Android Studio 專案使用 Gradle 組建系統來編譯和部署 app，而 Gradle 專案有一種標準結構，下面是在那個結構中，你將使用的一些關鍵檔案與資料。

你可以在 Android Studio 裡面將 explorer 畫面從 Android 改成 Project 來查看資料夾結構，做法是按下 explorer 窗格頂部的箭頭，然後選擇 Project 選項。

使用這個箭頭來改變 explorer 畫面。我們通常使用 **Project** 畫面，因為它反映底層的檔案結構。



用 Android Studio 編輯器來編輯程式

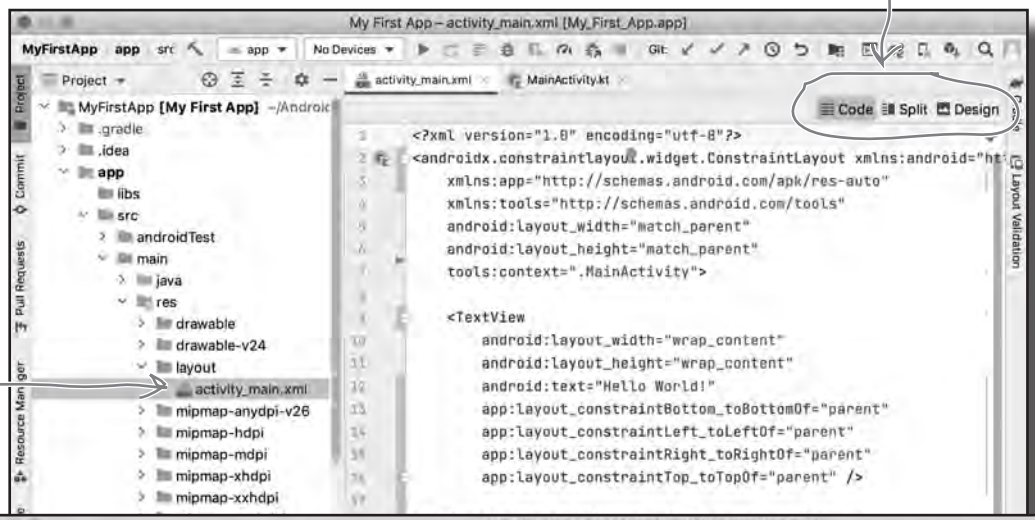
你要使用 Android Studio 編輯器來觀看與編輯檔案。在想要處理的檔案上面按兩下之後，檔案的內容會出現在 Android Studio 視窗的中間。

用這些按鈕來選擇編輯器。

程式（code）編輯器

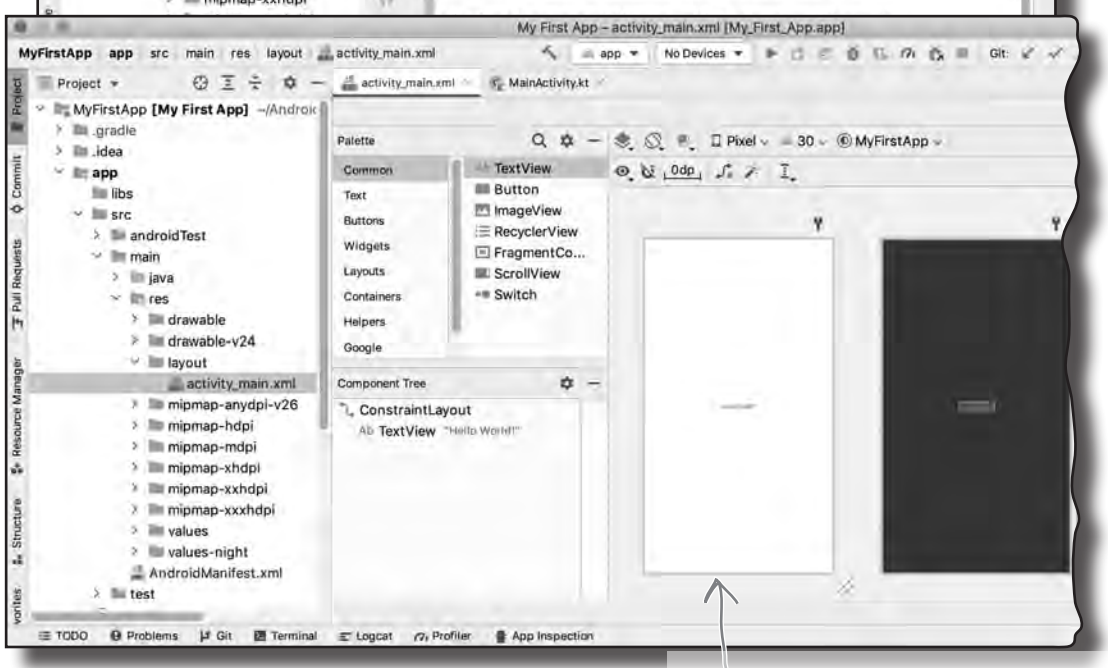
大部分的檔案都可以在程式編輯器裡面顯示，它很像文字編輯器，但是它有額外的功能，例如語法突顯，以及程式碼檢查。

在 explorer 裡面的檔案上面按兩下之後，檔案內容就會顯示在編輯器窗格裡面。



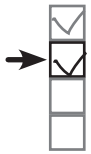
設計（design）編輯器

如果你要編輯 layout（例如 *activity_main.xml*），你有另一種辦法，你可以使用設計編輯器，它可以讓你將 GUI 組件拉入 layout，並且按照你的意思排列它們，而不需要編輯程式碼。同一個檔案在程式編輯器與設計編輯器裡面有不同的畫面，你可以在兩者之間來回切換。



你可以使用視覺化的編輯器，藉著拖曳與放下組件來編輯 *layout*。

前情提要



到目前為止，我們做了兩件事：

1. **設定開發環境。**
   我們要使用 Android Studio 來開發 Android app，所以你必須在電腦上安裝它。

2. **建立基本 app。**
   我們使用 Android Studio 來建立新的 Android 專案。

你已經看到 app 在 Android Studio 裡面的樣貌，以及它們如何組在一起了。但是你想要看它跑起來的樣子，對不對？

Android Studio 可讓你用兩種方式來執行 app：在實體 Android 設備上，以及在虛擬設備上。我們將在幾頁之後介紹這兩種方法。

問：為什麼 Android Studio 將 Kotlin 程式放在 *java* 資料夾裡面？

答：在 Android 團隊將 Kotlin 視為他們的首選語言之前，大多數的 Android app 都是用 Java 來開發的。*java* 資料夾是舊時代的遺產，但是它以後可能會改變。

問：你說專案有一個名為 *MainActivity.kt* 的 activity，以及一個名為 *activity_main.xml* 的 layout。它們來自何方？

答：我們在建立專案時，選擇 Empty Activity。Android Studio 自動將那個 activity 檔命名為 *MainActivity.kt*，並加入對映的 layout 檔，名為 *activity_main.xml*。本章稍後會進一步討論這些檔案。

問：你說 Android Studio 專案使用 Gradle，能不能再說一次那是什麼？

答：Gradle 是可讓你編譯、組建與交付程式碼的工具。大多數的 IDE 都使用 Gradle，它不是只能用來開發 Android app。

所有的 Gradle 專案都有一個標準的資料夾結構，而 Android Studio 讓它的所有專案使用這個資料夾結構。

問：為什麼 Android 使用 Gradle？

答：在開發 Android app 時，你通常需要加入 Android SDK 沒有的其他程式庫，Gradle 可以幫你下載這些程式庫，讓你的程式設計生活更輕鬆。

Gradle 也將 Groovy 與 Kotlin 當成腳本語言來使用，也就是說，你可以用 Gradle 來輕鬆地建立相當複雜的版本（build）。

問：我需要了解多少 Gradle？

答：你不需要有任何 Gradle 經驗就可以學會本書大多數的內容，我們會在需要的時候，解釋你需要知道的每一件事情。

連連看？

這是來自 *MainActivity.kt* 這個 activity 檔的程式碼。我們知道你還沒有看過 activity 的程式，但請你試試看，能不能將下方的敘述連到正確的程式碼。為了幫你開始，我們已經完成一題了。

MainActivity.kt

```
package com.hfad.myfirstapp

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }

}
```

這是程式包的名稱。

這些是在 MainActivity 裡面使用的 Android 類別。

指定想使用的 layout。

實作 AppCompatActivity 類別的 onCreate() 方法。當 Android 初次建立 activity 時，它會呼叫這個方法。

MainActivity 繼承類別 AppCompatActivity。

連連看？

解答

這是來自 *MainActivity.kt* 這個 activity 檔的程式碼。我們知道你還沒有看過 activity 的程式，但請你試試看，能不能將下方的敘述連到正確的程式碼。為了幫你開始，我們已經完成一題了。

MainActivity.kt

```
package com.hfad.myfirstapp

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }

}
```

這是程式包的名稱。

這些是在 MainActivity 裡面使用的 Android 類別。

指定想使用的 layout。

實作 AppCompatActivity 類別的 onCreate() 方法。當 Android 初次建立 activity 時，它會呼叫這個方法。

MainActivity 繼承類別 AppCompatActivity。