

書名頁

作 者／趙令文

執行編輯／單春蘭

特約美編／葳豐企業

封面設計／韓衣非

行銷副理／羅凱頤

總 編 輯／黃錫鉉

社 長／吳濱伶

發 行 人／何飛鵬

出 版／電腦人文化

發 行／城邦文化事業股份有限公司

歡迎光臨城邦讀書花園

網址：www.cite.com.tw

香港發行所／城邦（香港）出版集團有限公司

香港灣仔駱克道193號東超商業中心1樓

電 話：(852) 25086231

傳 真：(852) 25789337

E-mail：hkcite@biznavigator.com

馬新發行所／城邦（馬新）出版集團

Cite (M) Sdn Bhd

41, Jalan Radin Anum, Bandar Baru Sri Petaling,
57000 Kuala Lumpur, Malaysia.

電 話：(603) 90578822

傳 真：(603) 90576622

E-mail：cite@cite.com.my

國家圖書館出版品預行編目資料

Android App開發者必修16堂課：最強範例！經典得獎程式碼完全解析 / 趙令文 著. -- 初版. -- 臺北市：電腦人文化出版；城邦文化發行，民102.05；公分

ISBN 978-986-199-398-0(平裝)

1.行動電話 2.行動資訊 3.軟體研發

448.845029

102006472

印刷／士達印刷有限公司

2013年(民102)5月 初版一刷

Printed in Taiwan

定價／520元

● 如何與我們聯絡：

1. 若您需要劃撥 購書，請利用以下郵撥帳號：

郵撥帳號：19863813 戶名：書虫股份有限公司

2. 若書籍外觀有破損、缺頁、裝訂錯誤等不完整現象，想要換書、退書，或您有大量購書的需求服務，都請與客服中心聯繫。

客戶服務中心

地址：10483 台北市中山區民生東路二段141號B1

服務電話：(02) 2500-7718、(02) 2500-7719

服務時間：週一～週五上午9：30～12：00，

下午13：30～17：00

24小時傳真專線：(02) 2500-1990～3

E-mail：service@readingclub.com.tw

3. 若對本書的教學內容有不明白之處，或有任何改進建議，可將您的問題描述清楚，以E-mail寄至以下信箱：pcuser@pcuser.com.tw

4. PCuSER電腦人新書資訊網站：

<http://www.pcuser.com.tw>

5. 電腦問題歡迎至「電腦QA網」與大家共同討論：<http://qa.pcuser.com.tw>

6. PCuSER專屬部落格，每天更新精彩教學資訊：
<http://pcuser.pixnet.net>

7. 歡迎加入我們的Facebook粉絲團：

<http://www.facebook.com/pcuserfans>
(密技爆料團)

<http://www.facebook.com/i.like.mei>
(正妹愛攝影)

※詢問書籍問題前，請註明您所購買的書名及書號，以及在哪一頁有問題，以便我們能加快處理速度為您服務。

※我們的回答範圍，恕僅限書籍本身問題及內容撰寫不清楚的地方，關於軟體、硬體本身的問題及衍生的操作狀況，請向原廠商洽詢處理。

版權聲明

本著作未經公司同意，不得以任何方式重製、轉載、散佈、變更全部或部分內容。

商標聲明

本書中及所附光碟所提及國內外公司之產品、商標名稱、網站畫面與圖片，其權利屬各該公司或作者所有，本書僅作介紹教學之用，絕無侵權意圖，特此聲明。

自序

從事 Android App 開發的實務教學已經三年多了，這三年多來的變化也相當多。不只是開發工具或是 API 上的變化，甚至於 App 市場上的變化。現在的您還在玩 Angry Bird 嗎？還是開始換玩 Candy Crush 了？之前 Google play 在台灣消費市場上的風風雨雨，終於又可以開始販售購買付費 App 了，三年前相關單位辦了一場「一千五百萬創意成金」活動，只要提案通過審核並如期上架，一個 App 就可能獲得 3-8 萬元獎金（敝人也提了三個 App 獲得獎金），而現在的市場機制沒有這樣的活動了，轉變為鼓勵的是質量較佳的 App。敝人也從教育訓練市場間接看到就業市場上的需求不斷攀升。

很多初學者想要以快速方式學習 Android App 的開發，卻往往適得其反，雖然相關學習文件資料琳瑯滿目，但是大部分都是片段的技術資料。可能很容易找到如何發出通知訊息的知識，卻不知該如何應用或是做出不同的變化；找到資料也將程式碼複製後可以反轉手機變成靜音，不知如何善加運用手機感應器成為方向控制器等等。相信我，你絕對可以找到成千上萬的 Android App 開發密技，但是真的並不適合初學者的學習。除非你知道每一列程式碼的存在意義，否則複製貼上絕對是最糟糕的學習模式。我的上課模式不會有投影片，也沒有事先寫好的程式碼，就是從專案建立開始開發，寫出來的每一列都要清楚在做什麼，觀念的建立非常重要，觀念清楚之後，想要做出不同的變化就不成問題了。因此，本書的撰寫希望可以將初學者導向建立觀念式學習模式，不是提供片斷的密技而已。書中最後專題範例就是 2012 年在資策會行動裝置開發班的 Android 遊戲開發課程中以 12 小時授課時間，從專案建立開始實際開發，再利用課後時間進行修正微調後的作品，竟然還能僥倖獲得 2012 年中華電信社會組的優勝作品。

我不是專家，只是愛玩而已。2011 年以改編 Lode Runner 經典遊戲參加其他比賽，專家評審建議我加上自編關卡網路分享功能，可以使這個 App 有加分效果。當下敝人極度不同意，因為這樣的行為表面看似增加玩家之間的互動，事實上卻增加玩家遊戲的潛藏隱私安全性，為了繼續參賽而將該功能開發上去，並還在電視上展現即時分享功能。賽後自行另外以原先版本約晚三個月在 Google play 上架發行。一年半過去了，單純遊戲版本的實際安裝數量已經是網路分享版本的兩倍以上的數量。

最後以此書獻給最摯愛的家父趙光明先生與家母謝寶秀女士，奉獻畢生心力於教育事業的父母，不斷地鼓勵指導也是投身教育訓練的敝人，才能完成此拙著。當然，也要感謝陪伴我趕稿的老婆，及幫我測試玩 App 遊戲的孩子們。另外，更是感謝協助處理 App 視覺藝術部份的墨比斯 — 雲雲手。

趙令文 Brad

2013.5.4

目 錄

01

開發環境建置與基本使用

1-1	學習開發的基本觀念.....	1-2
	Java 語言的角色	1-2
	Unix/Linux 的檔案系統.....	1-3
	學習目標.....	1-3
1-2	安裝 JDK	1-4
1-3	安裝 Eclipse	1-5
1-4	設定 Eclipse	1-7
	在 Eclipse 外掛 ADT	1-10
1-5	安裝設定 Android SDK	1-10
	建立及使用模擬器.....	1-12

02

基本程序運作原理與應用

2-1	「Hello, World? Hello, Lottery!」.....	2-2
	建立新專案	2-2
	版面配置.....	2-6
	開發程式.....	2-10
	安裝執行測試	2-12
	存取控制元件	2-13
2-2	「BMI?Lottery!」.....	2-13
	按鈕事件處理模式	2-15
	開發設計功能	2-17
	修改程式	2-18
	加上歡迎畫面	2-19
2-3	寫完了，然後呢？.....	2-19
	調整啟動程序	2-22
2-4	Activity 的生命週期	2-25
	生命週期的觀念	2-26
	測試實作	2-28

	開始觀察	2-31
2-5	Activity 切換 Activity	2-32
	僅作啟動切換	2-32
	傳遞資料過去	2-33
	切換之後回來確認	2-34
	將資料傳達回來	2-35
2-6	Service 的運作應用	2-36
	生命週期實測	2-37
	與執行緒共舞	2-40
	透過 Broadcast 發送資料給前景	2-42

03

基本使用者介面與事件觸發

3-1	條列顯示元件 ListView	3-2
	基本款式	3-2
	進階款式	3-6
3-2	線性配置 LinearLayout	3-8
3-3	相對配置 RelativeLayout	3-13
3-4	表格配置 TableLayout	3-17
3-5	網格顯示 GridView	3-20
3-6	滑動顯示 ViewFlipper	3-24

04

對話框與通知事件處理

4-1	AlertDialog 對話框的使用	4-2
	建立 AlertDialog 物件	4-2
	訊息對話框	4-2
	確認式對話框	4-5
	選擇式對話框	4-7
	進階選擇式對話框	4-11
	自訂對話框	4-13
4-2	自訂對話框（Dialog）與日期時間對話框	4-13
	日期選擇對話框	4-17
	時間選擇對話框	4-19
	一般的 Toast	4-21

4-3	Toast 及自訂 Toast.....	4-21
	自訂 Toast	4-23
4-4	進度顯示對話框	4-25
	ProgressDialog	4-26
4-5	通知列處理模式	4-29
	版本差異.....	4-30
	API Level 11 之前	4-30
	API Level 11+	4-31
	應用場合	4-33

05 進階程序運作原理與應用

5-1	多重執行緒 Thread	5-2
	開發重點觀念	5-2
	存取 View 元件	5-5
	提早結束執行緒的生命週期	5-7
	另外一種開發方式	5-8
5-2	定時及週期任務 (Timer & TimerTask)	5-9
	使用觀念	5-13
	生命週期	5-13
5-3	同步任務 AsyncTask	5-13
	定義泛型參數	5-16
	基本開發程序	5-16
	程式架構	5-17
5-4	倒數計時器	5-19
	開發模式	5-20
	直接實作練習	5-20

06 選單與動作列處理

6-1	選單 Menu	6-2
	Options menu 選項選單 (硬體選單鍵)	6-2
	Context menu 內容選單	6-6
	Popup menu 彈出式選單	6-10
6-2	動作列 Action Bar	6-12

07

自訂 View 與 SurfaceView

7-1	自訂 View：繼承 View	7-2
	一般觸控事件偵測處理	7-12
	手勢偵測事件處理	7-12
7-2	自訂 View 與觸控手勢事件處理	7-12
7-3	自訂 SurfaceView: 繼承 SurfaceView	7-16
	前置準備	7-21
7-4	以自訂 View 來實現手寫簽名 App 範例實作	7-21
	開始處理簽名的手勢偵測處理	7-23
	處理外部功能	7-28

08

資料存取

8-1	偏好設定	8-2
	處理方式	8-2
	基本處理程序	8-3
	範例說明	8-3
	完整範例	8-6
	使用觀念	8-8
	寫出基本程序	8-8
8-2	內部檔案存取機制	8-8
	讀入基本程序	8-10
	SDCard 檔案系統基本觀念	8-12
8-3	外部檔案存取	8-12
	判斷 SDCard 的掛載點 (Mount Point)	8-13
	應用程式檔案應該在哪裡	8-14
	開啟寫出資料的權限	8-14
	開始進行程式開發	8-16
	寫出資料檔案	8-16
	讀入資料檔案	8-17
	建立資料庫的輔助類別物件	8-18
	預先處理模式	8-18
8-4	行動裝置資料庫處理機制 SQLite	8-18
	簡單查詢資料	8-20

新增資料.....	8-21
刪除資料.....	8-21
修改資料.....	8-22
進一步了解查詢方式.....	8-23
8-5 應用 App 資源中的資料存取資料：	
遊戲關卡資料處理為例	8-23
定義資料.....	8-24
讀取資料檔案	8-25
程式中讀取方式	8-26

09

網際網路相關

9-1 網路介面及 IP Address	9-2
裝置的網路狀態	9-2
網路介面的 IP Address	9-3
取得裝置連線 IP Address	9-5
建構 IP Address 物件實體.....	9-5
處理模式.....	9-6
9-2 UDP 通訊協定的資料存取	9-6
實作測試.....	9-7
9-3 TCP 通訊協定的資料存取	9-12
處理模式.....	9-13
實做測試.....	9-13
以 AndroidHttpClient 及 DefaultHttpClient 實做	9-17
9-4 Http 通訊協定的資料存取	9-17
以 java.net.HttpURLConnection 實作	9-21
9-5 WebView 使用	9-22
基本的處理方式 - 直接放進 Activity 中.....	9-23
基本的處理方式 - 以版面配置方式處理.....	9-24
進一步設定 WebView 功能.....	9-30

10

影音多媒體與相機

10-1 播放音樂	10-2
基本概念	10-2

SDCard 上的音樂播放	10-3
播放專案資源中音樂檔案	10-5
播放 URL 的音樂檔案	10-5
暫停繼續播放	10-6
停止播放	10-7
10-2 音效處理	10-7
建構 SoundPool 物件實體	10-8
即時播放音效	10-8
呼叫其他錄音程式	10-9
10-3 錄音處理	10-9
自訂錄音處理程序	10-11
錄影	10-13
10-4 錄影放映	10-13
呼叫其他錄影程式	10-14
自訂錄影程序	10-15
播放影片	10-17
10-5 相機	10-18
呼叫其他照相程式	10-19
自訂相機程式	10-21

11 地圖與衛星定位系統

11-1 GPS 定位	11-2
開始基本實作	11-2
較佳位置取得	11-4
11-2 基本 Google Map	11-9
開發前置作業	11-10
Hello, Map	11-11
在 Android 開發上的應用	11-13
JavaScript 處理說明	11-15
11-3 進階 Google Map	11-15
JavaScript 資料傳回 Android	11-18
以 Android 傳遞資料給 JavaScript	11-18

12 感應器運作原理及應用

12-1	感應器運作原理與應用	12-2
	基本概念.....	12-2
	處理原則.....	12-3
	實作開發.....	12-3
	使用者裝置支援處理.....	12-5
12-2	三軸加速感應器	12-6
12-3	重力加速度感應器.....	12-10
12-4	磁極方向感應器	12-13
12-5	光線 / 溫度 / 濕度 / 壓力感應器.....	12-17

13 資源與國際化

13-1	提供資源內容	13-4
	預設資源內容及架構.....	13-4
	替代選擇性資源內容.....	13-6
	程式碼中存取資源內容	13-9
13-2	存取資源內容	13-9
	XML 中存取資源內容.....	13-10
13-3	應用程式執行中的改變	13-11
	設計一個保留及回存物件	13-12
	支援的區域國別	13-12
13-4	資源內容的區域化	13-12
	進一步認識專案資源.....	13-14
	資源型態.....	13-22
	區域化確認檢查	13-23

14 系統功能與裝置控制

14-1	行動裝置相關辨識	14-2
14-2	行動電話通話狀態	14-4
14-3	行動電話用戶相關資料	14-6
	使用者帳號	14-7

取得聯絡人姓名 14-7

使用者的相簿 14-8

14-4 開發者基本道德 14-9

15 實際專案開發

15-1	彈指磚塊王 (Bricks Fighter)	15-2
	App 簡易架構	15-3
	歡迎頁面	15-3
	遊戲關卡選單	15-6
	遊戲主頁	15-10
	開發動機	15-19
15-2	掏金沙 (Lode Runner)	15-19
	著手規劃	15-20
	遊戲架構	15-21
	關卡選單	15-25
	遊戲畫面	15-25
	關卡地圖	15-28
	敏感爭議	15-37
15-3	炸彈超人 (Bomb King)	15-37
	歡迎畫面	15-38
	個性簽名產生器	15-40
15-4	其他應用程式開發專案	15-40
	開發觀念原則	15-52

16 APP 發佈

16-1	包裝發佈到 Google Play	16-2
	包裝成為 apk	16-2
	首次註冊開發者	16-6
	發佈 Apk 到 Google Play	16-8
16-2	App 創意開發與比賽經驗心得分享	16-10

01

Chapter

開發環境建置與基本使用

- 1-1 學習開發的基本觀念
- 1-2 安裝 JDK
- 1-3 安裝 Eclipse
- 1-4 設定 Eclipse
- 1-5 安裝設定 Android SDK





1-1 學習開發的基本觀念

Android App 的開發應用在這幾年之間，已經非常的普遍。光是在 Google play 上面的 App 數量就非常的驚人，還不包含其他各家市集，以及各不同行業領域業者提供自家相關的 App。事實上，Android App 的開發應用上手非常容易，只需要基本的 Java 程式設計能力，加上熟習 Android API 即可開始進行開發，而本書的重點就是放在 Android API 的學習上。

Java 語言的角色

以 Android SDK 的開發模式，完全就是採用 Java 程式語言。所以進入開始開發之前，最好能夠對於 Java 語言可以熟悉了解，這樣絕對有助於開發過程。其實有許多剛開始入門的學習者，可能一心想要以快速的方式學習開發 Android App，於是就買書、上網爬文，當看到與自己打算開發專案類似功能的範例，馬上進行複製貼上的慣性動作，順利的話，可以先看到原開發者的呈現執行效果；但是大部分卻都不是那麼順利，可能是 API Level 設定下載不同，可能是開發環境或是模擬器差異，或是編譯使用函式庫放置路徑不同等等一大堆問題，還可能不是解決一個問題就結束。因此就需要開發 Java 語言的基本觀念來進行排解，所以 Java 的基本認識相當重要。

當可以開始將原開發者的原始碼執行之後，想要修改成為自己想要的模式。此時最基本的動作就是認識原開發者當時設定各自變數所代表的意義何在，才能正確的進行修改調整，而一般開發者的變數名稱的命名應該都還容易判斷出來。但是整段程序方法的作用何在？你應該如何下手呢？還是需要 Java 語言的基本觀念來處理。

好，那麼就不要複製貼上的學習模式，從頭開始來學習開發，那麼就更是完全在寫 Java 程式語言。

至此，原本還不會 Java 語言的人可能已經不再打算買這本書了。對！買了之後的學習效果不大，放回書架上去吧。但是，提供一種自認為還不錯的學習模式，Java 語言的學習可以在目前發達的網際網路上面搜尋出許多資料，一邊學習 Android App 的開發，當遇到 Java 語言的問題或是無法理解之處，馬上跳出來弄到清楚之後，再回來繼續學習開發。切記，寫在自己專案中的程式碼，沒有不認識的東西。這樣你可以考慮再從書架上那下這本書，到櫃台結帳囉。

1

■ Unix/Linux 的檔案系統

Android 作業系統來自於 Linux，所以其檔案系統是以 Linux 的單根作業系統來進行，如果熟悉 MS Windows 的多磁碟的作業系統的學習者，只需要想成只有一個 C 磁碟機而已，既然只有一個 C 磁碟機，就乾脆不需要提及 C 磁碟機這件事。再來就是路徑符號剛好與 MS Windows 的作業系統相反，而是使用 / 反斜線，對於許多程式語言開發而言，\ 斜線符號是和跳脫字元一樣，容易造成開發上的困擾，例如在使用表示網址的路徑符號也是使用 / 反斜線（大部分的作業系統都是使用 / 反斜線，少部份的作業系統是使用 \ 斜線，但卻是大多數使用者在用）。再來就是大小寫嚴格區分這件事與 MS Windows 是不一樣的。

大致上常見的開發上差異如此而已，也倒不需要先去熟悉了解 Linux 之後再來學習 Android。

■ 學習目標

既然想學習 Android App 開發，總有個想要開發的目標吧！如果有個目標想要開發，對於學習效果而言會比較佳；沒有特定目標的話，可以依照本書上面的專案來進行亦可。

本書無法提供創意的思考模式，如果硬要說不在本書設定的主題範疇內，那是藉口。因為筆者本身不是這分面的專長，所有筆者開發的專案都是

來自於個人的需求所產生的。因為偏愛當年紅白機的遊戲，所以想自行改編開發 Lode Runner，炸彈超人等等；因為從小第一次接觸的電玩是打磚塊，所以也來改寫回味一番；因為自己想要在平板電腦上面有個好用的萬用筆記功能，可以一般筆記，同時照相錄音錄影，衛星定位等等多樣化的功能，所以就想要自行開發，這就是我的創意來源。



重點：複製貼上是最爛的學習模式，可能一開始會有感覺（錯覺），中間一定會卡關，而且卡很久。只要是寫在自己專案中的程式碼，沒有不認識的東西。開發出來的 App 才是你的。



1-2 安裝 JDK

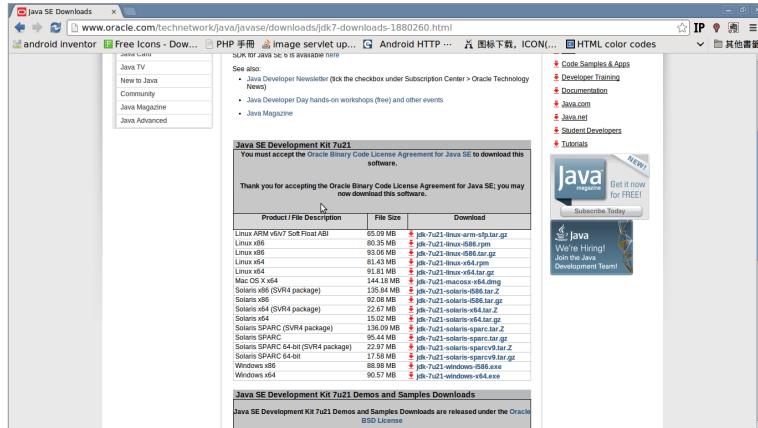
Java 語言的是 Android 的基本，建議先將開發環境安裝上 JDK (Java Development Kit)，這是當年 Sun 公司針對 Java 程式語言的開發人員發行的 SDK (Software Development Kit)，自從 2006 年之後，Sun 公司宣佈基於 GPL 協議使其成為自由軟體。

目前下載點放在：<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

如下畫面：

The screenshot shows the Oracle Java SE Downloads page. On the left sidebar, there's a list of Java products: Java SE, Java EE, Java ME, Java Support, Java Advanced & Suite, Java Embedded, JavaFX, Java DB, Web Tier, Java Card, Java TV, New to Java, Community, Java Magazine, and Java Advanced. In the center, there's a section titled "Java SE Downloads" with four download buttons: "Java Platform (JDK) 7u21", "JavaFX 2.2.21", and "JDK 7 + NetBeans". Below these buttons, there's a "Java Platform, Standard Edition" section with a "Java SE 7u21" link. To the right, there's a "Java SDKs and Tools" sidebar with links like Java SE, Java EE and Glassfish, Java ME, JavaFX, Java Card, and NetBeans IDE. At the bottom right, there's a "Java Magazine" banner.

點擊「Java Platform (JDK)」進入如下畫面：



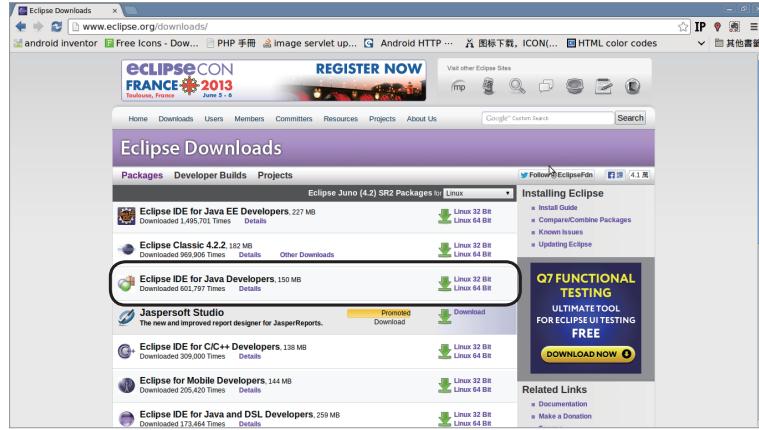
1

就可以依照開發作業系統平台，下載適當地版本進行安裝程序。

1-3 安裝設定 Eclipse

Eclipse 是開發 Android App 的主要整合開發工具。所謂的整合開發工具的意思，就是將開發過程中，從基本的程式編輯器開始，除錯工具、Log 紀錄、模擬器等等全部具備。開發者只要開始執行 Eclipse 之後，全部所需要的開發資源大部分都具備了，是相當方便的開發利器。

先至官方網站：<http://www.eclipse.org>，找到 Download Eclipse 連結過去，看到如下網頁：



找到「Eclipse IDE for Java Developer」該項目下載。下載回來的檔案只需要進行解壓縮到自己喜歡的特定目錄下即可，無須安裝程序。

就在解壓縮後的目錄下找到「eclipse」，跑起來吧！



當詢問使用的工作區路徑的時候，此時只是決定這次執行預設使用的工作區路徑，可以另外指定或是沿用詢問的預設值，無論如何，重點是要知道開發的專案放在什麼路徑下即可。

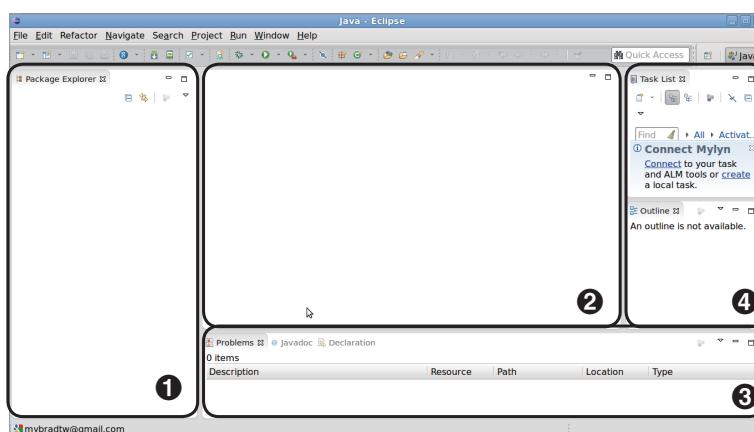


1-4 安裝設定 Eclipse

首次來到 Eclipse 的整合開發環境，看到以下歡迎的畫面。



直接將該頁籤關閉吧！看到以下會令首次接觸 Eclipse 開發者不知所措的畫面。

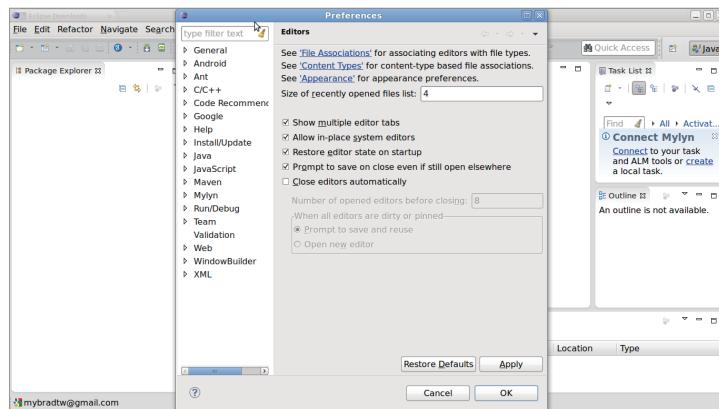


簡要說明：

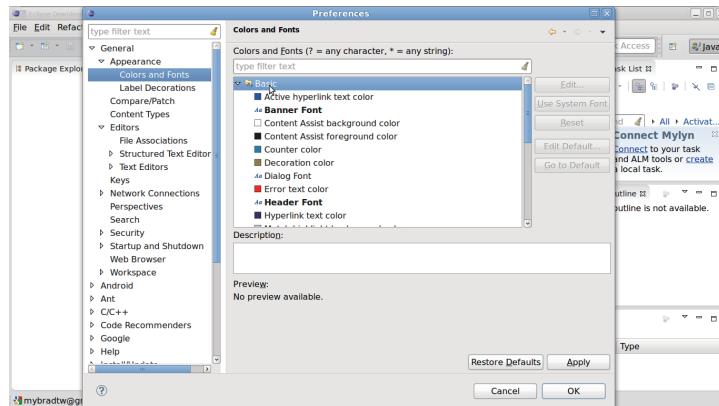
- ① 左側有個 Package Explorer 區，用來顯示目前工作區下管轄的專案列表，及處理專案架構下的檔案目錄，非常重要。
- ② 中間空白區域就是平常開發程式的編輯器。
- ③ 下方有其他輔助開發的視窗，以個別頁籤進行切換。
- ④ 右側兩個暫時不需要，可以直接關閉。

以下簡略介紹基本的設定項目

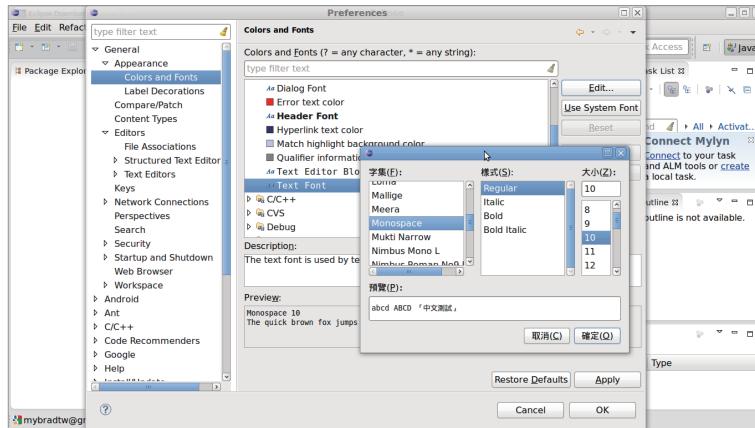
設定程式編輯器字體放在上方選單列「Window → Preference」



左側內容視安裝外掛而定，找到 General 後點擊展開，「Appearance → Colors and Fonts」，看到中間視窗的 Basic 展開 ...

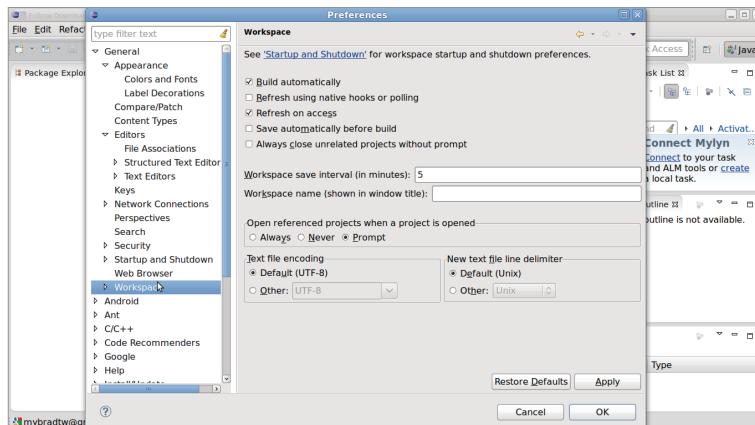


找到「Text Font → Edit...」就可以設定編輯器的字體。



1

再來設定編輯程式碼的文字編碼，建議讀者使用 UTF-8。「General → Workspace」

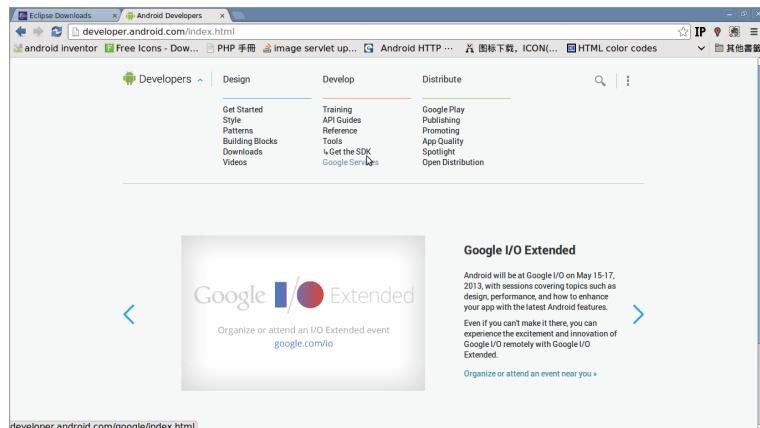


1-5

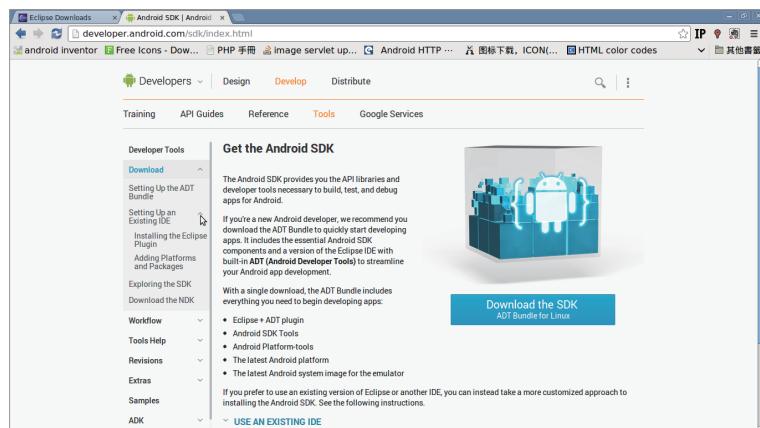
安裝設定 Android SDK

在 Eclipse 外掛 ADT |

先到 Android 開發者官方網站：<http://developer.android.com>



點按「Design → Tools → Get the SDK」後，看到如下網頁：



在左側的「Setting Up an Existing IDE」下的「Installing the Eclipse Plugin」後看到如下網頁內容：

Installing the Eclipse Plugin

Android offers a custom plugin for the Eclipse IDE, called Android Development Tools (ADT). This plugin provides a powerful, integrated environment in which to develop Android apps. It extends the capabilities of Eclipse to let you quickly set up new Android projects, build an app UI, debug your app, and export signed (or unsigned) app packages (APKs) for distribution.

If you need to install Eclipse, you can download it from eclipse.org/mobile.

Note: If you prefer to work in a different IDE, you do not need to install Eclipse or ADT. Instead, you can directly use the SDK tools to build and debug your application.

Download the ADT Plugin

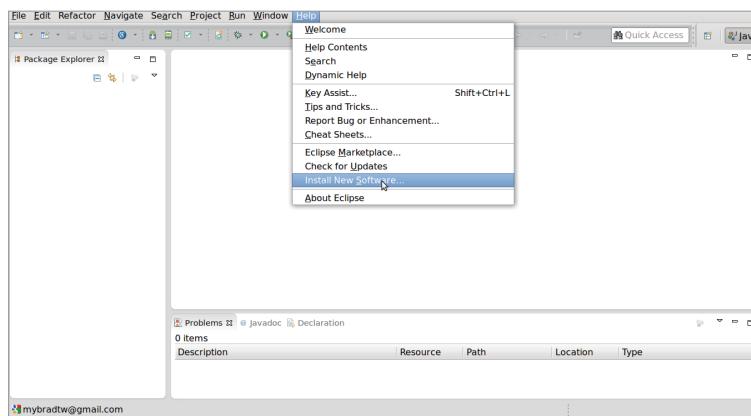
1. Start Eclipse, then select Help > Install New Software.
2. Click Add... in the top-right corner.
3. In the Add Repository dialog that appears, enter "ADT Plugin" for the Name and the following URL for the Location:
<https://dl-ssl.google.com/android/eclipse/>
4. Click OK.
- If you have trouble acquiring the plugin, try using "http" in the Location URL, instead of "https" (https is preferred for security reasons).
5. In the Available Software dialog, select the checkbox next to Developer Tools and click Next.
6. In the next window, you'll see a list of the tools to be downloaded. Click Next.

1

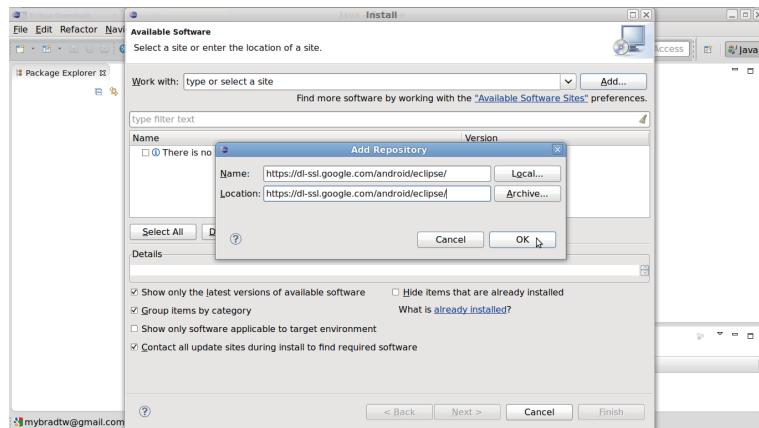
將其網頁內提及 Download the ADT Plugin 中的網址：

<https://dl-ssl.google.com/android/eclipse/>

複製起來，回到 Eclipse 中，在選單列中「Help → Install New Software...」



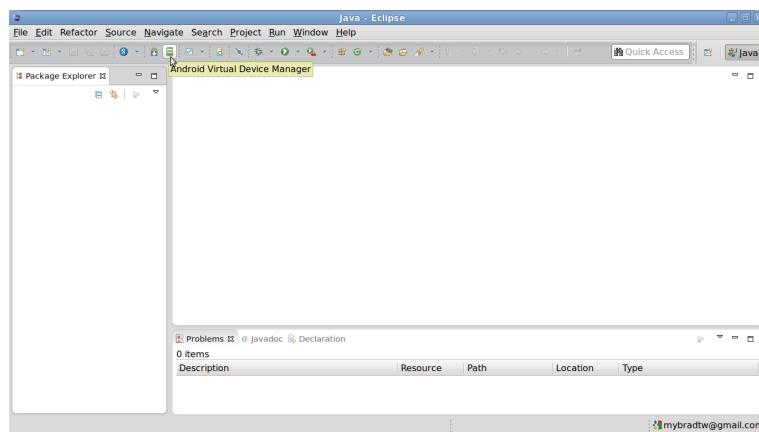
出現的對話框中，按下 Add 後，將剛才複製的網址貼上 ...



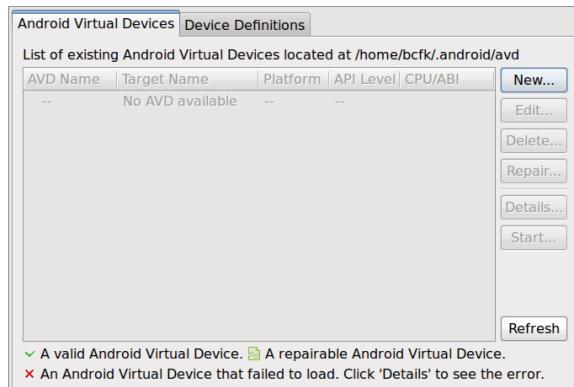
這樣就開始進行 Android ADT 外掛安裝，相當容易。

建立及使用模擬器

安裝之後，在工具列中找到如下圖中指標所示：

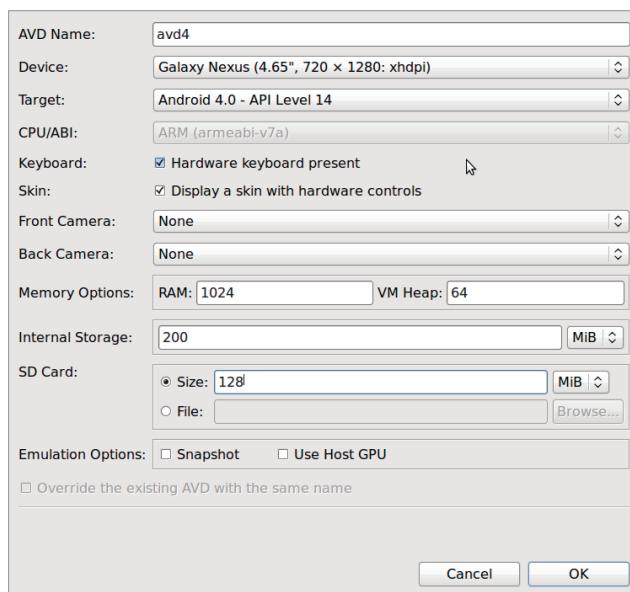


Android Virtual Device Manager 用來進行模擬器的建立與啟動相關管理工具，按下之後：



1

接著按下對話框右側的 New... 用來建立模擬器。

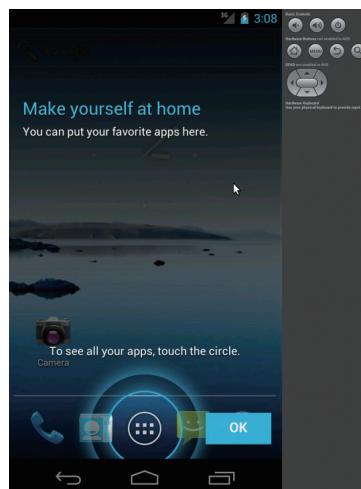


在以上對話框中輸入想要建立模擬器的基本資料：

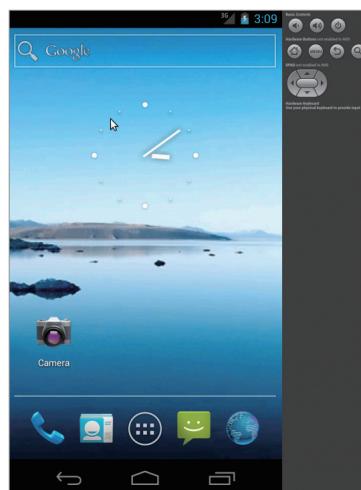
- AVD Name：自訂名稱
- Device：設定想要模擬的裝置
- Target：安裝 Android 的版本
- Keyboard：務必勾選 Hardware keyboard present

- Skin：務必勾選 Display a skin with hardware controls
- Front Camera：如果有安裝 Cam 的讀者可以考慮使用
- SD Card：請適量輸入模擬 SD Card 的空間大小

按下「OK」後啟動看看。(可能會需要稍微久的時間進行啟動，請耐心等候...)



按下螢幕上的「OK」，出現如下：



其他部份就不一一贅述，就是開始玩這隻手機囉。

02

Chapter

基本程序運作原理與應用

- 2-1 「Hello, World? Hello, Lottery!」
- 2-2 「BMI?Lottery!」
- 2-3 寫完了，然後呢？
- 2-4 Activity 的生命週期
- 2-5 Activity 切換 Activity
- 2-6 Service 的運作應用



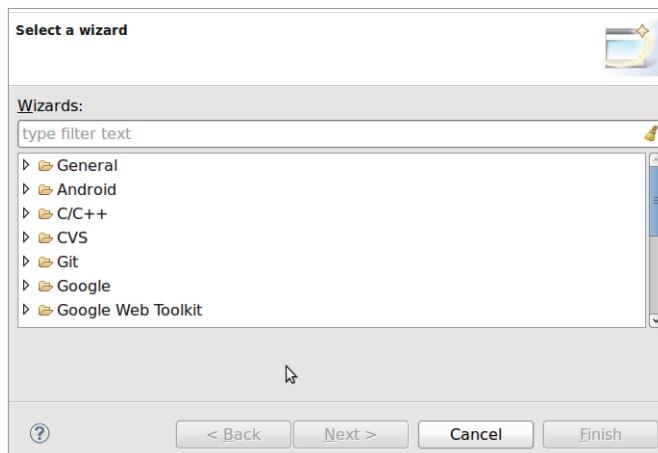


2-1 「Hello, World? Hello, Lottery!」

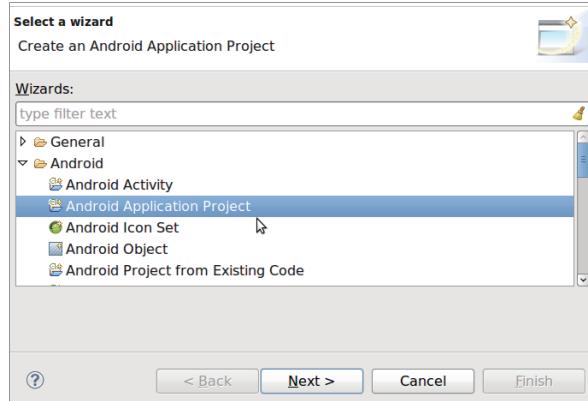
「Hello, World」專案的開發，相信讀者可以在一百本的 Android 相關書籍中看到。因此，筆者不打算浪費有限的篇幅來介紹，直接一開始就來寫一個「Hello, Lottery!」，沒有錯！就是利用電腦選號來產生樂透號碼的程式，程式開發邏輯上非常簡單，重點放在整個開發專案架構，以及一個 App 的執行生命週期的相關學習。事實上，讀者只需要略作些許修改，就可以堂而皇之的放在 Google play 上面發佈，而目前約略有一百零一個樂透號碼產生器的 App 在 Google play 上面賺取廣告費喔。

建立新專案

先進行專案的建立，點選視窗選單列的選項「File」→「New」，顯示子選單中如果有出現「Android Application Project」，那就點選下去；但是大部份一開始初始建置開發環境之後，應該還不會出現在這個子選單，所以請在這個子選單中點選「Other...」選項，之後將會出現一個對話框如下圖：



實際畫面不一定與筆者相同，其內容會因 Eclipse 中所外掛安裝的軟體而有所不同，重點是找出 Android 的資料夾，並將其點選展開後如下圖，點選「Android Application Project」，並按下下方的「Next >」按鈕。



2

正式開始新專案的建立，首先先針對新專案的名稱來處理。有以下幾個重要項目填寫：

- Application Name（應用程式名稱）：也就是使用者看到的名稱。
- Project Name（專案名稱）：是開發者自行定義的名稱，通常會與 Application Name 相同，但是為了使讀者了解兩者的差異性，故意寫成兩個不同的內容。
- Package Name（軟體名稱）：是這套軟體安裝在使用者裝置下的識別名稱。使用者可能會在其行動裝置上，裝了來自於世界各地開發者的軟體，而這些軟體是以 Package Name 識別方式來存放這些不同的開發軟體。通常在 Java 程式專案開發中，慣例是以開發者的公司組織學校，或是個人的專屬網域名稱倒過來命名。例如筆者有一個開發測試網域為 ez2test.com，則開發 Android 的遊戲類的「炸彈王 (BombKing)」軟體，可能在此處就是：com.ez2test.android.games.bombking

因此，目前的「Hello, Lottery」專案資料如下：

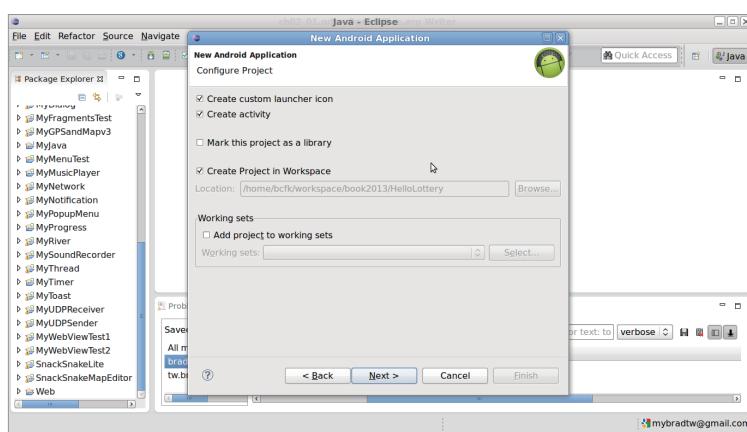
- Application Name：億萬富翁大樂透
- Project Name：HelloLottery
- Package Name：tw.brad.android.book.hellolottery
- 其他下面的四個項目，就先以預設值為主，先不進行處理。

這些項目都可以在專案建立之後再做修改調整，包含上述的三個命名項目。

如下圖所示：

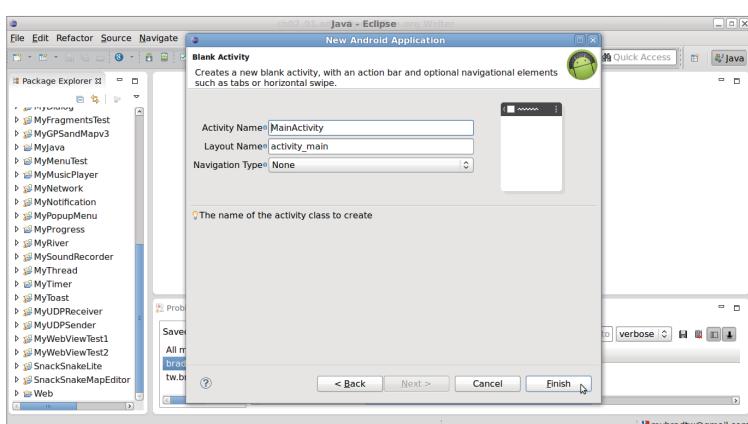
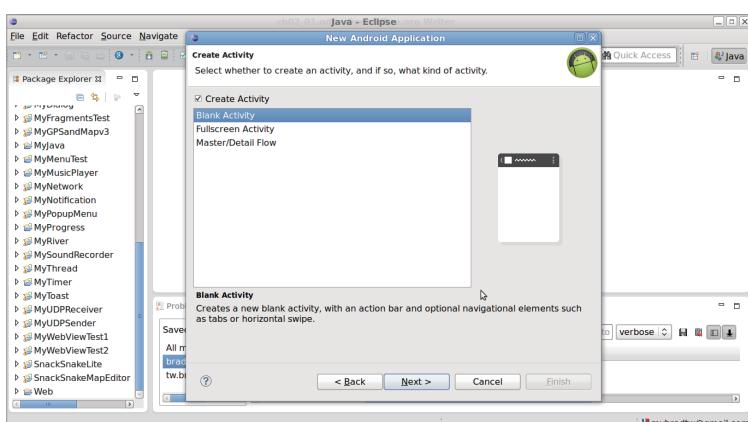


按下「Next >」按鈕之後的對話框就都可以直接按下「Next >」按鈕跳過，一直到可以按下「Finish」，新專案就建立完成。成果如下列圖示程序：



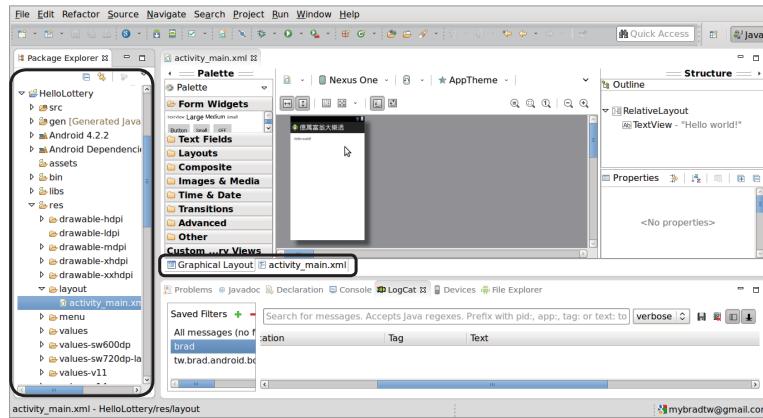


2



當按下「Finish」按鈕之後，將會開始自動產生新專案的架構，可能會需要稍候數秒時間，請耐心等候，千萬不要以為當機或是其他不可預期狀況發生。(暫時閉上雙眼休息一下，等一下有更長的路要走)。

終於等到以下畫面出現：



左邊就是專案目錄架構，Project Name 就出現在此，以下介紹一開始必須先認識的子目錄。

- src/：開發的程式
- res/：應用程式使用到的相關資源
- drawable-xxx/：應用程式專案的影像圖形檔案
- layout/：應用程式的版面配置檔案，就是呈現的外觀部份
- values/：應用程式專案使用的資料值

版面配置

目前就是停留在版面配置檔案 `activity_main.xml` 上面，使得右邊開發視窗要呈現該檔案的內容，可以在該視窗下方看到兩個頁籤

- Graphical Layout：以視覺化方式來規劃配置版面（筆者建議用來參考觀看用）
- `activity_main`：以 XML 文件格式來規劃配置版面（筆者強烈建議開發使用）

因此，請點按下「activity_main」的頁籤，將會出現如下 XML 文件格式內容如下：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

2

千萬別被內容複雜給嚇到，先以架構來分析：

- <RelativeLayout>
- <TextView />
- </RelativeLayout>

就這樣而已，最外層定義了一種版面配置方式，其名稱為 `RelativeLayout`（其實就是 Java 的類別名稱，通常有個命名特性，就是駝峰式命名，首字母大寫，其他小寫，如果是複合字，就很像駱駝的駝峰囉）。該類別物件是具有容器特性，也就是說可以裝進其他顯示元件，因此，會另外有相對應的結尾標籤 `</RelativeLayout>`，前置斜線符號字元，也就是有頭有尾的完整結構。而目前裝載了一個 `TextView` 的顯示元件，用來顯示一般的文字內容，並非容器特性的顯示元件，雖然也可以有頭有尾的加上 `</TextView>`，但是通常避免麻煩，就直接在最後加上斜線符號字元 `<Xxx />` 的方式，有頭無尾自我了結。

而其中會有一些 `android:xxx="xxx"` 的項目，稱其為屬性設定。等號左邊為屬性項目，等號右邊為設定值，例如：

```
    android:layout_width="wrap_content"
```

表示屬性設定 android:layout_width 該元件在版面寬度上，設定為 "wrap_content" 依照內容決定其顯示寬度。注意一點就是設定值一定被包在雙引號之中，因為 XML 是文件格式，所有資料都是字串型態。

至此，先簡單修改如下：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >

    <Button
        android:id="@+id/torich"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="致富按鈕" />

    <TextView
        android:id="@+id/richnum"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_vertical|center_horizontal"
        android:text="此處將會出現致富號碼 ..." />

</LinearLayout>
```

往下閱讀學習過程中，請記得筆者建議的口訣，無論是本書或是其他文章學習，請不要將不認識的內容複製貼上到你的開發專案中，那可能會造成你處理上不必要的麻煩，甚至於是錯誤。所有自行開發的程式內容都是自己可以掌握的東西。

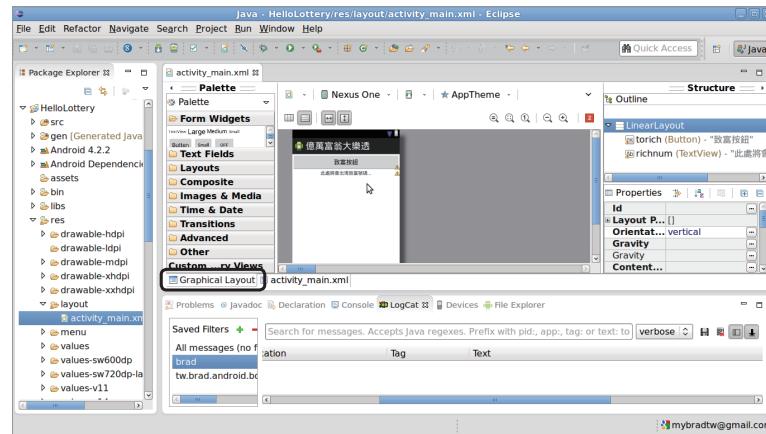
所以，以下先就目前簡單的內容作說明。

- LinearLayout 是一種版面配置，算是最簡單的版面配置方式，只有兩種模式，設定在屬性為 android:orientation 項目中。可以先將該屬性項目的設定值清除後，按下「Alt」+「/」輔助輸入
- "horizontal"：由左至右的水平排列容器中的元件

- "vertical"：由上至下，垂直排列容器中的元件
- xmlns:android：表示以下的 XML 命名空間，通常設定在最頂端的元件即可，其他元件無須再做重複設定，除非有所不同。
- android:layout_width：各元件的版面配置寬度（大多數的元件都必須設定的項目）
- fill_parent：填滿該元件之父容器
- match_parent：符合該元件之父容器（與 fill_parent 相同）
- wrap_content：依照該元件內容決定
- android:layout_height：各元件的版面配置高度（大多數的元件都必須設定的項目）
- Button 是一種顯示元件，就是顯示一個按鈕的元件，其實只是視覺上的按鈕，骨子裡和 TextView 沒有兩樣。在 Android 中的顯示元件，全部都可以有按下的事件處理，不一定是按鈕才有。
- TextView 用來顯示一般文字資料內容
- android:id：為該顯示元件設定整個專案中的唯一識別名稱，其格式為 "@+id/ 識別名稱 "，命名原則與 Java 變數相同，不可以是關鍵字或是保留字，而且 [a-zA-Z\$_][a-zA-Z0-9\$_]*，首字母可以是 a-zA-Z 或是 \$, _ 字元，第二個字元之後可以多了數字使用。
- android:text：該元件的顯示字串內容
- android:gravity：資料內容排列原則，常見如下：
 - center_horizontal：水平置中
 - center_vertical：垂直置中
 - center_vertical|center_horizontal：同時垂直水平置中

修改過程中，隨時按下「Ctrl」+「s」進行儲存；按下「Ctrl」+「Shift」+「f」可以將文件以縮排架構排列。

查看目前的畫面配置狀況，點按「Graphical Layout」的頁籤，如下圖所示：



開發程式

點按左邊專案視窗中，「HelloLottery/ → src/ → tw.brad.android.book.hellolottery/ → MainActivity.java」，中間出現以下已經打好的程式碼如下：

```
package tw.brad.android.book.hellolottery;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        // Inflate the menu; this adds items to the action bar if it is present.

        getMenuInflater().inflate(R.menu.main, menu);

        return true;
    }
}
```

還是這句話：不認識的不要放在自己的專案中，先修改成如下：

```
package tw.brad.android.book.hellolottery;

import android.os.Bundle;
import android.app.Activity;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

    }
}
```

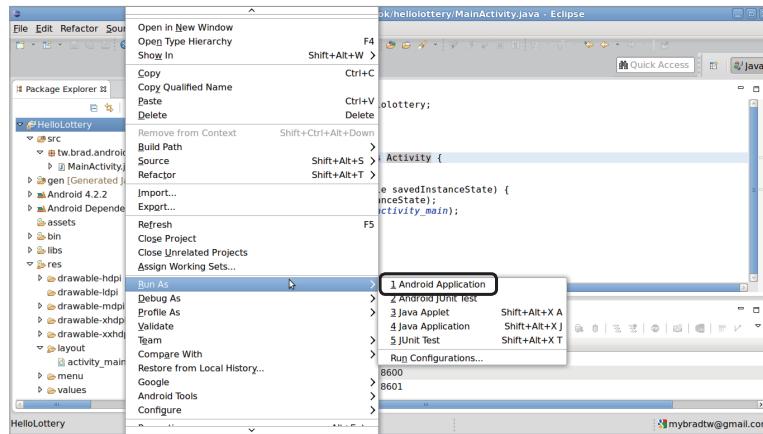
2

說明如下：

- package ... : 定義該 Android 程式所屬的 Package Name (通常不需要處理)
- import ... : 定義該 Android 程式中所需要的 API 的 Package Name (通常不需要處理，可以在一邊開發中，按下「Ctrl」+「Shift」+「o」自動處理即可)
- public class MainActivity extends Activity {...} , 定義自訂的 Android 應用程式的類別名稱，所有在應用程式中使用者看到的部份，都是繼承自 Activity，所以會有 extends Activity 的定義；{...} 程式區塊則為該類別的詳細定義。
- @Override , 用來說明以下定義的方法是改寫父類別的方法。
- protected void onCreate(){} , 用來開發撰寫生命週期的第一個階段應該要處理的程序內容。
- super.onCreate() , 用來呼叫父類別定義的生命週期的第一個階段程序。(必須)
- setContentView(R.layout.activity_main) , 則是開始進行自行開發的內容中，不同的內容畫面，指定前面所規劃的版面內容。

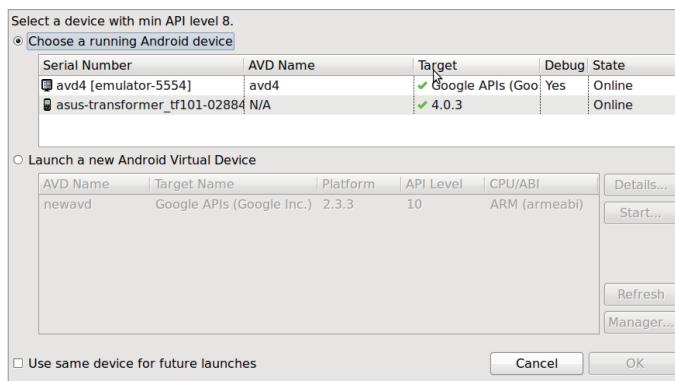
■ 安裝執行測試 ■

迫不及待的執行看看吧，執行方式可以如下圖示中，按下「Android Application」。



如果只有連接一個實體裝置或是一個已經啟動完成的模擬器，則將會直接安裝並執行到該裝置上。

如果已經連接多個實體裝置，或是一個實體裝置及一個模擬器，則將會出現類似以下對話框，選擇安裝執行的特定裝置：



筆者點選模擬器之後，看到模擬器的狀況如下：



至此，算是到了「Hello, World」的境界了。

2-2 「BMI?Lottery!」

這兩年開課教授 Android 課程，遇過三次類似的詢問：

- 老師，你會教如何寫 BMI 的 Android 程式嗎？
- 課程規劃中怎麼沒有教 BMI 的 Android 程式？
- 我們學校老師都先教 BMI 的 Android 程式喔。

我當然知道這是國內一本知名的 Android 書中的重要入門範例，用來練習與使用者輸入互動的部份。因為作者寫得非常好，仔細研讀一定可以學習到，如果我上課還要教一遍，豈不是侮辱各位學生的智慧嗎？而今寫這本書，如果還寫 BMI，豈不是侮辱各位讀者您的智慧，還侵犯之前作者的智慧。因此，延續上一小節專案「Hello, Lottery」，將其完整的進行後續開發 ...

存取控制元件

專案中有兩個元件 Button 和 TextView。使用者將會按下 Button 之後，出現一組樂透號碼（然後買了一張，中了頭獎，從此過著幸福快樂的生活 ...）。所以首要之務就是將兩個元件的參考指標找出來。

先在 MainActivity 的類別中定義屬性成員：

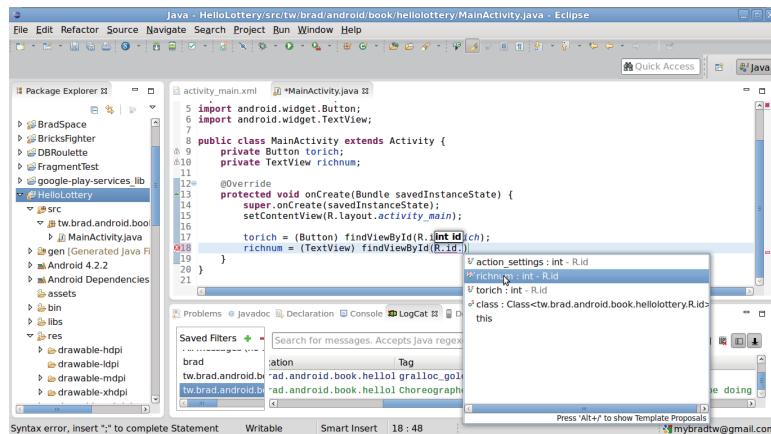
```
private Button torich;
private TextView richnum;
```

並在 onCreate() 方法中，setContentView() 敘述句之後，以 findViewById() 方法呼叫，傳回其物件參考。從邏輯性來看，因為先有 setContentView()，整個版面內容設定之後，再從其中找出顯示元件（所有顯示元件都是 View，也就是 is-a View）。如果顛倒順序，從 java 語言文法而言沒有錯誤，卻將會在執行階段拋出 Exception 而中斷執行，這就是出現邏輯錯誤所造成。

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    torich = (Button)findViewById(R.id.torich);
    richnum = (TextView)findViewById(R.id.richnum);
}
```

如果讀者在輸入 R.id. 之後，稍候一下，應該會出現選單，可以點選出當時在版面配置上所賦予的 android:id 的設定值，如果沒有出現的話，有一種可能性就是尚未在版面配置檔案編輯中存檔，以致於專案架構中尚未配置出該元件的物件參考指標資源。養成習慣隨時編輯，隨時儲存檔案內容「Ctrl」+「S」。



按鈕事件處理模式

按下 Button 之後的處理程序，就是為該 Button 元件設定按下事件的監聽物件 (XxxListener)，而由該監聽物件來處理按下之後該做的事。

```
torich.setOnClickListener();
```

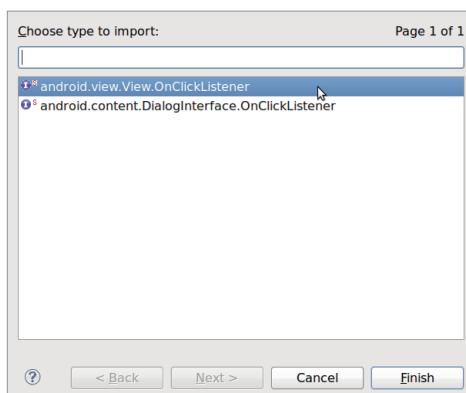
將會暫時出現底下紅色波浪底線，表示語法錯誤，因為尚未傳遞所需要的監聽物件參數。

增加其參數內容如下：

2

```
torich.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
    }
});
```

仍有錯誤來自於尚未 import，按下「Ctrl」+「Shift」+「o」，自動 import 處理。



目前的監聽物件是 View 的 OnClickListener，而不是 DialogInterface.OnClickListener，別選錯了。觀念就是 Button 是 View 元件，當然是以 View 的監聽物件來負責監聽觸發事件。後面單元還會為讀者介紹對話框的監聽物件，那個時候就是使用 DialogInterface.OnClickListener 為其監聽物件。

而在其中的 `onClick()` 方法中，開發撰寫當使用者按下按鈕之後的處理程序。首先，先另外定義出處理的自訂方法，筆者的開發模式比較不喜歡全部塞在一起，容易造成維護不易的困擾，所以另外定義出自訂方法來處理，如下：

```
package tw.brad.android.book.hellolottery;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends Activity {
    private Button torich;
    private TextView richnum;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        torich = (Button) findViewById(R.id.torich);
        richnum = (TextView) findViewById(R.id.richnum);

        torich.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                // 此處呼叫使用 createLottery() 方法
                createLottery();
            }
        });
    }

    // 該方法用來產生樂透號碼
    private void createLottery() {
    }
}
```

開發設計功能

好好專心來寫 createLottery() 方法吧 ...

```
// 該方法用來產生樂透號碼
private void createLottery(){
    HashSet<Integer> set = new HashSet<Integer>();
    while (set.size() < 6) {
        set.add((int)(Math.random() * 49 + 1));
    }

    richnum.setText("");
    Iterator<Integer> iterator = set.iterator();
    while (iterator.hasNext()) {
        int num = iterator.next();
        richnum.append(num + " ");
    }
}
```

2

說明如下：

- HashSet 用來宣告一個可以存放資料的資料結構，利用資料不重複的特性（樂透號碼也不會重複出現），並且泛型 Integer。
- while() 迴圈偵測資料要是小於 6 個號碼，繼續執行到有 6 個不重複的號碼。
- Math.random() 會傳回大於或等於 0.0 與小於 1.0 之間的亂數，將該數乘以 49 加 1 後，其範圍為大於或等於 1.0 到小於 50.0（最大就是 49.999...）之間的浮點數，再經過強制轉型為 int 後就是介於 1 到 49 之間的亂數了。
- 將 TextView 物件變數 richnum 呼叫 setText("")，將顯示內容清除處理。
- HashSet 物件實體 set 呼叫 iterator() 方法傳回 Iterator 物件實體，用來將資料內容依序取出使用。
- Iterator 的 hasNext() 方法傳回 boolean 值表示是否還有資料存在。
- Iterator 的 next() 方法傳回資料內容。
- 最後由 TextView 的物件實體 richnum 呼叫 append() 方法將資料一個一個連接出來。

執行看看囉 ...



當然，因為是亂數隨機產生的結果，所以千萬不要因為執行結果與左圖不一樣，就開始寫電子郵件 brad@brad.tw 問候筆者喔。如果，因此而中了大獎，那歡迎來信告知將會分給筆者一半獎金的消息，我應該不會拒絕您的善意。

修改程式

開始有使用者的抱怨聲音出現，出現的號碼沒有排序，拿去彩券行不好簽注 ... 好，要用力傾聽使用者的使用經驗，並進行改善。如果，還要翻出當年差點被當掉的資料結構聖經來看的話，那可能下次改版時間將會是明年。此時善加運用 Java 中的 TreeSet 吧。

```
// 該方法用來產生樂透號碼
private void createLottery() {
    TreeSet<Integer> set = new TreeSet<Integer>();
    while (set.size() < 6) {
        set.add((int) (Math.random() * 49 + 1));
    }

    richnum.setText("");
    Iterator<Integer> iterator = set.iterator();
    while (iterator.hasNext()) {
        int num = iterator.next();
        richnum.append(num + " ");
    }
}
```



就不再重複說明了，我知道善良真誠的讀者，您會知道知恩圖報的。

只有將 HashSet 改成 TreeSet 而已，看看結果吧 ...





2-3 寫完了，然後呢？

幸福了，然後呢？當然是繼續幸福囉。那寫完了，然後呢？包裝整理成為完整的 App 吧！

加上歡迎畫面

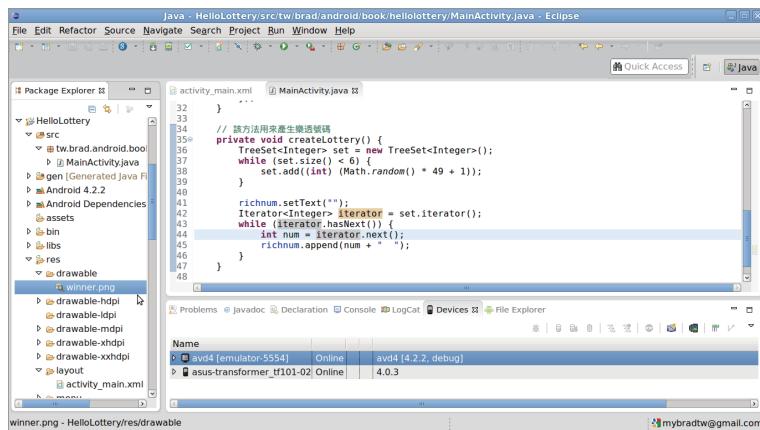
"別人正式發佈的 App，好像都有一個感覺還不錯的歡迎畫面，或是來一個開發公司的標記..."

那就來吧！先請專業的美工，不是美工，應該是視覺創意達人，設計了勝利標章。

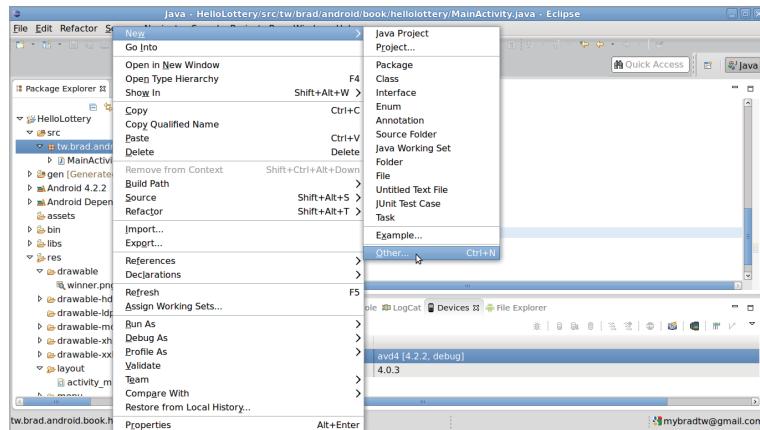


2

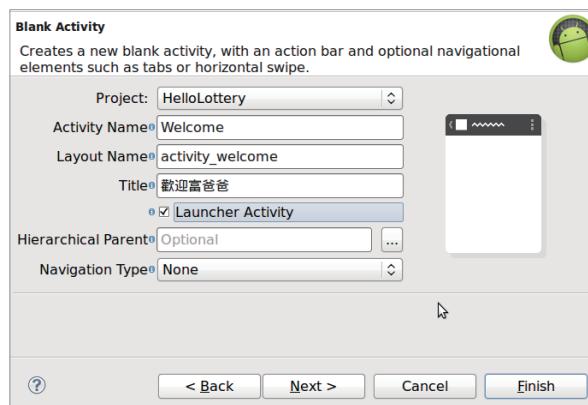
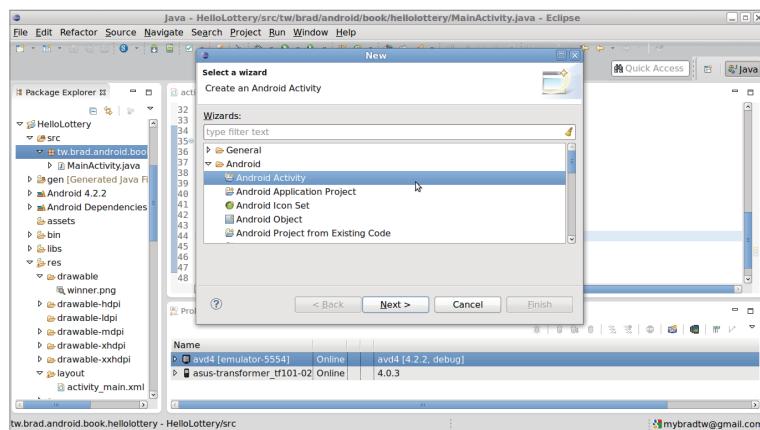
利用準備好的圖檔 `winner.png` 來設計歡迎畫面，先將專案目錄下 `res/` 目錄下建立出一個子目錄，名稱為 `drawable`（全小寫，請先按照我的命名 `drawable`）。並將圖檔放在該目錄下。



接著在 Package Name 下，按右鍵點出開啟建立 Activity 的對話框。



以下能直接按下「Next」按鈕就按下吧。



- Activity Name：輸入歡迎畫面的名稱
- Layout Name：會自動產生，當然也可以變更
- Title：是顯示在畫面上方的文字內容

看見可以使用「Finish」按鈕，那就接下去囉！

一樣的先設計處理版面：activity_welcome.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#fffff00"
    >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_alignParentTop="true"
        android:text="億萬富翁大樂透"
        android:textColor="#0000ff"
        android:textSize="36sp"
        android:textStyle="bold|italic" />
    <ImageView
        android:id="@+id/weinner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:layout_centerHorizontal="true"
        android:src="@drawable/winner" />
</RelativeLayout>
```

2

說明如下：

- RelativeLayout 這次採用相對是的版面配置，配置元件方式都是採用相對關係。
- 在 TextView 中的 layout_centerHorizontal 就是相對於父容器元件的水平置中。
- 在 TextView 中的 layout_alignParentTop 就是說明其配置的位置是沿著父容器元件的頂端。

- RelativeLayout 中的 background 用來指定背景處理，#ffff00 則是以 HTML 的顏色設定處理 RGB，目前顯示紅色加上綠色產生黃色作為背景顏色。
- ImageView 中的 src 則是用來指定呈現剛剛放在 res/drawable/ 下的 winner.png 的圖檔。

|| 調整啟動程序 ||

修改目前的專案目錄下的 AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="tw.brad.android.book.hellolottery"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="tw.brad.android.book.hellolottery.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="tw.brad.android.book.hellolottery.Welcome"
            android:label="@string/title_activity_welcome" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

因為一開始只有 MainActivity 為啟動專案的類別，因此後來加上的 Welcome 要成為啟動應用程式的類別，可以看到各自文件架構下，都有以下內容：

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />

    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

拿掉 MainActivity 中的這個部份，整個改成以下結構：

2

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="tw.brad.android.book.hellolottery"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="tw.brad.android.book.hellolottery.MainActivity"
            android:label="@string/app_name" >
        </activity>
        <activity
            android:name="tw.brad.android.book.hellolottery.Welcome"
            android:label="@string/title_activity_welcome" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

開發撰寫 Welcome.java

```

package tw.brad.android.book.hellolottery;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ImageView;

public class Welcome extends Activity {
    private ImageView winner;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_welcome);

        winner = (ImageView)findViewById(R.id.winner);
        winner.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                gotoMain();
            }
        });
    }

    private void gotoMain() {
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
        finish();
    }
}

```

處理結構與產生樂透非常類似，以下說明點按勝利圖片之後的部份：

- Intent 類別物件用來設定呼叫其他 Activity 或是 Service 的物件實體
- 呼叫 `startActivity()`，並傳入設定好的 Intent 物件實體，即可將控制權轉移到 MainActivity 類別物件了
- 之後不再需要看到歡迎畫面，於是呼叫 `finish()`

結果如下：



2



2-4 Activity 的生命週期

一個簡單的專案開發完成後，開始對於 Android 的 App 開發詳細部份進行了解。

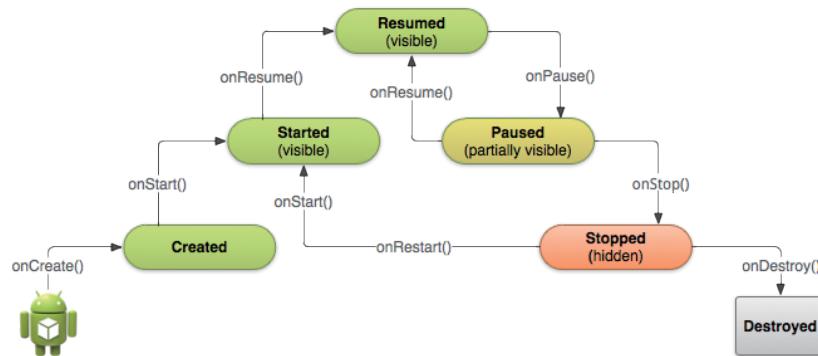
事實上，對於開發過網頁設計並搭配 JavaScript 處理動態效果的讀者，應該感受到異曲同工之處了。處理版面配置就如同設定規劃 HTML 的網頁內容，稍候要程式控制的元素賦予 id；而開發程式部份就如同處理 JavaScript 的部份。甚至於還呼叫 `findViewById()` 的方法，幾乎和 JavaScript 中的 `getElementById()` 沒有兩樣。算是非常容易學習的。

生命週期的觀念

生命週期只是將 Activity 從啟動後，執行、暫停，恢復到結束的過程，看待為一個生命週期。

在 Android App 中的 Activity 扮演的角色，其實已經在前幾小節中看到作用了。只要是使用者看得到的部份，都是由 Activity 負責進行控制處理（如同 JavaScript 負責的部份）。

在 google 的開發者官方網站上有一個非常重要的圖，呈現出一個 Activity 的生命週期：



一般啟動 Activity 的基本程序如下：

- onCreate()
- onStart()
- onResume()

之後就進入到執行狀態。當使用者按下返回鍵之後，會使該 Activity 從執行狀態中，執行以下程序：

- onPause()
- onStop()
- onDestroy()

最後進入到摧毀完畢階段。如果在執行狀態下，啟動其他的 Activity 之後，會使得目前的 Activity 執行以下程序：

- onPause()
- onStop()

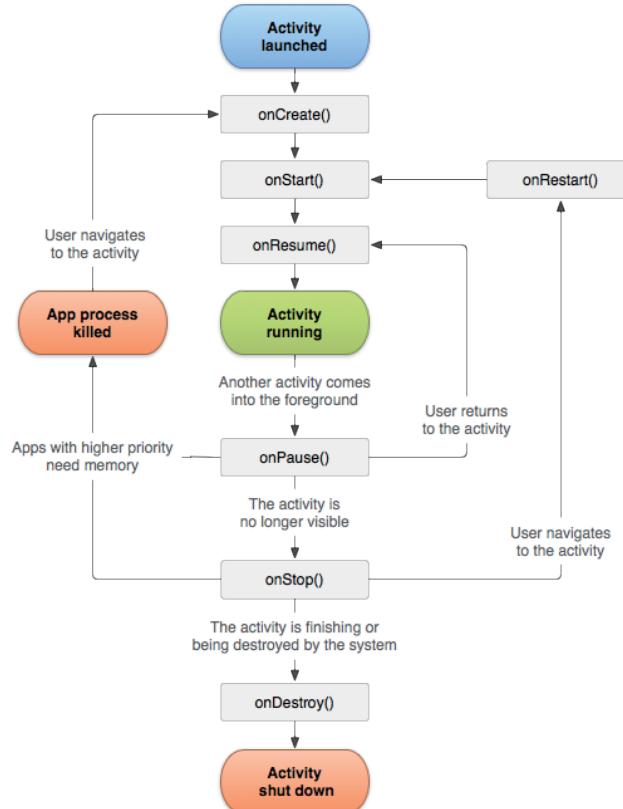
而進入到暫停狀態，當再度被恢復執行時，將會執行以下程序：

- onRestart()
- onStart()
- onResume()

2

再度回到執行狀態中。

從 Activity 的 API 中可以看到另外一張示意圖：



|| 測試實作 ||

可以開發一個簡單的測試專案來實作生命週期：

處理版面配置 res/layout/activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >

    <Button
        android:id="@+id/next_page"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Goto Next Page" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Main Page" />

</LinearLayout>
```

按鈕設計是為了啟動另一個 Activity，此時可以觀察目前 Activity 的運作狀況。

src/MainActivity.java

```
package tw.brad.android.book.helloactivity;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MainActivity extends Activity {
    private Button next_page;

    @Override
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Log.i("brad", "onCreate");

    next_page = (Button)findViewById(R.id.next_page);
    next_page.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(MainActivity.this,
                Page2Activity.class);
            startActivity(intent);
        }
    });
}

@Override
protected void onDestroy() {
    super.onDestroy();
    Log.i("brad", "onDestroy");
}

@Override
protected void onPause() {
    super.onPause();
    Log.i("brad", "onPause");
}

@Override
protected void onRestart() {
    super.onRestart();
    Log.i("brad", "onRestart");
}

@Override
protected void onResume() {
    super.onResume();
    Log.i("brad", "onResume");
}

@Override
protected void onStart() {
    super.onStart();
    Log.i("brad", "onStart");
}

@Override
```

```

protected void onStop() {
    super.onStop();
    Log.i("brad", "onStop");
}

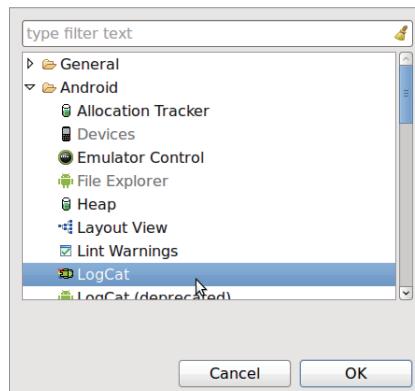
}

```

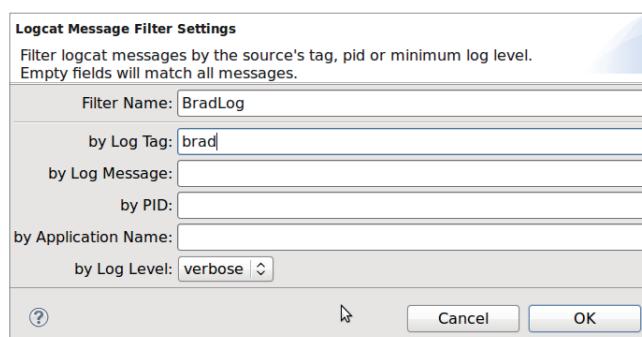
說明如下：

- 在各個 onXxx() 方法中呼叫 Log.i("brad", "Xxx") 以呼叫產生 LogCat 的觀察顯示。
- 按下按鈕啟動下一個 Activity，藉此觀看目前 Activity 的運作狀況。

LogCat 視景如果沒有顯示，可以在 Eclipse 的選單列點選 Window 後下拉選單中選擇「Show View...」→「Other」：



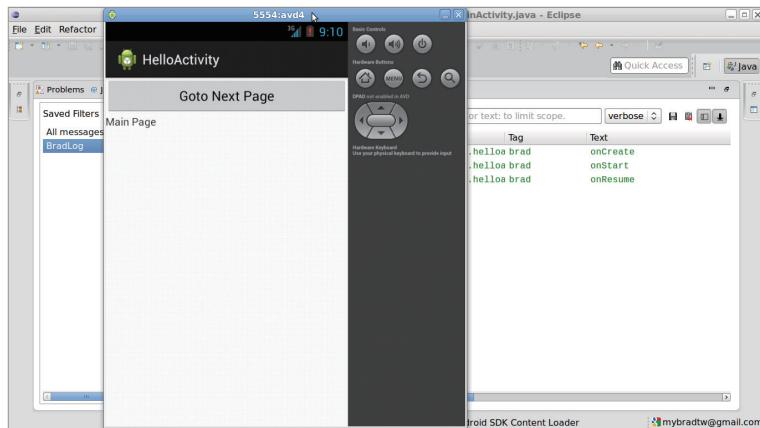
應該就會顯示出 LogCat 的頁籤，點按之後，可以在左側中的加號點按後，輸入要過濾的 Log 資料。



這樣就會過濾出 App 執行中，Log 呼叫傳入字串為 "brad" 的 tag 訊息資料。

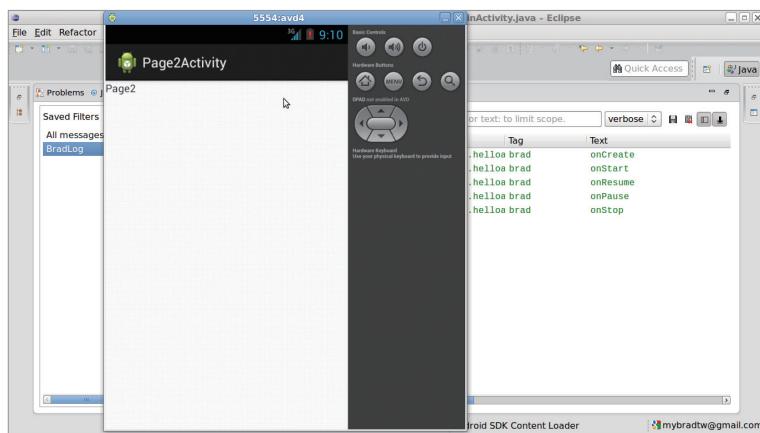
開始觀察

執行啟動之後，看到右邊的 LogCat 紀錄內容。

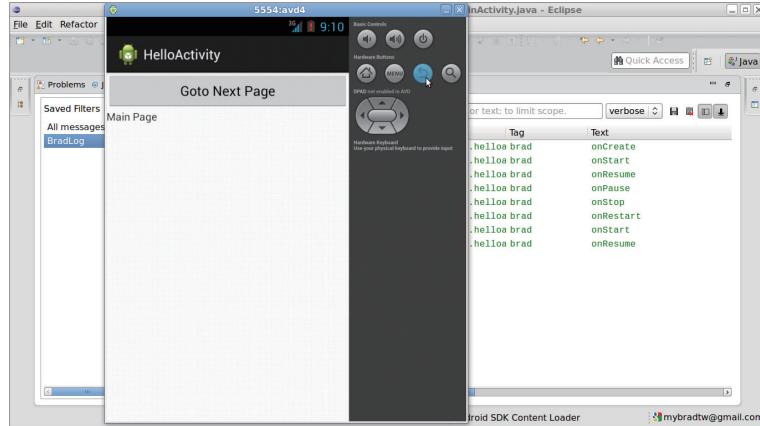


2

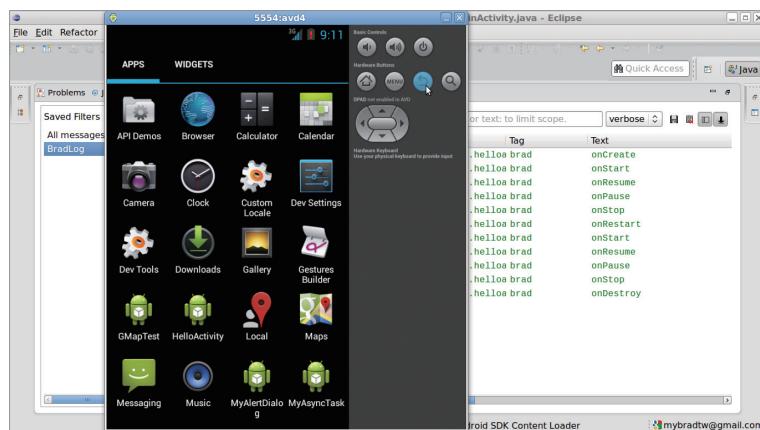
按下按鈕來啟動下一個 Activity。



再從 Page2Activity 按下返回鍵回到 MainActivity。



最後結束 MainActivity。



2-5 Activity 切換 Activity

僅作啟動切換

不同 Activity 之間的切換，已經在前幾小節中認識。就是透過設定 Intent 物件實體，並呼叫 `startActivity()` 方法啟動傳入參數的 Intent 物件實體即可。如下：

```
Intent intent = new Intent(MainActivity.this, Page2Activity.class);
startActivity(intent);
```

■ 傳遞資料過去 ■

當從原來的 Activity 中要將特定的資料傳遞給下一個 Activity 處理或是判斷，則可以透過 Intent 物件實體進行設定。呼叫 putExtra("Key", Xxx 型態資料值) 即可，例如：

```
private void gotoPage2() {
    Intent intent = new Intent(this, Page2Activity.class);
    intent.putExtra("出版", "電腦人");
    intent.putExtra("作者", "趙令文");
    intent.putExtra("isAndroid", true);
    intent.putExtra("price", 10000);

    startActivity(intent);
}
```

2

而在 Page2Activity 中，先呼叫 Activity 的 getIntent() 方法傳回由之前 Activity 所傳入的 Intent 物件實體。再由該物件實體呼叫 getXxxExtra() 方法，傳入指定字串 "Key"，而將其資料值傳回，如果不存在該指定字串 "Key" 的資料，則第二個參數設定其預設值。

```
package tw.brad.android.book.a2a;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;

public class Page2Activity extends Activity {
    private TextView page2_msg;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_page2);

        page2_msg = (TextView) findViewById(R.id.page2_msg);
    }
}
```

```

Intent it = getIntent();
String pub = it.getStringExtra("出版");
String auth = it.getStringExtra("作者");
int price = it.getIntExtra("price", 0);
boolean isAndroid = it.getBooleanExtra("isAndroid", false);

page2_msg.setText("出版社：" + pub + "\n" +
    "作者：" + auth + "\n" +
    "售價：" + price + "\n" +
    "Android：" + (isAndroid?"Yes":"No"));
}

}

```

切換之後回來確認

如果只是單純的一對一，則不用判斷就知道是哪個 Activity 切換回來。而如果是一對多的切換，而回來之後需要知道分別是哪個 Activity 切換回來，並繼續針對差異處理。那麼就必須在啟動切換的時候，改呼叫 `startActivityForResult()`，並設定一個自訂的要求碼（Request Code）。

```

private void gotoPage2() {
    Intent intent = new Intent(this, Page2Activity.class);
    intent.putExtra("出版", "電腦人");
    intent.putExtra("作者", "趙令文");
    intent.putExtra("isAndroid", true);
    intent.putExtra("price", 10000);

    //      startActivityForResult(intent);
    startActivityForResult(intent, 2);
}

private void gotoPage3() {
    Intent intent = new Intent(this, Page3Activity.class);
    //      startActivityForResult(intent);
    startActivityForResult(intent, 3);
}

```

並且同時也改寫 Activity 的 `onActivityResult()` 方法。當由其他 Activity 返回此 Activity 時會執行該方法，傳遞當時過去的 RequestCode 回來，因此

就可以判斷如何區分之後的後續工作。例如照完相片回來之後觀看影像檔案，與錄音完畢之後要聆聽錄音檔案等等，是不同的處理模式。

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == 2){
        msg.setText("Come back from Page2");
    }else if (requestCode == 3){
        msg.setText("Come back from Page3");
    }else {
        msg.setText("Hello, Page");
    }
}
```

2

將資料傳遞回來

再來就是如何將啟動切換回原來的 Activity，也順便將新資料帶回來。處理重點就是在切換過去的 Activity 在結束之前，呼叫 setResult() 方法，透過 Intent 物件實體將資料帶回來，原本的 Activity 也是改寫 onActivityResult() 方法接收參數處理。

可以只傳遞自訂結果碼 Result Code(int 型態)：

```
@Override
public void finish() {
    setResult(RESULT_OK);
    super.finish();
}
```

或是順便透過 Intent 物件實體帶資料：

```
@Override
public void finish() {
Intent it = new Intent();
it.putExtra("data", "OK");
setResult(RESULT_OK, it);

super.finish();
}
```

而回到原來的 Activity 中：

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == 2) {
        msg.setText("Come back from Page2");
        if (resultCode == RESULT_OK) {
            String result = data.getStringExtra("data");
            msg.append("\n" + result);
        }
    } else if (requestCode == 3) {
        msg.setText("Come back from Page3");
    } else {
        msg.setText("Hello, Page");
    }
}
```

一般應用程式大多數是一個以上的 Activity 在運作，因此這小節的內容算是非常基本的工夫。



2-6 Service 的運作應用

Activity 負責處理使用者介面的部份，而 Service 則是負責背景中執行的程序，通常很常見的方式是應用在有週期性的工作任務，例如遊戲中的背景音樂播放，或是每隔一段時間就向遠端伺服器發出詢問要求，當有特定的資料從遠端伺服器傳回行動裝置之後，可能會發出訊息通知。類似這樣的工作程序內容，不需要有任何使用者介面的處理及運作，因此就可以以 Service 的模式進行開發。

因為上述的 Service 的背景運作執行特性，Service 完全無法直接存取操作使用者介面，可以透過 android.os.Handler 類別物件實體來進行存取前景的顯示內容，或是以 Broadcast 的模式發出廣播資料給 Activity 處理。

Service 的生命週期通常會以 Activity 來進行啟動或是綁定，也可以在 Activity 的執行期間結束 Service 的生命週期，但是並不會因為 Activity 的結束生命週期的同時也結束 Service 的生命週期。所以，經常看到在一般行動裝

置中的應用程式，看似已經按下返回鍵結束特定的應用程式，事實上該應用程式可能早就在 Activity 的執行期間已經啟動 Service 的運作執行。所以使用者可能已經結束執行程式，但是等一下剛才明明已經結束離開的應用程式可能會跳出一個通知，告訴你一個好消息："Brad 要出新專輯了"。

Service 不是分開的程序，也不屬於執行緒，但是可以和執行緒共同合作演出。

生命週期實測

2

先直接以一個 Activity 中透過兩個 Button 來負責啟動動 Service，及結束 Service 進行實測。

```
res/layout/activity_main.xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >

    <Button
        android:id="@+id/start"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Start Service"
        />

    <Button
        android:id="@+id/stop"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Stop Service"
        />

    <TextView
        android:id="@+id/msg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />

</LinearLayout>
```

而在 MainActivity.java 中：

```

package tw.brad.android.book.myservicetest;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends Activity {
    private Button start, stop;
    private TextView msg;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        msg = (TextView)findViewById(R.id.msg);

        start = (Button)findViewById(R.id.start);
        start.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                startMyService();
            }
        });
        stop = (Button)findViewById(R.id.stop);
        stop.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                stopMyService();
            }
        });
    }

    // 用來負責啟動 MyService
    private void startMyService(){
        Intent it= new Intent(this, MyService.class);
        startService(it);
    }

    // 用來負責結束 MyService
    private void stopMyService(){
        Intent it= new Intent(this, MyService.class);
        stopService(it);
    }
}

```

啟動 Service 的方式非常簡單，與啟動另一個 Activity 的模式非常類似，還是透過 Intent 的物件實體，設定要啟動的 Service 類別，接著呼叫 startService() 方法，而將設定好的 Intent 物件實體帶入參數即可。

以下是測試生命週期的 Service：

```
MyService.java  
package tw.brad.android.bookmyservicetest;  
  
import android.app.Service;  
import android.content.Intent;  
import android.os.IBinder;  
import android.util.Log;  
  
public class MyService extends Service {  
    public MyService() {  
    }  
  
    @Override  
    public IBinder onBind(Intent intent) {  
        // TODO: Return the communication channel to the service.  
        throw new UnsupportedOperationException("Not yet implemented");  
    }  
  
    @Override  
    public void onCreate() {  
        super.onCreate();  
        Log.i("brad", "onCreate");  
    }  
  
    @Override  
    public int onStartCommand(Intent intent, int flags, int startId) {  
        Log.i("brad", "onStartCommand");  
        return super.onStartCommand(intent, flags, startId);  
    }  
  
    @Override  
    public void onDestroy() {  
        super.onDestroy();  
        Log.i("brad", "onDestroy");  
    }  
}
```

2

開始進行測試了解 ...

第一次按下 Start :

- onCreate()
- onStartCommand()

再度按下 Start :

- onStartCommand()

接著按下 Stop :

- onDestroy()

當第一次啟動 Service 時，會先執行 onCreate()，之後重複呼叫執行 startService() 時，則只會再度執行 onStartCommand()，而不再執行 onCreate()，表示可以透過 startService() 的呼叫傳遞不同的 Intent 資料內容給 Service 進行處理。而當已經 stopService() 之後，Service 會執行 onDestroy()，再度 startService()，則會從 onCreate() 開始執行。

|| 與執行緒共舞 ||

接著在 MyService 中啟動一個簡單的 java.util.TimerTask，以週期性執行特定的工作任務。

```
package tw.brad.android.book.myservicetest;

import java.util.Timer;
import java.util.TimerTask;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;

public class MyService extends Service {
    private Timer timer;
    private MyTask task;
    private int i;
```

```
public MyService() {
    timer = new Timer();
}

@Override
public IBinder onBind(Intent intent) {
    // TODO: Return the communication channel to the service.
    throw new UnsupportedOperationException("Not yet implemented");
}

private class MyTask extends TimerTask{
    @Override
    public void run() {
        i++;
    }
}

@Override
public void onCreate() {
    super.onCreate();
    task = new MyTask();
    timer.schedule(task, 100, 400);
}

@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    i = intent.getIntExtra("i", -1);
    return super.onStartCommand(intent, flags, startId);
}

@Override
public void onDestroy() {
    super.onDestroy();
    timer.cancel();
}
}
```

2

當按下 Start 後，背景中啟動了 Service，並且開始 Timer 的週期任務執行 MyTask 物件實體，表示已經開始運作執行，而當再度重複按下 Start，則會變更目前 Service 物件實體的屬性的 i 值，週期運作狀況照舊，直到按下 Stop 後取消 Timer 的所有週期任務的執行。

目前看到的是前景操控背景的 Service。

透過 Broadcast 發送資料給前景

先在 Service 中的適當時機，將資料放在一個 Intent 物件實體，再呼叫 sendBroadcast() 方法將該 Intent 物件實體廣播出去。而 intent 物件實體會設定一個字串 Action (範例中的 "fromMyService")。等一下在接收 Broadcast 時會用此 Action 字串進行 Intent 的過濾機制。

```
MyServview.java
package tw.brad.android.book.myservicetest;

import java.util.Timer;
import java.util.TimerTask;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;

public class MyService extends Service {
    private Timer timer;
    private MyTask task;
    private int i;
    private Intent it2a;

    public MyService() {
        timer = new Timer();
        it2a = new Intent("fromMyService");
    }

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        throw new UnsupportedOperationException("Not yet implemented");
    }

    private class MyTask extends TimerTask{
        @Override
        public void run() {
            i++;
            it2a.putExtra("i", i);
            sendBroadcast(it2a);
        }
    }

    @Override
    public void onCreate() {
        super.onCreate();
    }
}
```

```

        task = new MyTask();
        timer.schedule(task, 100, 400);
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        i = intent.getIntExtra("i", -1);
        return super.onStartCommand(intent, flags, startId);
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        timer.cancel();
    }
}

```

2

回到前景的 Activity 處理。

- 開發一個自訂類別繼承 android.content.BroadcastReceiver
- 改寫 onReceive() 方法，針對傳進來的 Intent 物件實體參數，建構出該 BroadcastReceiver 物件實體
- 建構一個 IntentFilter 物件實體，設定 Action 字串，與 service 端配合使用
- 由 Activity 呼叫 registerReceiver() 方法，傳入 BroadcastReceiver 物件實體，與 intentFilter 物件實體

MainActivity.java

```

package tw.brad.android.book.myservicetest;

import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends Activity {
    private Button start, stop;
    private TextView msg;

```

```
private MyReceiver mr;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    msg = (TextView)findViewById(R.id.msg);

    start = (Button)findViewById(R.id.start);
    start.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            startMyService();
        }
    });

    stop = (Button)findViewById(R.id.stop);
    stop.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            stopMyService();
        }
    });
}

mr = new MyReceiver();
registerReceiver(mr, new IntentFilter("fromMyService"));

}

// 用來負責啟動 MyService
private void startMyService(){
    Intent it= new Intent(this, MyService.class);
    it.putExtra("i", (int)(Math.random()*49+1));
    startService(it);
}

// 用來負責結束 MyService
private void stopMyService(){
    Intent it= new Intent(this, MyService.class);
    stopService(it);
}

private class MyReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        int i = intent.getIntExtra("i", -1);
        msg.setText("i = " + i);
    }
}
}
```

03

Chapter

基本使用者介面與事件觸發

- 3-1 條列顯示元件 ListView
- 3-2 線性配置 LinearLayout
- 3-3 相對配置 RelativeLayout
- 3-4 表格配置 TableLayout
- 3-5 網格顯示 Grid View
- 3-6 滑動顯示 ViewFlipper





3-1 條列顯示元件 ListView

大部分的 Android 初學書籍或是網路文章部落格，都不太會將 ListView 放在一開始介紹，因為會有些許難度。筆者認為反正早晚都會面對，又不是一定要先學會較簡單的 LinearLayout 或是 RelativeLayout 才能夠了解 ListView，再加上 ListView 可用來當作第一個畫面，讀者可以運用第二章學習到 Activity 之間的切換，練習切換到其他的版面配置，往後這個專案 App 可以留著參考或是增加更多學習的版面配置，如此一來更具有學習上的實用價值意義，因此，就開始吧！

■ 基本款式 ■

這次並非先從版面配置下手，直接回到 Activity 設計下手。先將 extends Activity 改成 ListActivity。其實 ListActivity 從名稱來看大約可以判斷其為 Activity 的子類別，所以具有與 Activity 相同的生命週期運作，特色就是本身直接含有一個 ListView 元件。直接呼叫 getListView() 將會回傳本身的 ListView 物件實體。並且記得將 setContentView() 那一列刪除掉，因為整個畫面就是一個 ListView，不再需要另外指定。

MainActivity.java

```
package tw.brad.book.hellolayout;

import android.app.ListActivity;
import android.os.Bundle;
import android.widget.ListView;

public class MainActivity extends ListActivity {
    private ListView lview;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        lview = getListView();
```

```
    }  
}
```

接著下面的處理重點在於條列的單一項目版面配置（不是整個畫面）。就先用原來已經存在的 `activity_main.xml`，如果嫌名稱不好，可在其上按下滑鼠右鍵「Refactor → Rename」更改。筆者換成「`menuitem.xml`」

內容略作些許修改，因為要當成使用者點按的項目，所以將字體略作調整。

`menuitem.xml`:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:paddingBottom="@dimen/activity_vertical_margin"  
    android:paddingLeft="@dimen/activity_horizontal_margin"  
    android:paddingRight="@dimen/activity_horizontal_margin"  
    android:paddingTop="@dimen/activity_vertical_margin"  
    >  
  
    <TextView  
        android:id="@+id/item_txt"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:textSize="24sp" />  
  
</RelativeLayout>
```

3

再度回到 `MainActivity.java` 中。

`ListView` 可以透過呼叫 `setAdapter()` 方法將資料以條列方式呈現，呈現版面如上的 `menuitem.xml`，接下來就是處理這中間的關係。負責串起中間關係的關鍵就是實作 `android.widget.ListAdapter` 的物件實體，可以經由 `android.widget.SimpleAdapter` 來建構出來。要顯示的資料以字串陣列存放，全部資料放在 `java.util.List` 的資料結構中，每個元素會透過 `java.util.Map` 的資料的 Key 將值對應到版面中 `id` 為 `item_txt` 的元件內容上。

宣告定義：

```
private SimpleAdapter adapter;
private String[] items = {
    "LinearLayout", "RelativeLayout", "TableLayout"
};
private String[] from = {"item_txt"};
private int[] to = {R.id.item_txt};
private ArrayList<HashMap<String, String>> data;
```

再來在 onCreate() 中：

1. 建構出 data 的物件實體
2. 設計 for-each 迴圈將 items[] 的資料逐筆跑過
3. 每跑一筆 item，先建構出單一的 HashMap 資料結構物件實體
4. 放進 Key 為 from[0] 的字串，其值為 item
5. 再將 HashMap 物件加進 data 中
6. 跑玩 for-each 迴圈後，就可以建構出 SimpleAdapter 物件實體了
7. 最後 ListView 設定 adapter 即可

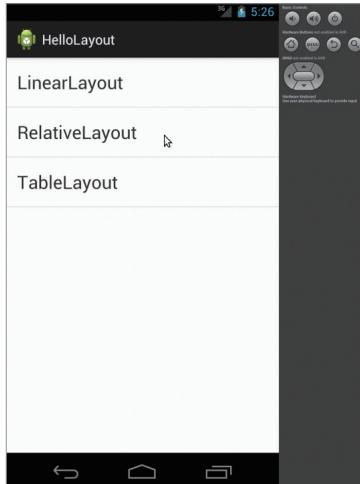
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    lview = getListView();

    data = new ArrayList<HashMap<String, String>>();
    for (String item : items) {
        HashMap<String, String> temp = new HashMap<String, String>();
        temp.put(from[0], item);
        data.add(temp);
    }
    adapter = new SimpleAdapter(this, data, R.layout.menuitem, from, to);

    lview.setAdapter(adapter);
}
```

應該會有以下畫面呈現出來：



3

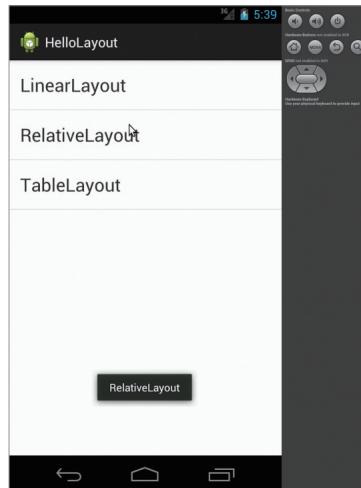
應該是相當容易處理的使用者介面。

接著處理重點在於，按下一選項的動作處理。處理觀念就是針對 ListView 物件設定呼叫 `setOnItemClickListener()` 將選項按下監聽物件參數傳入即可。

下面這段處理方式是將傳入的第三個 int 參數，以顯示 `String[index]` 方式取得，並透過 `Toast` 方式顯示出來。

```
lvview.setOnItemClickListener(new OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> aview, View view, int index,  
        long arg3) {  
        Toast.makeText(MainActivity.this, items[index], Toast.LENGTH_  
        LONG).show();  
    }  
});
```

當使用按下第二個選項：RelativeLayout 之後，顯示如下結果：



|| 進階款式 ||

寫到這邊一定會覺得單一選項的內容過於單調，應該可以弄的比較豐富一點。

假設會有個生動活潑的圖示，有標題列含有簡單說明列。也先將準備好的圖示放在 res/drawable/ 子目錄下，如果沒有存在 drawable/ 子目錄的話，直接新增一個吧。（記得大小寫嚴格區分）

所以先回到版面配置單一選項中處理

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:paddingBottom="@dimen/activity_vertical_margin"  
    android:paddingLeft="@dimen/activity_horizontal_margin"  
    android:paddingRight="@dimen/activity_horizontal_margin"  
    android:paddingTop="@dimen/activity_vertical_margin" >  
  
    <ImageView  
        android:id="@+id/item_img"  
        android:layout_width="wrap_content"
```

```

        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true" />

    <TextView
        android:id="@+id/item_txt"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_toRightOf="@+id/item_img"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/item_desc"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/item_txt"
        android:layout_toRightOf="@+id/item_img"
        android:textSize="16sp" />

</RelativeLayout>

```

3

增加一個 ImageView(id=item_img) , 及一個 TextView(id=item_desc)。

回到 MainActivity.java 中，因為增加了兩個項目，所以也增加了兩組陣列來存放。

```

private String[] items = { "LinearLayout", "RelativeLayout", "TableLayout" };
private String[] items_desc = { "線性配置：針對元件進行垂直式或是水平式的排列",
    "相對配置：其所屬的所有元件都是依照相對位置擺設",
    "表格配置：配置元件的原理與表格相同" };
private int[] items_img = { R.drawable.item0, R.drawable.item1,
R.drawable.item2 };

```

同時，對應關係的 from[] 與 int[] 也增加對應項目元素：

```

private String[] from = { "item_txt", "item_desc", "item_img" };
private int[] to = { R.id.item_txt, R.id.item_desc, R.id.item_img };

```

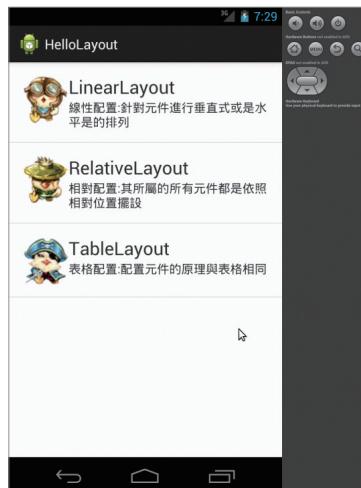
而原本宣告泛型 <String, String>，也要修改為 <String, Object>，因為不再只有 String 的資料，還包含了整數 (Integer) 型態的資料 (Auto-boxing)。

```
private ArrayList<HashMap<String, Object>> data;
```

因此在整理資料的程式碼也進行了些許異動：

```
data = new ArrayList<HashMap<String, Object>>();
for (int i=0; i < items.length; i++){
    HashMap<String, Object> temp = new HashMap<String, Object>();
    temp.put(from[0], items[i]);
    temp.put(from[1], items_desc[i]);
    temp.put(from[2], items_img[i]);
    data.add(temp);
}
adapter = new SimpleAdapter(this, data, R.layout.menuitem, from, to);
```

執行起來就將會有以下的呈現效果。



3-2 線性配置 LinearLayout

LinearLayout 的線性配置版面方式，應該算是所有版面配置中最簡單，也是最常用最好用的配置方式。只有兩種方式配置，不是垂直就是水平，從上至下或是從左至右。雖然如此，卻有一個非常好用的權重屬性，可以以比例方式來配置所屬元件所佔有的空間。

先來看一個基本 LinearLayout 配置架構：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
```

3

- </LinearLayout>
 xmlns:XXX="" → 用來定義該元件的 XML tag 的命名空間，通常不需要特別去處理。XXX 是可以自訂名稱，如果自訂名稱之後，後面的屬性應該就是 xxx: 屬性名稱，目前預設為 android 就可以。
- layout_width 與 layout_width 標示該元件配置的寬高，通常在最外層的 Layout 應該都是 match_parent，而這兩個屬性是大多數元件都必須設定的項目。
- orientation 表示配置方向，不是 vertical (垂直) 就是 horizontal (水平)

現在加上一個元件（都先以 TextView 作為內容元件）：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#00ff00"
        android:text="Hello, Android"
    />

</LinearLayout>
```

TextView 加上 background 屬性不是用來好看，而是可以幫助學習了解元件配置的空間範圍。

目前被放在左上角。

如果只有一個元件，其實也可以省略 LinearLayout 的 orientation 的屬性；但是兩個以上就不可以省略了。

再加上一個 TextView，並使其各自寬度佔滿 LinearLayout 這個父容器。

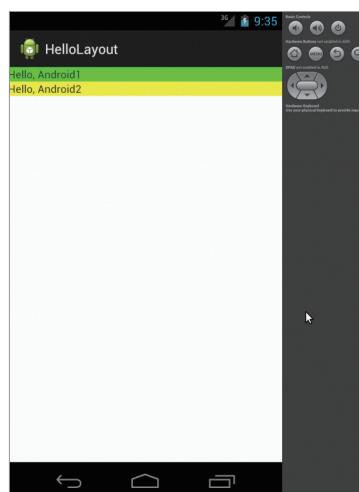
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#00ff00"
        android:text="Hello, Android1"
    />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ffff00"
        android:text="Hello, Android2"
    />

</LinearLayout>
```

呈現出來的效果如下：



再來在各自的屬性中加上 `layout_weight`，並設定其值為 "1"。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#00ff00"
        android:text="Hello, Android1"
        android:layout_weight="1"
    />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ffff00"
        android:text="Hello, Android2"
        android:layout_weight="1"
    />

</LinearLayout>
```

3

呈現效果就是兩個元件填滿父容器，各自以 1:1 的比例佔有空間。



如果想要配置上下元件各為 1:4 的比例，則設定如下：

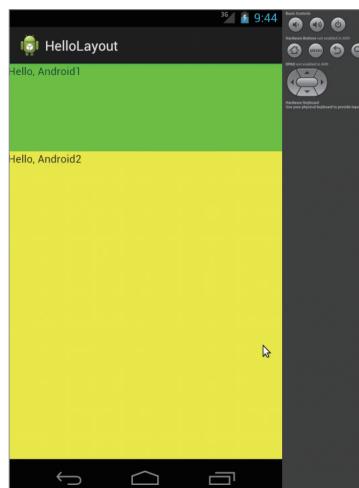
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#00ff00"
        android:text="Hello, Android1"
        android:layout_weight="1"
    />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ffff00"
        android:text="Hello, Android2"
        android:layout_weight="4"
    />

</LinearLayout>
```

呈現效果如下：



善加運用 layout_weight 的比例配置，可以適用於多種不同螢幕規格尺寸的行動裝置上。



3-3 相對配置 RelativeLayout

RelativeLayout 的版面配置就是其所屬的元件，配置的位置全部都是以相對的方式進行指派。可能會相對於父容器的上下左右，或是在特定已經完成指派位置之元件的上下左右。這樣的配置方式也比較能夠符合大多數的螢幕尺寸。

先來進行基本的架構：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

</RelativeLayout>
```

3

直接先放兩個背景顏色不一樣的 TextView 元件上去。先是一個 "Hello, World and Brad"，再來一個 "Hello, Android"。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#ffff00"
        android:text="Hello, World and Brad" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#00ff00"
        android:text="Hello, Android" />

</RelativeLayout>
```

實驗結果：

- 看到完整的 "Hello, Android"，而 "Hello, World and Brad" 只看到後面部份字符串。

- 沒有指定相對位置的元件，預設從左上角開始擺放
- XML 文件中最先提及的元件在最下面，之後一個一個疊上去

開始進行相對位置的設定。先將 "Hello, World and Brad" 放在父容器的正中央，而 "Hello, Android" 則放在 "Hello, World and Brad" 上面，並且水平取中。

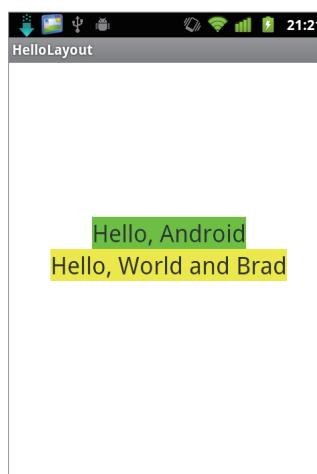
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:id="@+id/tv1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#ffff00"
        android:layout_centerInParent="true"
        android:text="Hello, World and Brad" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/tv1"
        android:layout_centerHorizontal="true"
        android:background="#00ff00"
        android:text="Hello, Android" />

</RelativeLayout>
```

呈現以下結果：



重點說明：

- 雖然呈現結果是 "Hello, Android" 在上，"Hello, World and Brad" 在下，但是在進行版面配置文件時，卻是先提及 "Hello, World and Brad"，因為其元件的位置條件為在父容器的正中央，只有與父容器有相對位置存在。而 "Hello, Android" 却必須相依於 "Hello, World and Brad"，也就是在其上面。因此必須在之後提及。
- "Hello, World and Brad" 的 id 屬性賦予，是因為要給 "Hello, Android" 元件使用在 layout_above 屬性使用指定。並不一定是在程式開發中會使用，才會賦予 id 屬性。
- 通常在位置中只有相對父容器者會先提及，再來就是相對於已經確定位置之元件的元件位置。

3

再來一個練習，先看到想要設計的畫面：



設計要點：

- 可以分成三大區塊，上下中
- 上列有一個 Button 在最右邊，EditText 在最左邊，但是 EditText 也沿著 Button 的左邊，其下方也沿著 Button 下方

- 下列中有四個寬度一樣的 Button
- 中間是一個 TextView，其位置在上列的下面；在下列的上面

如下實際規劃配置：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <RelativeLayout
        android:id="@+id/top"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <Button
            android:id="@+id/search"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_alignParentTop="true"
            android:text="Search" />

        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
            android:layout_toLeftOf="@+id/search"
            android:layout_alignBottom="@+id/search"
            android:background="#ffff00" />
    </RelativeLayout>

    <LinearLayout
        android:id="@+id/func"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:orientation="horizontal" >

        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Add" />
    </LinearLayout>

```

3

```
<Button  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_weight="1"  
    android:text="Delete" />  
  
<Button  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_weight="1"  
    android:text="Config" />  
  
<Button  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_weight="1"  
    android:text="Help" />  
</LinearLayout>  
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_above="@+id/func"  
    android:layout_below="@+id/top"  
    android:background="#00ff00"  
    android:text="Hello, OK"  
/>  
  
</RelativeLayout>
```



練習版面配置時，可以為元件加上不同的 background 屬性，用意不是好看，而是會清楚該元件所佔有的空間有多大。



3-4 表格配置 TableLayout

顧名思義就是以表格方式呈現內容，很常見於呈現特定的表格資料內容，例如個人流水帳中的帳務內容，或是統計資料，股票資訊等等。

基本上是以 `<TableLayout>...</TableLayout>` 為版面配置的方式。

基本架構為 `<TableLayout> - <TableRow> - <View 元件>` 如果玩過 HTML 者，大概都會聯想到 `<table> - <tr> - <td>`。

以 `<TableRow>` 標示出一列，該列的寬度就是佔滿 `<TableLayout>`，先來最基本的測試：

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <TableRow>

        <TextView
            android:background="#ffff0000"
            android:text="Data 1" />

        <TextView
            android:background="#ff00ff00"
            android:text="Data 2" />

        <TextView
            android:background="#ff0000ff"
            android:text="Data 3" />
    </TableRow>

</TableLayout>
```

其呈現結果如下圖：

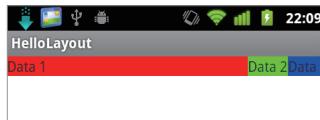


還蠻醜的。因為 `TextView` 預設下是以依照內容多寡來決定其佔有空間。

開始進行修正處理：在 `<TableLayout>` 中加上屬性：

```
    android:stretchColumns="0"
```

馬上就有不一樣的效果呈現，發現到是將第 1 個資料欄進行擴展，而第 2, 3 個資料項，就是依照資料內容決定大小。



再做些修正。

```
android:stretchColumns="0,2"
```

發現到是將第 1 個及第 3 個資料欄進行平均分配的擴展，而第 2 個資料項，就是依照資料內容決定大小。

3

通常還可以在 <TableRow> 之間加上一個 <View> 類似做出分隔線的效果。

```
<View  
    android:layout_width="fill_parent"  
    android:background="#fffff00"  
    android:layout_height="2dp"  
/>
```

如果特定資料項的位置要留白，則其他資料項最好指定其 layout_column 屬性，例如：



```
<?xml version="1.0" encoding="utf-8"?>  
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:stretchColumns="0,1,2" >  
  
    <TableRow>  
  
        <TextView
```

```
        android:layout_margin="2dp"
        android:background="#ffff0000"
        android:text="Data 11" />

    <TextView
        android:layout_margin="2dp"
        android:background="#ff00ff00"
        android:text="Data 12" />

    <TextView
        android:layout_margin="2dp"
        android:background="#ff0000ff"
        android:text="Data 13" />
</TableRow>

<View
    android:layout_width="fill_parent"
    android:layout_height="2dip"
    android:background="#ffff00" />

<TableRow>

    <TextView
        android:layout_column="0"
        android:background="#ffff0000"
        android:text="Data 21" />

    <TextView
        android:layout_column="2"
        android:background="#ff0000ff"
        android:text="Data 23" />
</TableRow>

</TableLayout>
```



3-5 網格顯示 GridView

GridView 用來呈現出格子式的配置，最常見的就是相片的陳列。先以早期官方網站的方式來解說，再以不同的變化來更彈性的處理說明。

建立以 GridView 為版面配置的 xml：

```
<?xml version="1.0" encoding="utf-8"?>
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gv"
```

```
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:columnWidth="90dp"
    android:numColumns="auto_fit"
    android:verticalSpacing="10dp"
    android:horizontalSpacing="10dp"
    android:stretchMode="columnWidth"
    android:gravity="center"
/>
```

開發一個自訂類別繼承 BaseAdapter：

3

```
private class ImageAdapter extends BaseAdapter {
    private Context mContext;

    public ImageAdapter(Context c) {
        mContext = c;
    }

    public int getCount() {
        return mThumbIds.length;
    }

    public Object getItem(int position) {
        return null;
    }

    public long getItemId(int position) {
        return 0;
    }

    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView imageView;
        if (convertView == null) {
            imageView = new ImageView(mContext);
            imageView.setLayoutParams(new GridView.LayoutParams(85, 85));
            imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
            //imageView.setPadding(8, 8, 8, 8);
        } else {
            imageView = (ImageView) convertView;
        }

        imageView.setImageResource(mThumbIds[position]);
        return imageView;
    }

    private Integer[] mThumbIds = {
        R.drawable.android, R.drawable.apple,
        R.drawable.bells, R.drawable.brick_dig,
```

```

        R.drawable.bricks, R.drawable.cup,
        R.drawable.delete, R.drawable.edit,
        R.drawable.enemy1, R.drawable.enemy2,
        R.drawable.eraser, R.drawable.gift,
        R.drawable.help, R.drawable.mango,
        R.drawable.next, R.drawable.open,
        R.drawable.save, R.drawable.share,
        R.drawable.speed
    } ;
}

```

回到 onCreate() 方法中

```

private GridView gv;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.gridview1);

    gv = (GridView)findViewById(R.id.gv);
    gv.setAdapter(new ImageAdapter(this));
}

```

將會呈現以下效果。(前提是也將相對的影像圖檔放進 res/drawable/ 子目錄下)



其實 GridView 的原理觀念與 ListView 類似，因此以下利用在 3-1 小節的觀念原理實作。

先從單一項目的版面配置處理：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical" >

    <ImageView
        android:id="@+id/gvimg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/android"
    />
    <TextView
        android:id="@+id/gtvv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="Android"
    />
</LinearLayout>
```

3

再回到程式中處理，一樣利用 SimpleAdapter 來進行轉換 ...

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.gridview1);

    // gv = (GridView) findViewById(R.id.gv);
    // gv.setAdapter(new ImageAdapter(this));

    gv = (GridView) findViewById(R.id.gv);

    String[] from = {"gvimg", "gtvv"};
    int[] to = {R.id.gvimg, R.id.gtvv};
    ArrayList<HashMap<String, Object>> data = new ArrayList<HashMap<String, Object>>();
    HashMap<String, Object> data1 = new HashMap<String, Object>();
```

```

        data1.put("gvimg", R.drawable.android); data1.put("gvtv", "data1");
data.add(data1);
        HashMap<String, Object> data2 = new HashMap<String, Object>();
        data2.put("gvimg", R.drawable.apple); data2.put("gvtv", "data2");
data.add(data2);
        HashMap<String, Object> data3 = new HashMap<String, Object>();
        data3.put("gvimg", R.drawable.mango); data3.put("gvtv", "data3");
data.add(data3);
        HashMap<String, Object> data4 = new HashMap<String, Object>();
        data4.put("gvimg", R.drawable.bells); data4.put("gvtv", "data4");
data.add(data4);
        HashMap<String, Object> data5 = new HashMap<String, Object>();
        data5.put("gvimg", R.drawable.cup); data5.put("gvtv", "data5");
data.add(data5);

        gv.setAdapter(new SimpleAdapter(this, data, R.layout.gvtest, from,
to));

    }
}

```

就可以呈現出圖文並茂的網格項目：



3-6 滑動顯示 ViewFlipper

ViewFlipper 通常用來處理在一個 Activity 中，會有兩個以上的 ViewGroup，而切換方式可能會搭配手勢左右滑動。這樣的處理模式意味著一個 Activity 的想要顯示畫面在行動裝置硬塞下去不好看，所以分成兩個以上的畫面呈現。

以下實例說明處理方式

假設有三個呈現畫面：

1. 第一個畫面

```
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:background="#fffff00"  
>  
  
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:text="第一個畫面" >  
</TextView>  
</LinearLayout>
```

3

2. 第二個畫面

```
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:background="#fffff00"  
>  
  
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:text="第二個畫面" >  
</TextView>  
</LinearLayout>
```

3. 第三個畫面

```
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:background="#00ff00"  
>  
  
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:text="第三個畫面" >
```

```
</TextView>
</LinearLayout>
```

三個畫面的背影顏色也是讓練習時了解切換的實際狀況。

將三個畫面直接包進 ViewFlipper 中間（別客氣，別猶豫）

res/vflipper1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ViewFlipper xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/vf"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000000" >

</ViewFlipper>
```

這樣就處理好版面配置的程序了，接下來處理 Activity 的控制了。

宣告變數：

```
private ViewFlipper vf;
```

收回使用並設定觸摸事件：

```
vf = (ViewFlipper) findViewById(R.id.vf);

vf.setOnTouchListener(new OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        vf.showNext();
        return super.onTouchEvent(event);
    }
});
```

ViewFlipper 物件實體的 showNext() 會直接跳到其下一個子 View 呈現； showPrevios() 則剛好相反。所以上面的處理下，使用者摸一下就換頁，無效果可言。為了達到手勢偵測，先建立一個 GestureDetector 物件實體，並處理其事件程序：

宣告手勢偵測物件變數：

```
private GestureDetector gd;
```

先開發手勢偵測事件監聽器類別：

```
private class MyGDLListener extends GestureDetector.SimpleOnGestureListener {
    @Override
    public boolean onDown(MotionEvent e) {
        return super.onDown(e);
    }
    @Override
    public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
        float velocityY) {
        return super.onFling(e1, e2, velocityX, velocityY);
    }
}
```

3

onDown() 方法要 Override，並將其傳回值設定為 true，才會往下 onFling() 方法進行偵測手勢劃過螢幕事件。

- 第三個參數 velocityX，表示在 X 軸的劃過速度（每秒像素）
- 往右為正值 velocityX
- 往左為負值 velocityX

所以改寫如下：

```
private class MyGDLListener extends GestureDetector.SimpleOnGestureListener {
    @Override
    public boolean onDown(MotionEvent e) {
        return true;
    }
    @Override
    public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
        float velocityY) {
        if (velocityX < -10) {
            vf.showNext();
        } else if (velocityX > 10) {
            vf.showPrevious();
        }
    }
}
```

```

        return super.onFling(e1, e2, velocityX, velocityY);
    }
}

```

建構出 GestureDetector 物件實體：

```
gd = new GestureDetector(this, new MyGDListener());
```

從 ViewFlipper 物件實體的 onTouchEvent() 設定傳回值，交給 gd.onTouchEvent()：

```

vf.setOnTouchListener(new OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        return gd.onTouchEvent(event);
    }
});
```

再來處理動畫效果，

在 res/anim/ 中建立以下四種動畫：

用在 ViewFlipper 的 showNext()，也就是翻下一页

從左到右的進入：in_leftright.xml

```

<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" >
    <translate
        android:duration="1000"
        android:fromXDelta="-100%"
        android:toXDelta="0" />
</set>
```

- 針對 X 軸的動態效果
- 進入者之前的的左邊為 -100%
- 進入後為 0 定位顯示

從左到右的離開：out_leftright.xml

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

    <translate
        android:duration="1000"
        android:fromXDelta="0"
        android:toXDelta="100%" />

</set> •
```

- 出去者之前的左邊是 0
- 出去後為 100%

用在 ViewFlipper 的 showPrevious()，也就是回上一頁。

3

從右到左的進入：in_rightleft.xml

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

    <translate
        android:duration="1000"
        android:fromXDelta="100%"
        android:toXDelta="0" />

</set>
```

從右到左的離開：out_rightleft.xml

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

    <translate
        android:duration="1000"
        android:fromXDelta="0"
        android:toXDelta="-100%" />

</set>
```

再度回到 Activity 中。

宣告動畫處理物件變數：

```
private Animation in_l2r, out_l2r, in_r2l, out_r2l;
```

載入動畫效果：

```
in_l2r = AnimationUtils.loadAnimation(this, R.anim.in_leftright);
out_l2r = AnimationUtils.loadAnimation(this, R.anim.out_leftright);
in_r2l = AnimationUtils.loadAnimation(this, R.anim.in_rightleft);
out_r2l = AnimationUtils.loadAnimation(this, R.anim.out_rightleft);
```

再度改寫 onFling()：

```
public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
    float velocityY) {
    if (velocityX < -10) {
        vf.setInAnimation(in_r2l);
        vf.setOutAnimation(out_r2l);
        vf.showNext();
    } else if (velocityX > 10) {
        vf.setInAnimation(in_l2r);
        vf.setOutAnimation(out_l2r);
        vf.showPrevious();
    }
    return super.onFling(e1, e2, velocityX, velocityY);
}
```

OK，相當單純簡單。

04

Chapter

對話框與通知事件處理

- 4-1 AlertDialog 對話框的使用
- 4-2 自訂對話框 Dialog 與日期時間對話框
- 4-3 Toast 及自訂 Toast
- 4-4 進度顯示對話框
- 4-5 通知列處理模式





4-1 AlertDialog 對話框的使用

對話框對於行動裝置而言，可以說是使用頻率相當高的項目之一。在使用者介面中，透過對話框與使用者互動，進而決定程式運作的下一步，或是確認動作的執行等等。在 Android API 中提供了 AlertDialog 的類別用以快速的處理對話框的開發，`android.app.AlertDialog` 是繼承自 `android.app.Dialog`，所以其所建構的物件實體 is-a Dialog，只需要針對開發者自訂部份進行處理即可。

建立 AlertDialog 物件

最常見的方式就透過 AlertDialog 的內部類別 Builder 進行相關的設定，設定之後再呼叫 `create()` 方法而傳回一個 AlertDialog 物件實體，之後就可以呼叫 AlertDialog 物件實體的 `show()` 方法將對話框呈現出來。

- `AlertDialog.Builder builder = new AlertDialog.Builder(this);`
- `builder.setXxx();`
- `AlertDialog alert = builder.create();`
- `alert.show()`

訊息對話框

只有提供訊息給使用者對話框，使用者看完之後按下行動裝置的返回鍵即可離開。只需要透過 `AlertDialog.Builder` 物件進行設定兩項內容：

- `setTitle():` 設定對話框標題
- `setMessage():` 設定訊息內容
- `setCancelable():` 設定是否允許使用者自行關閉（按下返回鍵）對話框
(預設為 true)
- `setIcon():` 設定顯示的圖示



因此處理方式非常單純簡單：

```
private void showAlertDialog1(){
    AlertDialog alert = null;
    AlertDialog.Builder builder = new AlertDialog.Builder(this);

    builder.setTitle("確認對話框");
    builder.setMessage("歡迎使用超級強大功能的 App");

    alert = builder.create();
    alert.show();
}
```

4

如果覺得要使用者自行按下返回鍵的動作不恰當，也可以考慮搭配執行緒的處理，使該訊息在指定的時間內呈現，之後自行消失。通常這種模式是強迫使用者閱讀對話框，一般會搭配週期時間來使對話框自動消失，而使用者無法按下返回鍵使其提早消失。

以下範例的處理將會使對話框出現 4 秒後自動消失，使用者無法提早按下返回鍵使其消失。

```
package tw.brad.android.book.MyAlertDialog;

import java.util.Timer;
import java.util.TimerTask;

import android.app.Activity;
import android.app.AlertDialog;
```

```
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends Activity {
    private Button dialog1, dialog2;
    private AlertDialog alert;
    private Timer timer;
    private MyHandler handler;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        timer = new Timer();
        handler = new MyHandler();

        dialog1 = (Button)findViewById(R.id.dialog1);
        dialog1.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                showAlertDialog1();
            }
        });
    }

    private void showAlertDialog1() {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);

        builder.setTitle("確認對話框");
        builder.setMessage("歡迎使用超級強大功能的 App");
        builder.setCancelable(false);

        alert = builder.create();
        alert.show();
        timer.schedule(new CloseDialogTask(), 4000);
    }

    private class CloseDialogTask extends TimerTask {
        @Override
        public void run() {
            handler.sendEmptyMessage(0);
        }
    }
}
```

```

private class MyHandler extends Handler {
    @Override
    public void handleMessage(Message msg) {
        alert.dismiss();
    }
}

```

確認式對話框

與上個對話框處理模式相同，但是使用者可以按下確認鍵或是再多一個取消按鍵作出確認或是決定，甚至於同時有三種狀況的按鈕。處理方式就是加上三種按鈕的設定，任何一個按鈕都會使其對話框自動消失。

Builder 為 AlertDialog.Builder 的物件實體，以下來處理三個按鈕：(不一定同時三個按鈕都實作，視需要而定)

4

- setPositiveButton() => 右邊
- setNegativeButton() => 左邊
- setNeutralButton() => 中間

```

builder.setPositiveButton("確定 (右)", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        // 按下確定後的執行程序
    }
});

builder.setNegativeButton("取消 (左)", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        // 按下取消後的執行程序
    }
});

builder.setNeutralButton("再說 (中)", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        info.setText("剛剛按下再說 ");
    }
});

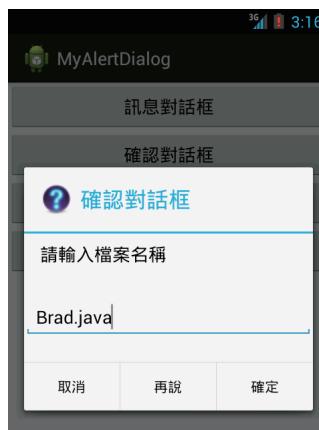
```

當然，也可以只任選一個按鈕事件來處理。但是要切記的一點是，帶入的第二個參數為實作 DialogInterface.OnClickListener 介面的物件實體，所以是 new DialogInterface.OnClickListener()，而不是 new View.OnClickListener()。

使用一個按鈕的狀況，通常是讓使用者閱讀完畢後自行按下按鈕確認，例如顯示一份同意書，條文規定等等，或是告知目前提供資訊；使用兩個按鈕的狀況，通常是請使用者當下做出決定，例如是／否，確定／取消，或是登入／重置等等。



或是改成詢問確認對話框：



```
private void showAlertDialog2() {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);

    builder.setTitle("確認對話框");
    builder.setMessage("請輸入檔案名稱");
    builder.setIcon(R.drawable.question);

    filename = new EditText(MainActivity.this);
    builder.setView(filename);

    builder.setPositiveButton("確定", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            info.setText("剛剛按下確定：" + filename.getText().toString());
        }
    });

    builder.setNeutralButton("再說", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            info.setText("剛剛按下再說");
        }
    });

    builder.setNegativeButton("取消", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            info.setText("剛剛按下取消");
        }
    });

    builder.setCancelable(false);

    alert = builder.create();
    alert.show();
}
```

4

選擇式對話框

通常用於四個以上選項的對話框，要求使用者在多個選項中做出一個或多個選擇。例如以下狀況：

影像檔案 brad.jpg 已經修改 ...

- 立即儲存
- 另存新檔
- 放棄修改
- 刪除檔案

處理要點如下：

- 通常也會有 setTitle()
- 但是沒有 setMessage() => 重要
- 選擇項目是透過 setItems()，傳入第一個參數為一個字串陣列，為其選項內容，第二個參數為按下選項後的事件處理。



```
// 單一直接選擇式
builder.setItems(opts, new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        info.setText("您剛剛選擇的是：" + opts[which]);
    }
});
```

使用者按下特定項目後就直接視為完成選擇。

有人會覺得使用者可能因為手指觸控誤差，而改成確認處理模式，就必須使用呼叫 `setSingleChoiceItems()` 方法來處理，並加上 `setXxxButton()` 作確認的動作。

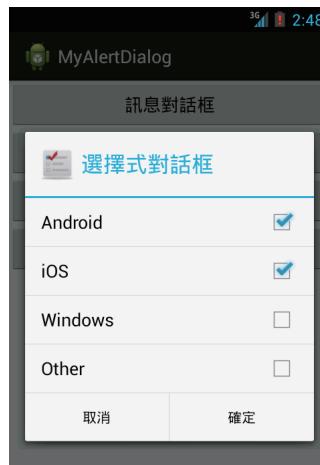


4

```
// 單一確認選擇式
builder.setSingleChoiceItems(opts, 0, new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        whichItem = which;
    }
});
builder.setPositiveButton("確定", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        info.setText("剛剛按下確定的項目是 :" + opts[whichItem]);
    }
});

builder.setNegativeButton("取消", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        info.setText("剛剛按下取消");
    }
});
```

也可以處理成多選式，就是以呼叫 `setMultiChoiceItems()` 來處理，但是就必須處理一個以上的 `setXxxButton()` 方法，負責將最後結果作確認或是取消。



```
// 多重選擇式
builder.setMultiChoiceItems(opts, checkedItems, new OnMultiChoiceClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which, boolean isChecked) {
    }
});
builder.setPositiveButton("確定", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        info.setText("剛剛按下確定的項目是 :\n");
        for (int i=0; i<opts.length; i++){
            if (checkedItems[i]){
                info.append(opts[i] + "\n");
            }
        }
    }
});

builder.setNegativeButton("取消", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        info.setText("剛剛按下取消 ");
    }
});
```

進階選擇式對話框

如果覺得前三者的對話框過於單調，可以考慮多點處理動作，以 `listAdapter` 來處理會比較有質感。這種型態的對話框與上述的選擇式對話框類似，只是將 `setItems()` 改為 `setAdapter()` 來處理，而 `listAdapter` 的處理模式與 `ListView` 相同。

1. 建立一個選項內容的版面
2. 設定 Adapter 的對應關係
3. `builder.setAdapter()`



4

單一選項的版面處理：

```
res/layout/menuitem.xml <?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    >
    <ImageView
        android:id="@+id/item_img"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
    />
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    >
    <TextView
        android:id="@+id/item_title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="32sp"
        android:textStyle="bold|italic"
    />
    <TextView
        android:id="@+id/item_content"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="16sp"
    />
</LinearLayout>
</LinearLayout>
```

程式處理：

```
private void showAlertDialog4(){
    AlertDialog.Builder builder = new AlertDialog.Builder(this);

    builder.setTitle(" 您的行動裝置作業系統為何 ?");
    builder.setIcon(R.drawable.opts);

    data = new ArrayList();

    HashMap<String, Object> item0 = new HashMap();
    item0.put(from[0], R.drawable.android);
    item0.put(from[1], opts[0]);
    item0.put(from[2], "就是你現在正在學習的東西呀 ");
    data.add(item0);

    HashMap<String, Object> item1 = new HashMap();
    item1.put(from[0], R.drawable.apple);
    item1.put(from[1], opts[1]);
    item1.put(from[2], "另一個陣營的好東西 ");
    data.add(item1);

    HashMap<String, Object> item2 = new HashMap();
    item2.put(from[0], R.drawable.windows);
```

```
item2.put(from[1], opts[2]);
item2.put(from[2], "PC 市場上無人不知無人不曉");
data.add(item2);

HashMap<String, Object> item3 = new HashMap();
item3.put(from[0], R.drawable.other);
item3.put(from[1], opts[3]);
item3.put(from[2], "嗯 ...");
data.add(item3);

SimpleAdapter adapter = new SimpleAdapter(
    this, data,
    R.layout.menuitem,
    from, to);

builder.setAdapter(adapter, new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        info.setText("您剛剛選擇的是：" + opts[which]);
    }
});

alert = builder.create();
alert.show();
}
```

4



4-2 自訂對話框（Dialog）與日期時間對話框

自訂對話框

自訂對話框通常應用於對話框內容，會提供不同於單純選項的互動輸入狀況，或是想要提供美觀性更高的使用者介面。處理的對象為以 android.app.Dialog 類別物件為主，版面配置通常會以 xml 格式的 layout 檔案為呈現內容，再從程式中建構出 Dialog 物件實體，兩者進行結合使用。所以連使用者介面的相關確認按鈕、相關元件及事件處理，都不像上一小節中的 AlertDialog 中只需要 Override 其特定事件傾聽事件即可，甚至於連關閉對話框的動作程序都不是自動處理，而優點就是版面配置的彈性比較大。

以下的註冊對話框就是必須以自訂方式來處理：



先設計出希望呈現的版面配置：

res/layout/newreg.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:text="拉拉網帳號"
        />
    <EditText
        android:id="@+id/reg_username"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:singleLine="true"
        />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:text="拉拉網暱稱"
        />
```

```
<EditText  
    android:id="@+id/reg_nickname"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:singleLine="true"  
/>  
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:textSize="18sp"  
    android:text="使用的密碼"  
/>  
<EditText  
    android:id="@+id/reg_passwd"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:password="true"  
    android:singleLine="true"  
/>  
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:textSize="18sp"  
    android:text="再次輸入以確認"  
/>  
<EditText  
    android:id="@+id/reg_passwd2"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:password="true"  
    android:singleLine="true"  
/>  
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:textSize="18sp"  
    android:text="手機號碼"  
/>  
<EditText  
    android:id="@+id/reg_phonenum"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:singleLine="true"  
    android:numeric="integer"  
/>  
  
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:orientation="horizontal"  
>
```

4

```

<Button
    android:id="@+id/reg_ok"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="申請"
/>
<Button
    android:id="@+id/reg_cancel"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="取消"
/>
</LinearLayout>
</LinearLayout>

```

在樣式上也設定了背景顏色相關：

values/styles.xml 中

```

<style name="dialogStyle" parent="@android:style/Theme.Dialog">
    <item name="android:background">#449900</item>
</style>

```

而在程式中先將 Dialog 物件建構出來

```
dialog_newreg = new Dialog(this, R.style.dialogStyle);
```

並設定為上述的版面配置

```
dialog_newreg.setContentView(R.layout.newreg);
```

進行與 AlertDialog 相同的基本設定

```

dialog_newreg.setTitle(" 拉拉網 - 新用戶註冊 ");
dialog_newreg.setCancelable(false);

```

而在版面中，使用者互動輸入相關元件的處理，則必須由 Dialog 物件實體來呼叫 findViewById() 方法傳回。

```

eUsername = (EditText) dialog_newreg.findViewById(R.id.reg_username);
eNickname = (EditText) dialog_newreg.findViewById(R.id.reg_nickname);

```

```
ePasswd = (EditText) dialog_newreg.findViewById(R.id.reg_passwd);
ePasswd2 = (EditText) dialog_newreg.findViewById(R.id.reg_passwd2);
ePhonenum = (EditText) dialog_newreg.findViewById(R.id.reg_phonenum);

Button reg_ok, reg_cancel;
reg_ok = (Button) dialog_newreg.findViewById(R.id.reg_ok);
reg_cancel = (Button) dialog_newreg.findViewById(R.id.reg_cancel);
```

最後兩個 Button 的處理與 AlertDialog 中的 Button 不一樣的是，一般 Button 的 OnClick 事件，而不是使用 DialogInterface.OnClickListener 事件。

```
reg_ok.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
    }
});
reg_cancel.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
    }
});
```

4

使用者按下 Button 之後並不會使 Dialog 物件自動消失，除非呼叫 Dialog 物件之 dismiss() 方法，才會使其消失。

```
dialog_newreg.dismiss();
```

當然，最後設定完成後，必須呼叫 Dialog 物件之 show() 方法才能使其呈現出來。

日期選擇對話框

android.app.DatePickerDialog 類別物件用來產生出選擇日期的選取對話框，提供使用者互動友善的選取日期。

建構物件方式通常會需要設定初始的年月日資料，可以透過 java.util.Calendar 類別物件取得目前的年月日資料。

```
int year, month, day;
Calendar cal = Calendar.getInstance();
```

```
year = cal.get(Calendar.YEAR);
month = cal.get(Calendar.MONTH);
day = cal.get(Calendar.DAY_OF_MONTH);
```

接著就可建構出物件實體：

```
dpd = new DatePickerDialog(this, DatePickerDialog.THEME_HOLO_DARK,
    new OnDateSetListener() {
        @Override
        public void onDateSet(DatePicker view, int year, int monthOfYear,
                              int dayOfMonth) {
            info.setText(year + "/" + monthOfYear + "/" + dayOfMonth);
        }
    },
    year, month, day);
```

參數說明：

- 目前的 Context
- 呈現的樣式風格
 - DatePickerDialog.THEME_TRADITIONAL
 - DatePickerDialog.THEME_HOLO_DARK
 - DatePickerDialog.THEME_HOLO_LIGHT
 - DatePickerDialog.THEME_DEVICE_DEFAULT_DARK
 - DatePickerDialog.THEME_DEVICE_DEFAULT_LIGHT
- 變更的事件處理器
- 初始的年份值
- 初始的月份值
- 初始的日期值

再來就是進行一般設定：

```
dpd.setTitle("設定日期");
dpd.setButton(DialogInterface.BUTTON_POSITIVE, "確定", new
DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        dpd.dismiss();
    }
});
```

```
});  
dpd.setButton(DialogInterface.BUTTON_NEGATIVE, "取消", new  
DialogInterface.OnClickListener() {  
    @Override  
    public void onClick(DialogInterface dialog, int which) {  
        dpd.dismiss();  
    }  
});
```

最後記得將其呈現出來。

```
dpd.show();
```



4

時間選擇對話框

android.app.TimePickerDialog 類別物件用來產生出選擇時間的選取對話框，提供使用者互動友善的選取時間。

建構物件方式通常會需要設定初始的時、分資料，可以透過 java.util.Calendar 類別物件取得目前的時分資料。

```
int hour, minute;  
Calendar cal = Calendar.getInstance();  
hour = cal.get(Calendar.HOUR_OF_DAY);  
minute = cal.get(Calendar.MINUTE);
```

接著就可建構出物件實體：

```
tpd = new TimePickerDialog(this,TimePickerDialog.THEME_HOLO_DARK,
    new OnTimeSetListener() {
        @Override
        public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
            info.setText(hourOfDay + ":" + minute);
        }
    },
    hour, minute, true);
```

參數說明：

1. 目前的 Context
2. 呈現的樣式風格
 - TimePickerDialog.THEME_TRADITIONAL
 - TimePickerDialog.THEME_HOLO_DARK
 - TimePickerDialog.THEME_HOLO_LIGHT
 - TimePickerDialog.THEME_DEVICE_DEFAULT_DARK
 - TimePickerDialog.THEME_DEVICE_DEFAULT_LIGHT
4. 變更的事件處理器
5. 初始的小時值
6. 初始的分鐘值
7. 是否為 24 小時制 (boolean)

再來就是進行一般設定：

```
tpd.setTitle(" 設定時間 ");
tpd.setButton(DatePickerDialog.BUTTON_POSITIVE, " 確定 ", new
DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        tpd.dismiss();
    }
});
tpd.setButton(DatePickerDialog.BUTTON_NEGATIVE, " 取消 ", new
DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
```

```
        tpd.dismiss();  
    }  
});
```

最後記得將其呈現出來。

```
tpd.show();
```



4



4-3 Toast 及自訂 Toast

android.widget.Toast 就像是剛從烤麵包機彈跳出來的土司一樣，突然從螢幕上彈出一段訊息。不需要去特別作任何操作，只是提供訊息而已，稍候一下就會自動消失。

一般的 Toast

最簡單的處理方式，就是直接呼叫 Toast 的 static 方法 makeText()，傳遞三個參數：

- Context

- 顯示字串
- 呈現停留時間
 - Toast.LENGTH_LONG
 - Toast.LENGTH_SHORT

然後直接呼叫 show() 方法呈現出來。

```
Toast.makeText(this, "大家好，偶素 Brad", Toast.LENGTH_LONG).show()
```

或是設定其顯示的位置，呼叫其 setGravity() 方法，傳遞三個參數：

- Gravity: (舉例)
 - Gravity.CENTER: 螢幕正中央
 - Gravity.FILL: 填滿螢幕
 - Gravity.FILL_HORIZONTAL+Gravity.TOP : 螢幕上方水平填滿
- x 軸偏移
- y 軸偏移

如下處理範例：

```
Toast tst1 = Toast.makeText(this, "大家好，偶素 Brad", Toast.LENGTH_LONG);
tst1.setGravity(Gravity.FILL_HORIZONTAL+Gravity.CENTER_VERTICAL, 0, 0);
tst1.show();
```



自訂 Toast

而自訂的 Toast 的彈性就比較大，整個版面內容可以自行規劃設計，這樣可以使開發專案 App 的整體風格較為一致，經常看到許多 App 在版面設計上非常用心，甚至於畫面精緻的遊戲，結果當出現上述一般的 Toast 來呈現即時訊息時，整個感覺就非常遜掉了，不妨考慮來撰寫個自訂版面的 Toast 呈現方法來處理，會比較有整體的質感表現。

先來設計自訂的版面配置

```
res/layout/toast_view.xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/toast_layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:background="@drawable/gold_banner"
    >
    <ImageView
        android:id="@+id/toast_icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/sms"
        android:layout_margin="12dp"
        />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_margin="12dp"
        >
        <TextView
            android:id="@+id/toast_title"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="24sp"
            android:textStyle="bold"
            android:textColor="#00ff00"
            />
        <TextView
            android:id="@+id/toast_content"
```

4

```

        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:textStyle="bold|italic"
        android:textColor="#ffffff"
    />
</LinearLayout>

</LinearLayout>

```

呈現一個橫列資訊，左邊為一個圖示，右邊分成上下表現出一個標題與內容。

回到程式中來開發一個方法，可以接收兩個參數，分別就是標題字串與內容字串，當然，讀者也可以將圖示同時變更設定處理。

處理重點如下：

- 呼叫 `getLayoutInflator()` 將會傳回一個 `LayoutInflater` 物件實體
- 執行該 `LayoutInflater` 物件實體的 `inflate()` 方法，傳入兩個參數
 - 自訂版面的資源位置 (`R.layout.toast_view`)
 - 自訂版面中的 `ViewGroup` 物件
- 傳回該版面的 `View` 物件實體
- 再由 `View` 物件實體取得標題與內容的 `TextView` 物件實體
- 設定標題與內容的 `TextView` 物件實體所呈現的文字內容
- 建立 `Toast` 物件實體
- 呼叫 `setView()` 傳入之前的 `View` 物件實體
- 其他處理方式與一般 `Toast` 一樣

如下：

```

private void showToast2() {
    showMyToast("重要訊息", "恭喜您中了大大大樂透");
}

private void showMyToast(String title, String content) {
    LayoutInflator inflater = getLayoutInflator();
    View layout = inflater.inflate(R.layout.toast_view,
        (ViewGroup) findViewById(R.id.toast_layout));

```

```
TextView toast_title = (TextView) layout.findViewById(R.id.toast_title);
TextView toast_content = (TextView) layout.findViewById(R.id.toast_content);

toast_title.setText(title);
toast_content.setText(content);

Toast toast = new Toast(getApplicationContext());
toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
toast.setDuration(Toast.LENGTH_LONG);
toast.setView(layout);
toast.show();

}
```



4

4-4 進度顯示對話框

進度顯示對話框通常是用在當 App 執行中，可能無法立即執行完畢，而必須請使用者稍候得以繼續，例如正在下載網際網路相關的檔案時；或是當 App 正在載入較大的程序時，告知使用者目前的現況，以免使用者以為 App 發生非預期狀態（當機），例如 App 在首次載入執行時，可能必須花較多的時間進行相關資源載入程序。

而進度對話框一般而言可以分成兩種，一種是可以掌握運作程序的進度，例如一開始就知道要下載的檔案大小，而進行檔案傳輸串流過程中也能夠取得目前已經下載的大小，此時可以考慮以呈現目前執行進度的對話框較為恰當。

另一種是無法掌握整體的執行進度相關數據資料，而只能知道執行的結束點，則只好提供動態循環運轉的對話框，讓使用者知道目前仍然在執行中。

■ ProgressDialog

ProgressDialog 的建構方式是直接呼叫該類別所提供的建構式，筆者最常使用的如下：

```
new ProgressDialog(this, ProgressDialog.THEME_HOLO_DARK);
```

傳遞兩個參數：

- Context
- 呈現風格
 - ProgressDialog.THEME_DEVICE_DEFAULT_DARK(API Level 14)
 - ProgressDialog.THEME_DEVICE_DEFAULT_LIGHT(API Level 14)
 - ProgressDialog.THEME_DEVICE_HOLO_DARK(API Level 11)
 - ProgressDialog.THEME_DEVICE_HOLO_LIGHT(API Level 11)
 - ProgressDialog.THEME_DEVICE_TRADITIONAL(API Level 11)

決定是否為進度可以掌控的模式，若無法掌握進度，則呼叫 setProgressStyle() 方法時傳遞 ProgressDialog.STYLE_SPINNER。

```
.setProgressStyle(ProgressDialog.STYLE_SPINNER);
```

通常會搭配訊息的呈現，則呼叫 setMessage() 方法，傳遞訊息字串參數進去。

這樣就算是事先準備工作完成。

以下以一個單純執行緒為例，當然，通常也會將網際網路相關處理程序放在執行緒中處理，算是相當常見的處理模式。而執行緒中必須透過 android.os.Handler 來進行前景元件的異動，因此先來定義一個自訂的 Handler 類別。

```
private class MyHandler extends Handler {  
    @Override  
    public void handleMessage(Message msg) {  
        super.handleMessage(msg);  
        switch(msg.what){  
            case 1:  
                pDialog1.show();  
                break;  
            case 2:  
                if (pDialog1.isShowing()) {  
                    pDialog1.dismiss();  
                }  
                break;  
        }  
    }  
}
```

4

執行緒類別：(handler 為上述之 MyHandler 類別物件)

```
private class MyTask1 extends Thread {  
    @Override  
    public void run() {  
        handler.sendEmptyMessage(1);  
        for (int i=0; i<100; i++){  
            try {  
                Thread.sleep(100);  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
            handler.sendEmptyMessage(2);  
        }  
    }  
}
```

當執行緒開始執行的時候，handler 物件會送值為 1 的 which 值，並進行 ProgressDialog 物件的 show() 方法開始呈現 ...



若可以掌握進度，則呼叫 `setProgressStyle()` 方法時傳遞 `ProgressDialog.STYLE_HORIZONTAL`。

```
.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
```

另外撰寫一個執行緒，與上個範例不同之處是在執行中隨時將 i 值包在 `Message` 物件中的 `Bundle`，並由 `handler` 物件傳送處理。

```
private class MyTask2 extends Thread {
    @Override
    public void run() {
        handler.sendEmptyMessage(3);
        for (int i = 0; i < 100; i++) {
            try {
                Message msg = new Message();
                msg.what = 5;
                Bundle data = new Bundle();
                data.putInt("progress", i);
                msg.setData(data);
                handler.sendMessage(msg);

                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        handler.sendEmptyMessage(4);
    }
}
```

改寫上述的 MyHandler 類別定義：

```
case 3:  
    pDialog2.setMax(100);  
    pDialog2.show();  
    break;  
case 4:  
    if (pDialog2.isShowing()) {  
        pDialog2.dismiss();  
    }  
    break;  
case 5:  
    if (pDialog2.isShowing()) {  
        pDialog2.setProgress(msg.getData().getInt("progress"));  
    }  
    break;  
}
```

即可呈現如下結果：

4



4-5 通知列處理模式

在行動裝置的通知列通常用來提醒使用者發生了特定的事件通知，在發出通知的當時會有特殊的字串顯示，並可以帶有其他提示音、振動或是閃光強化通知的效果，達到提醒重要通知的目的。大部分發出通知的狀況通常是在 App 的背景中處理，當使用者拉下通知列後，觸摸該通知事項時，可以將使用者帶回 App 中特定的前景頁面。

■ 版本差異 ■

通知基本的處理程序大致上有幾個不同 API Level 的處理模式。

- API Level 11 之前
- API Level 11 之後
- API Level 16+

先來了解共同的處理觀念。

1. Notification 的處理方式必須透過 NotificationManager 物件負責管理
2. 透過呼叫 getSystemService(NOTIFICATION_SERVICE) 取得
3. 建構 Notification 物件實體
4. 建構通知啟動的 Intent 物件實體
5. 以 Intent 物件實體建構出 PendingIntent 物件實體
6. 由 NotificationManager 呼叫 notify() 方法，將 Notification 物件實體發出通知

■ API Level 11 之前 ■

```
private void createNotification(){
    // 取得 NotificationManager 物件
    NotificationManager mNotificationManager =
        (NotificationManager) getSystemService(NOTIFICATION_SERVICE);

    // 設定通知的圖示，通知提示，通知顯示的時間
    Notification notification = new Notification(R.drawable.tab10, "重要通知",
        System.currentTimeMillis());

    // 通知啟動的 Intent 物件
    Intent notificationIntent = new Intent(this, Page2.class);
    notificationIntent.putExtra("which", 4); // 傳遞資料過去

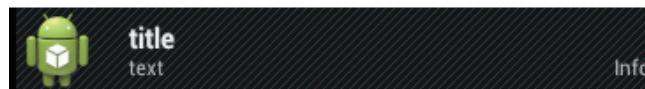
    // 以 Intent 來取得 PendingIntent 物件
    PendingIntent contentIntent = PendingIntent.getActivity(this, 0,
        notificationIntent, 0);

    notification.setLatestEventInfo(this, "期末考快被當", "期末考如未達 90
    分，總成績將會被當", contentIntent);
}
```

```
// 發出通知  
mNotificationManager.notify(1, notification);  
}
```

API Level 11+

本書重點放在 API Level 11+ 及 API Level 16+. 假設發出如下圖的通知：



先開發設計一個 Activity，用來當產生通知事件之後，使用者點按後所出現的畫面。畫面如下：

4

```
activity_my_notice_page.xml  
  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical"  
    >  
  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Notice Page" />  
  
</LinearLayout>
```

以及對應的 Activity：

```
MyNoticePage.java  
  
package tw.brad.android.book.MyNotification;  
  
import android.os.Bundle;  
import android.app.Activity;  
import android.view.Menu;  
  
public class MyNoticePage extends Activity {
```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_my_notice_page);
}

}

```

而回到主程式中，產生一個 Intent，用來指定跳到該 Activity。

```
Intent intent = new Intent(this, MyNoticePage.class);
```

透過 TaskStackBuilder 來產生一個 PendingIntent 物件實體。

```

TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
stackBuilder.addParentStack(MyNoticePage.class);
stackBuilder.addNextIntent(intent);
PendingIntent pending = stackBuilder.getPendingIntent(0, PendingIntent.
FLAG_UPDATE_CURRENT);

```

參數說明：

1. 本 Activity 物件實體
2. RequestCode，用來辨別呼叫 Intent.
3. 前述的 Intent 物件實體
4. Flag
 - PendingIntent.FLAG_ONE_SHOT
 - PendingIntent.FLAG_NO_CREATE
 - PendingIntent.FLAG_CANCEL_CURRENT
 - PendingIntent.FLAG_UPDATE_CURRENT

建構 Notification.Builder 物件實體

```
Notification.Builder builder = new Notification.Builder(this);
```

開始進行設定：

```
builder.setTicker("重要的通知");
builder.setAutoCancel(true);
builder.setSmallIcon(R.drawable.ic_launcher);
builder.setContentInfo("Info");
builder.setContentText("text");
builder.setContentTitle("title");
builder.setContentIntent(pending);
builder.setWhen(System.currentTimeMillis() + 10000); // 延遲 10 秒
builder.setContentIntent(pending);
```

進行建立 Notification 物件實體：

API Level 11+ (亦即 Android 3.0 以上)

```
Notification notification = builder.getNotification();
```

若是 API Level 16+ (亦即 Android 4.1.2 以上)

4

```
Notification notification = builder.build();
```

最後，以 `getSystemService()` 呼叫出 `NotificationManager` 物件實體，以發出通知。

```
NotificationManager mgr = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
mgr.notify(0, notification);
```

| 應用場合 |

以上是基本處理方式，但是許多狀況都會與 Service 或是背景執行緒搭配使用。當 App 已經從前景執行狀態離開，而背景的 Service 正在執行週期任務，例如每 5 分鐘連線網站伺服器，檢查是否有更新資料。如果有更新異動資料內容，馬上傳遞到行動裝置，此時也發出通知，導引使用者點擊通知內容，直接帶到指定 Activity 來呈現更新資料。

MEMO

05

Chapter

進階程序運作原理與應用

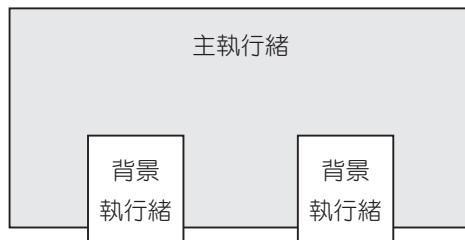
- 5-1 多重執行緒 Thread
- 5-2 定時及週期任務 Timer & TimerTask
- 5-3 同步任務 AsyncTask
- 5-4 倒數計時器





5-1 多重執行緒 Thread

在一個 App 的執行期間，當有多樣任務工作可能會同時進行的時候，就會使用到多重執行緒的處理方式。而多重執行緒的原理是將要進行的程式片段，放在執行背景中處理，而前景為主程式，也就是主執行緒。前景只能有一個，而背景執行緒可以有多個同時進行。



譬如在一個遊戲 App 中，主角是炸彈超人的執行緒，而在關卡中還有三個壞蛋的執行緒，分別在背景進行位移運算，並不會因為炸彈超人行進中，而所有壞蛋就停止其行進的動作，應該是各自有不同的執行緒物件分別處理各自的不同情境狀態。或是在打磚塊遊戲中，玩家拿到寶物後開始發射子彈，而每個子彈也都是個別獨立的執行緒物件來處理，也不會影響目前正在碰撞中的球體行進。

|| 開發重點觀念 ||

很常見的處理方式就是以內部類別來開發，這樣會比較容易存取外部類別物件的成員。

1. 自訂類別繼承 `java.lang.Thread`
2. Override 其 `run()`
3. 將該執行緒生命週期的程式定義撰寫在 `run()` 方法中

```

private class MyThread1 extends Thread {
    @Override
    public void run() {
        for (int i=0; i<10; i++) {
            Log.i("brad", "i = " + i);
            try {
                Thread.sleep(200);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

```

上述程式片段為一個內部類別的執行緒類別，生命週期執行一段 for 迴圈，每一圈執行 Log 顯示變數 i 值，執行後使其呼叫 Thread 的 static 方法 sleep()，傳入參數為千分之一秒為單位的值，讓該執行緒暫停至少 0.2 秒（值為 200），再繼續下一個迴圈。

先將該物件建構：

5

```
MyThread1 t1 = new MyThread1();
```

如果只是以一般物件方法呼叫 run()，則並不具有執行緒的生命週期特徵。而必須呼叫該物件的 start() 才會表現出生命週期的特徵，以執行 run() 方法中所定義的程式區塊。

不具生命週期的呼叫：

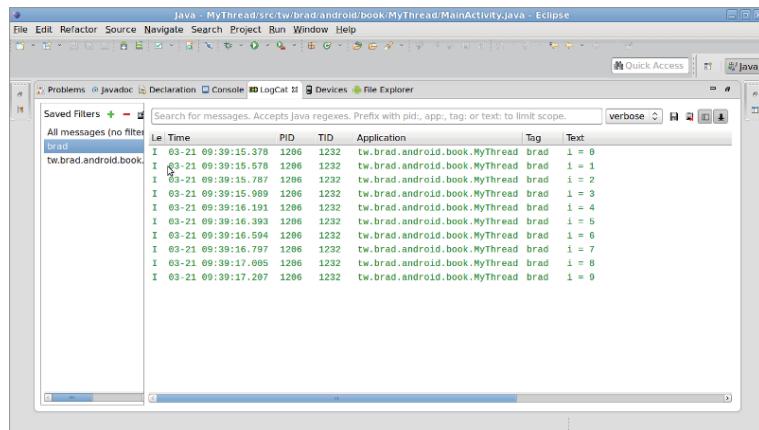
```
t1.run();
```

具有生命週期的呼叫：

```
t1.start();
```

雖然具有執行緒生命週期特徵，但是只能呼叫一次，當 run() 方法中的程式區塊執行完畢後，就結束其執行緒的生命週期，如果重複呼叫 start()，則將會在第二次呼叫執行後拋出 RuntimeException。

執行之後會在 LogCat 中看到狀況如下：



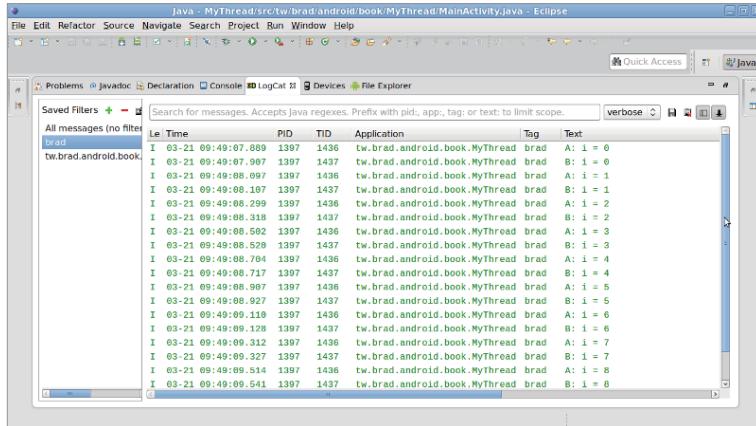
如果同時啟動兩個相同類別的執行緒物件來作實驗，並改寫執行緒物件的建構式，傳遞一個字串屬性。

```
private class MyThread1 extends Thread {
    String name;
    MyThread1(String n) {
        name = n;
    }
    @Override
    public void run() {
        for (int i = 0; i < 10; i++) {
            Log.i("brad", name + ": i = " + i);
            try {
                Thread.sleep(200);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

接著建構出兩個物件實體，並分別呼叫 start() 方法：

```
private void doThread1() {
    MyThread1 t1 = new MyThread1("A");
    MyThread1 t2 = new MyThread1("B");
    t1.start();
    t2.start();
}
```

執行之後的狀況可能如下：(讀者實作結果不一定會完全相同的順序性)



存取 View 元件

如果在 run() 方法中存取 View 元件，例如呼叫 TextView 物件實體的 setText() 方法。

5

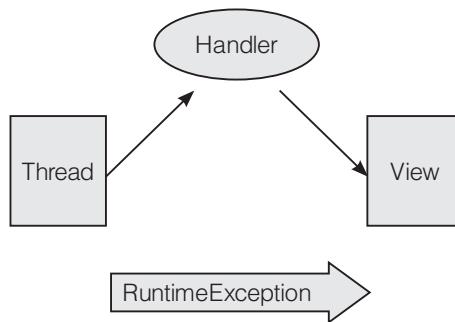
```
private class MyThread1 extends Thread {
    String name;

    MyThread1(String n) {
        name = n;
    }

    @Override
    public void run() {
        for (int i = 0; i < 10; i++) {
            Log.i("brad", name + ": i = " + i);
            msg.setText(name + ": i = " + i);
            try {
                Thread.sleep(200);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

上例中的 msg 為一個 TextView 物件。

這樣的狀況下，並不會發生編譯失敗，但是卻會在執行時期出現 RuntimeException。



多重執行緒在 Java Application 中已經是非常常見的運用，在 Android 中的使用方式是完全一樣，唯一要特別注意的是，在執行緒中是無法直接對 View 元件作任何操作，雖然在編譯階段沒有任何語法錯誤，卻會在執行時期拋出 Running Exception。因此可以透過繼承 android.os.Handler 類別的子類別物件來處理。

```

private class MyHandler extends Handler {
    @Override
    public void handleMessage(Message message) {
        msg.setText(t1.name + ": i = " + message.what);
    }
}
  
```

- t1 為上例中 MyThread1 執行緒類別的物件實體
- msg 為前景畫面中的 TextView 類別的物件實體
- message 為傳遞進來的 Message 物件實體

而在上例的執行緒中修改如下：

```

@Override
public void run() {
    for (int i = 0; i < 10; i++) {
  
```

```

        Log.i("brad", name + ": i = " + i);
        //msg.setText(name + ": i = " + i);
        handler.sendEmptyMessage(i);
        try {
            Thread.sleep(200);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

```

- handler 為上文中 MyHandler 類別物件
- 呼叫 MyHandler 類別物件方法 sendEmptyMessage()，傳遞參數 i 值
- 則將會觸發 handler 物件中的 handleMessage() 方法

提早結束執行緒的生命週期

當執行緒生命週期尚未執行完畢，可能會因其他因素如使用者取消任務執行或是特定事件觸發，而必須提早結束該執行緒生命週期，則可以使用以下模式進行開發。

5

- 在 run() 方法中以 try...catch 捕捉 InterruptedException
- 在 catch 程式區塊中提早結束，例如迴圈結構的 break 敘述句或是直接 return
- 而觸發 InterruptedException 方式就是呼叫該物件實體的 interrupt()

將上述範例 MyThread1 類別中的 run() 修改如下：

```

for (int i = 0; i < 100; i++) {
    Log.i("brad", name + ": i = " + i);
    //msg.setText(name + ": i = " + i);
    handler.sendEmptyMessage(i);
    try {
        Thread.sleep(200);
    } catch (InterruptedException e) {
        break;
    }
}

```

當執行緒物件 t1 呼叫了 interrupt()，就會提早脫離迴圈結構，而結束執行 run()，也就提早結束其生週期的表現了。

```
private void stopThread1(){
    if (t1 != null && t1.isAlive()) {
        t1.interrupt();
    }
}
```

|| 另外一種開發方式 ||

除了上述的自訂類別繼承自 java.lang.Thread 外，也常被討論的方式就是實作 (implements)java.lang.Runnable 介面的自訂類別。這樣處理執行緒的優點在於該自訂類別尚可繼承其他父類別，而相較於上述的繼承 java.lang.Thread 的方式，就無法繼承其他父類別。

定義類別方式：

```
private class MyRunnable1 implements Runnable {
    String name;

    MyRunnable1(String n) {
        name = n;
    }

    @Override
    public void run() {
        for (int i = 0; i < 100; i++) {
            Log.i("brad", name + ": i = " + i);
            //msg.setText(name + ": i = " + i);
            handler.sendMessage(i);
            try {
                Thread.sleep(200);
            } catch (InterruptedException e) {
                break;
            }
        }
    }
}
```

而啟動執行生命週期如下：

```
private void doRunnable1(){
    MyRunnable1 mr = new MyRunnable1("C");
    Thread t3 = new Thread(mr);
    t3.start();
}
```

重要觀念如下：

- 一個 Runnable 物件實體並非執行緒物件
- 執行緒物件為 Thread，也就是上例中的 t3
- t3 執行緒物件是傳入一個 Runnable 物件實體來建構的
- 所以是由 t3 呼叫 start() 方法來啟動執行緒的生命週期
- 執行過程則是表現定義在 Runnable 物件的 run() 方法中



Thread.sleep() 方法的時間值，只是表示該執行緒的暫停休眠時間，並不代表恢復休眠時間後就馬上執行，也就是說時間的精準度不夠，通常是大於指定的暫停休眠時間，因此通常用於確保任務執行的完成度，例如從網路擷取檔案或是資料的程序。如果要求較高的時間精準度，則建議使用下一小節的 Timer 來處理較佳。

5



5-2 定時及週期任務 (Timer & TimerTask)

在 App 中處理時間週期相關的部份，筆者通常會使用 java.util.Timer 類別物件來運用。凡是與定時或是特定週期循環的任務，都可以以一個 Timer 物件來總管整個所有不同的時間相關任務。而可以被交給 Timer 物件管理的任務，則必須是 java.util.TimerTask 類別的物件，但是 TimerTask 為抽象類別，因此會以自訂類別繼承 TimerTask 類別，並 override 其 run() 來實作定義任務的程式區塊。

通常處理模式：

1. 建構 Timer 物件實體
2. 自訂類別繼承 TimerTask
3. Override 其 public void run() 方法
4. 開發定義任務程式區塊
5. 建構出 TimerTask 物件實體
6. Timer 物件呼叫 schedule() 方法，傳入 TimerTask 物件實體，及時間週期相關參數

與 Thread 處理模式不同之處，在於時間的管控是由 Timer 物件負責，而且只負責時間任務控制而已；而任務內容就完全不涉及與時間相關，單純的開發撰寫工作任務的內容即可。

建構 Timer 物件實體

```
Timer timer = new Timer();
```

假設開發一個一開始就會有一個 TextView 呈現行動裝置上目前時間的分秒，所以也會需要有一個 Handler 物件實體來處理前景元件的存取動作。

```
private class MyHandler extends Handler {
    @Override
    public void handleMessage(Message msg) {
        if (msg.what == 0) {
            Calendar now = Calendar.getInstance();
            now.setTime(new Date(System.currentTimeMillis()));
            msg2.setText(now.get(Calendar.MINUTE) + ":" + now.
                get(Calendar.SECOND));
        } else {
        }
    }
}
```

也就是接下來的開發中，會傳遞 Message 物件的 what 值為 0，接著就將目前時間呈現在一個變數名稱為 msg2 的 TextView 物件中。以下假設有一個 MyHandler 類別物件已經建構為物件變數 handler 了。

自訂一個 TimerTask 類別的子類別

```
private class MyTask1 extends TimerTask {  
    @Override  
    public void run() {  
        handler.sendEmptyMessage(0);  
    }  
}
```

其工作任務內容就是發送 Message 而已。

也在建構出物件實體為 task1 之後，將該工作任務排進 Timer 物件中

```
timer.schedule(task1, 0, 200);
```

設定一開始不延遲馬上執行任務，之後每間隔 200 千分之一秒，也就是 0.2 秒的週期再度執行一次。

再做一個不同的 TimerTask 的工作內容，也同時排在相同的 Timer 物件的時間管理中。

5

```
private class MyTask2 extends TimerTask {  
    @Override  
    public void run() {  
        handler.sendEmptyMessage((int)(Math.random()*49+1));  
    }  
}
```

只比上一個複雜一點點，隨機產生 1-49 的大樂透號碼。

也稍微修改了 MyHandler 類別中 handleMessage() 方法的程式內容。

```
@Override  
public void handleMessage(Message msg) {  
    if (msg.what == 0){  
        Calendar now = Calendar.getInstance();  
        now.setTime(new Date(System.currentTimeMillis()));  
        msg2.setText(now.get(Calendar.MINUTE) + ":" + now.  
                    get(Calendar.SECOND));  
    }else {  
        msg1.setText(":" + msg.what);  
    }  
}
```

並開發一個啟動方法與一個停止方法。

```
private void startTask2() {
    if (task2 == null) {
        task2 = new MyTask2();

        //timer.schedule(task2, 0, 1000);
        timer.scheduleAtFixedRate(task2, 0, 1000);
    }
}

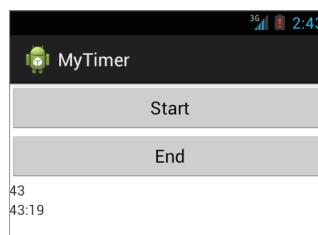
private void endTask2() {
    if (task2 != null) {
        task2.cancel();
        task2 = null;
    }
}
```

雖然是一樣的工作任務，讓使用者可以進行啟動與暫停，關鍵動作在於呼叫 TimerTask 物件的 cancel() 方法，可以使該物件結束在 Timer 中的時間週期。而如果希望一次全部在 Timer 中的所有任務都取消，則將會呼叫 Timer 物件的 cancel() 方法。

即使 App 在使用者最後按下返回鍵之後，Timer 的運作仍然在背景中持續進行。如果希望應該一併取消，則可以在 Activity 中 Override 其 finish() 方法中處理。

```
@Override
public void finish() {
    timer.cancel();
    timer = null;

    super.finish();
}
```





5-3 同步任務 AsyncTask

android.os.AsyncTask 是 android 中提供一種方便、容易使用的使用者執行緒 (UI Thread)，可以用來處理背景中執行程序，並可以將一或多個結果直接呈現在使用者介面上，而完全不需要透過 Handler 的機制。算是一種相當方便的處理機制。

使用觀念

1. 開發自訂類別繼承 AsyncTask
2. 指定其三個泛型參數型別，不可以是基本型別，若不指定，則必須使用 Void。這三項參數在開發中指定，才能善加發揮出 AsyncTask 直接而不透過 Handler 機制來處理使用者介面的優點。
3. Override 其以下方法：
 - doInBackground()
 - onCancelled()
 - onPostExecute()
 - onPreExecute()
 - onProgressUpdate()

5

生命週期

先從其生命週期的表現來觀察處理程序。因為在認識了解階段，所以三個泛型參數型別皆設定為 Void。

```
private class MyTask extends AsyncTask<Void,Void,Void> {
    @Override
    protected Void doInBackground(Void...params) {
        Log.i("brad", "doInBackground");
        return null;
    }
}
```

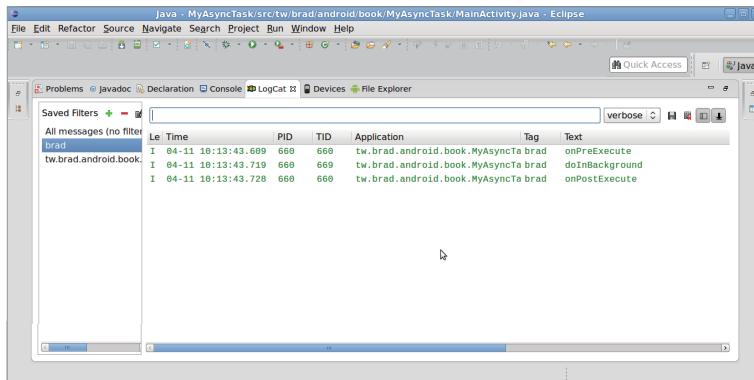
```
@Override  
protected void onCancelled() {  
    super.onCancelled();  
    Log.i("brad", "onCancelled");  
}  
  
@Override  
protected void onPostExecute(Void result) {  
    super.onPostExecute(result);  
    Log.i("brad", "onPostExecute");  
}  
  
@Override  
protected void onPreExecute() {  
    super.onPreExecute();  
    Log.i("brad", "onPreExecute");  
}  
  
@Override  
protected void onProgressUpdate(Void... values) {  
    super.onProgressUpdate(values[0]);  
    Log.i("brad", "onProgressUpdate" + values[1]);  
}  
}
```

並撰寫一段程式來進行測試，包含兩個 Button，分別作用為建構 AsyncTask 物件實體，並開始執行；另一個用來提早結束 AsyncTask 物件實體的生命週期。

```
start = findViewById(R.id.start);  
end = findViewById(R.id.end);  
  
start.setOnClickListener(new OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        task = new MyTask();  
        task.execute("Brad", null, null);  
    }  
});  
  
end.setOnClickListener(new OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        if (task != null && !task.isCancelled()) {  
            task.cancel(true);  
        }  
    }  
});
```

- start 與 end 為定義在 layout 中的 Button
- task 為宣告為 MyTask 的物件變數

執行結果將會觀察到如下：



表示在基本程序中的順序為：

5

- onPreExecute(): 執行前的準備工作
- doInBackground(): 主要執行內容
- onPostExecute(): 執行結束後的處理程序

接著在 doInBackground() 中呼叫 publishProgress()。

```
@Override
protected Void doInBackground(Void...params) {
    Log.i("brad", "doInBackground");
    publishProgress();
    return null;
}
```

發現其執行 onProgressUpdate() 方法，這就是重點所在的地方，
AsyncTask 在主要執行 doInBackground() 中可以隨時呼叫 publishProgress()，
並傳遞要表現在使用者介面的參數，而由 onProgressUpdate() 來負責更新使
用者介面。

而當呼叫 AsyncTask 物件實體的 cancel() 方法則會使其提早結束生命週期後，執行 onCancelled() 方法，通常用來辨別處理提早結束的程序使用。

|| 定義泛型參數 ||

1. 定義用來傳遞給 `doInBackground()` 方法的參數型別，通常會是在執行中會用到外部傳遞給執行緒的參數。
2. 定義 `doInBackground()` 中呼叫 `publishProgress()` 時傳遞的參數型別，通常會是表現在使用者介面的資料型別。
3. 定義最後結果的資料型別。

|| 基本開發程序 ||

開發程序只要觀念正確，就會非常簡單容易。

假設會傳遞多個 `String` 物件參數，並會傳回 `String` 物件參數結果，中間過程傳遞多個 `Integer` 物件參數。所以：

```
private class MyTask extends AsyncTask<String, Integer, String>
```

接著定義 `doInBackground()`：

```
protected String doInBackground(String... names)
```

接收參數型別與宣告第三個泛型型別是一樣的，而傳回型別與第三個泛型型別是一樣的。

當執行完畢之後，定義 `onPostExecute()` 方法：

```
protected void onPostExecute(String result)
```

接收參數型別與宣告第三個泛型型別是一樣的。

處理執行過程的顯示使用者介面：

```
protected void onProgressUpdate(Integer... values)
```

接收參數型別與宣告第二個泛型型別是一樣的。

程式架構

整體程式碼架構如下：

```
activity_main.xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >
    <Button
        android:id="@+id/start"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Start"
        />

    <Button
        android:id="@+id/end"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="End"
        />

    <TextView
        android:id="@+id/msg"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        />
</LinearLayout>
```

5

而主要程式如下：

```
MainActivity.java
package tw.brad.android.book.MyAsyncTask;

import android.app.Activity;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.TextView;

public class MainActivity extends Activity {
    private View start, end;
    private TextView mesg;
```

```
private MyTask task;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    start = findViewById(R.id.start);
    end = findViewById(R.id.end);
    mesg = (TextView) findViewById(R.id.msg);

    start.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            task = new MyTask();
            // 傳遞多個參數給 AsyncTask 運作處理
            task.execute("Brad", "Vivian", "Daniel");
        }
    });

    end.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            // 提早結束 AsyncTask
            if (task != null && !task.isCancelled()) {
                task.cancel(true);
            }
        }
    });
}

private class MyTask extends AsyncTask<String, Integer, String> {
    private String[] name;
    private boolean isOver;

    @Override
    protected String doInBackground(String... names) {
        name = names;
        for (int i = 0; i < 20; i++) {
            try {
                // 傳遞三個參數處理使用者介面
                publishProgress(i, i + 100, i+200);
                Thread.sleep(500); // 暫停休眠

                // 如果提早結束
                if (isCancelled()) {
                    isOver = true;
                    break;
                }
            }
        }
    }
}
```

```
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    // 提早結束傳會 null; 否則正常下傳回結果參數
    return isOver?null:"終於執行結束";
}

@Override
protected void onCancelled() {
    super.onCancelled();
    mesg.setText("提早結束");
}

@Override
protected void onPostExecute(String result) {
    super.onPostExecute(result);
    // 將接收執行結束后的結果參數顯示在使用者介面
    mesg.setText("Result:" + result);
}

@Override
protected void onPreExecute() {
    super.onPreExecute();
}

@Override
protected void onProgressUpdate(Integer... values) {
    super.onProgressUpdate();
    int i = values[0];
    // 將執行過程的傳遞參數顯示在使用者介面
    mesg.setText(name[i%3] + " = " + values[i%3]);
}
}
```

5

5-4 倒數計時器

字面上的意思似乎用來進行倒數計時使用，例如跨年倒數計時、碼表計時等等。其實還有許多場合非常常用到，例如在許多的遊戲 App 中，當寶物或是道具出現是有時間性，出現 8 秒後自動消失，並會在消失前 3 秒以閃爍方式提醒使用者，這樣的狀況下就非常簡單的以 android.os.CountDownTimer

來處理。或是，某個關卡必須限定在 20 秒內闖關，否則就算是失敗，這樣的情境也非常適用。如果以前面小節的執行緒來處理，相對的在開發上就顯得較為複雜許多。

|| 開發模式 ||

1. 自訂類別繼承 CountDownTimer
2. Override 其建構式
3. Override 以下方法：
 - onTick(): 時間間隔的處理程序
 - onFinish(): 時間到的處理程序

建構式中所傳遞的兩個參數為：

- long 型別的總執行時間，單位為千分之一秒
- long 型別的執行間隔時間，單位為千分之一秒



而在 onTick() 和 onFinish() 中可以直接更新使用者介面，不需要透過 Handler 的機制，就是最方便的地方。

|| 直接實作練習 ||

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >
    <Button
        android:id="@+id/start"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Start"
        />
    <Button
```

```
    android:id="@+id/cancel"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Cancel"
    />

    <TextView
        android:id="@+id/tv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        />

</LinearLayout>
```

MainActivity.java

```
package tw.brad.android.book.MyCountDownTest;

import android.app.Activity;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.TextView;

public class MainActivity extends Activity {
    private View start, cancel;
    private TextView tv;
    private MyCoundDownTask mytask;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        tv = (TextView) findViewById(R.id.tv);
        start = findViewById(R.id.start);
        cancel = findViewById(R.id.cancel);

        start.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                startCountDown();
            }
        });
    }
}
```

```
cancel.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        cancelCountDown();
    }
});

private void startCountDown(){
    mytask = new MyCoundDownTask(20000, 1000);
    mytask.start();
}

private void cancelCountDown(){
    if (mytask != null){
        mytask.cancel();
        tv.setText("提早結束");
    }
}

private class MyCoundDownTask extends CountDownTimer {

    public MyCoundDownTask(long millisInFuture, long countDownInterval) {
        super(millisInFuture, countDownInterval);
    }

    @Override
    public void onTick(long millisUntilFinished) {
        tv.setText("剩下秒數：" + millisUntilFinished/1000);
    }

    @Override
    public void onFinish() {
        tv.setText("時間到");
    }
}
```

06

Chapter

選單與動作列處理

6-1 選單 Menu

6-2 動作列 Action Bar





6-1 選單 Menu

Options menu 選項選單（硬體選單鍵）

當 Activity 在執行狀態下，使用者按下行動裝置的選單鍵，通常在螢幕下方將會出現選單的功能。一般而言，用在較不常見的功能使用，例如輔助說明或是關於功能。官方建議在 Android 3+ 以後的版本，將會以 Action Bar 取代之，該部份將會在下一小節中介紹使用。

事先定義的選單內容

通常會以 XML 格式來表現出選單內容，而 XML 格式檔案會被放在專案架構下 res/ 目錄下的 menu/ 子目錄中。目前的開發環境下，當建立出一個新的 Android 專案後，都已經自動產生預設的檔案，放在 res/menu/main.xml，其內容如下：

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

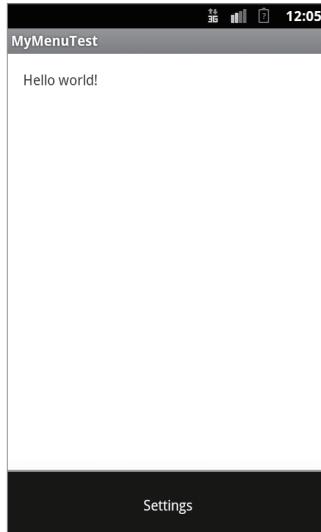
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:showAsAction="never"
        android:title="@string/action_settings"/>

</menu>
```

而在主要的 Activity 中也已經寫好該段處理方法，如下：

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}
```

直接執行後，當按下硬體選單鍵後出現如下圖：



假設想要設計出一個選單內容為：

6

- 設定
- 輔助
- 關於

則修改 menu.xml 基本如下：

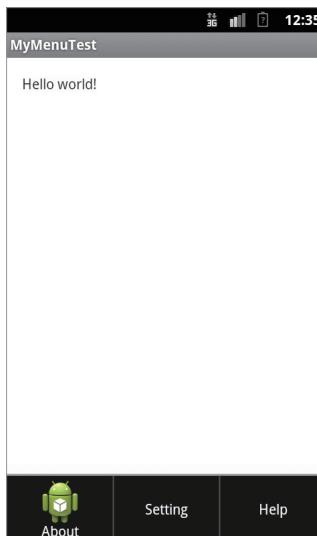
```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/action_settings"
        android:orderInCategory="200"
        android:showAsAction="never"
        android:title="Setting"
    />
    <item
        android:id="@+id/action_help"
        android:orderInCategory="400"
        android:showAsAction="never"
        android:title="Help"
    />
```

```
<item  
    android:id="@+id/action_about"  
    android:orderInCategory="100"  
    android:showAsAction="never"  
    android:title="About"  
    android:icon="@drawable/ic_launcher"  
/>  
  
</menu>
```

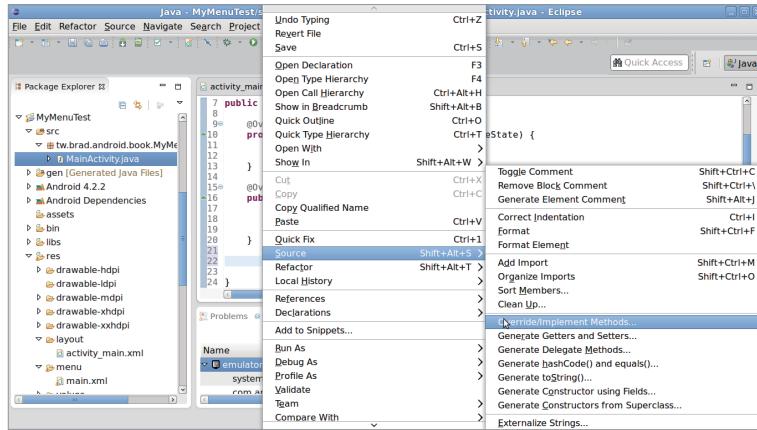
- 多加了兩個 `<item>`
- 調整了 `orderInCategory` 的值，數值表示出現的優先順序，小先大後
- 第三個 `<item>`，增設 icon，指定為 `res/drawable/` 下的 `ic_launcher` 的影像檔案

呈現如下內容：

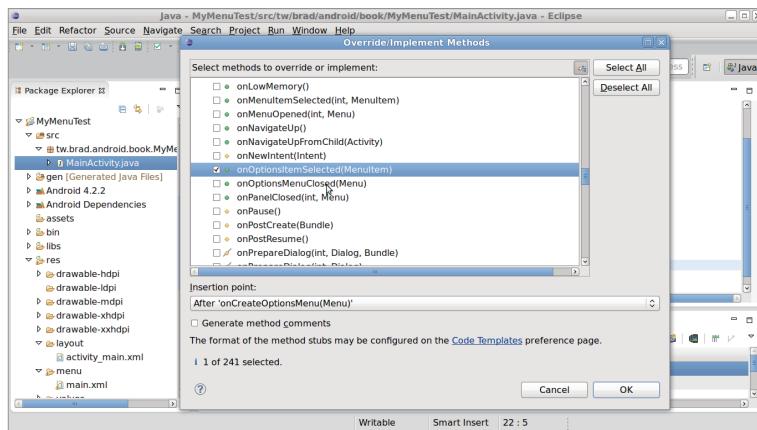


偵測使用者操作

回到程式開發工作區中，在類別中按下滑鼠右鍵出現選單，點選 Source 後再點選 Override/Implement Methods 如下畫面：



接著出現以下對話框畫面，勾選 onOptionsItemSelected()



6

程式中自動產生以下程式碼：

```

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // TODO Auto-generated method stub
    return super.onOptionsItemSelected(item);
}

```

這段方法的開發撰寫，是針對當使用者點選了選單中的特定項目之後所執行的方法。重點在於將會傳遞 MenuItem 物件的參數進來，該 MenuItem

物件即為使用者所按下的選單項目。接著透過呼叫 MenuItem 物件實體的 getItemId() 方法傳回一個整數值，代表其在 res/menu/ 下的配置資源整數值，如下進行判斷：

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_about:
            Toast.makeText(this, "選擇了 about", Toast.LENGTH_SHORT).show();
            break;
        case R.id.action_help:
            Toast.makeText(this, "選擇了 help", Toast.LENGTH_SHORT).show();
            break;
        case R.id.action_settings:
            Toast.makeText(this, "選擇了 setting", Toast.LENGTH_SHORT).show();
            break;
    }

    return super.onOptionsItemSelected(item);
}
```

Context menu 內容選單

內容選單通常始於 ListView 之類的 View 一起運作。例如當在一個 ListView 的多個選項中，使用者針對特定的選項進行長按，將會出現一個浮出選單，導引使用者進行該選項的進一步處理選擇，而每個選項都有一樣的浮出選單。假設 ListView 中呈現出的是檔案列表，使用者長按特定的檔案項目，就會浮現出一個浮出選單，導引使用者進行檔案複製，更名或是刪除。

初始環境

以下以範例實際進行開發，假設在一個 ListActivity 中呈現一個 ListView，專案結構如下處理：

res/layout/activity_main.xml 負責處理 ListView 的單一選項版面

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    >
<TextView
    android:id="@+id/filename"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="24sp"
    android:textStyle="bold"
    />
</RelativeLayout>
```

只有一個文字內容，其 id 為 filename

MainActivity 繼承 ListActivity 類別，假設資料已經處理完畢。(因為本單元並非介紹 ListView，所以資料來源是從程式中產生，實際狀況應該是從檔案系統中，撈取資料放進選單中)。

6

```
public class MainActivity extends ListActivity {
    private ListView lview;
    private SimpleAdapter adapter;
    private String[] from = {"filename"};
    private int[] to = {R.id.filename};
    private ArrayList<HashMap<String, String>> data;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        data = new ArrayList<HashMap<String, String>>();

        HashMap<String, String> d0 = new HashMap<String, String>();
        d0.put(from[0], "file0.txt");
        data.add(d0);
        HashMap<String, String> d1 = new HashMap<String, String>();
        d1.put(from[0], "file1.txt");
        data.add(d1);
        HashMap<String, String> d2 = new HashMap<String, String>();
        d2.put(from[0], "file2.txt");
        data.add(d2);
        HashMap<String, String> d3 = new HashMap<String, String>();
        d3.put(from[0], "file3.txt");
        data.add(d3);

        adapter = new SimpleAdapter(this, data, R.layout.activity_
```

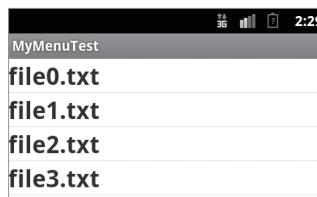
```

        main, from, to);

        lview = getListView();
        lview.setAdapter(adapter);
    }
}

```

因此呈現如下圖的初始狀態：



建立 ContextMenu 選單

開始處理 ContextMenu 的長相，在 res/menu/ 子目錄下，新建一個選單名為 cmenu.xml

```

<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/item1"
        android:title="編輯"
    />
    <item
        android:id="@+id/item2"
        android:title="刪除"
    />
    <item
        android:id="@+id/item3"
        android:title="列表"
    />
    <item
        android:id="@+id/item4"
        android:title="測試"
    />
</menu>

```

- Id 用來設定資源內容
- title 用來設定顯示內容

回到程式中，針對 ListView 或是 GridView 中進行註冊 ContextMenu，也就是呼叫 registerForContextMenu() 方法，例如：

```
registerForContextMenu(lview);
```

在 Activity 類別中，Override 其 onCreateContextMenu() 方法：

```
@Override  
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuItemInfo  
menuInfo) {  
    super.onCreateContextMenu(menu, v, menuInfo);  
    getMenuInflater().inflate(R.menu.cmenu, menu);  
}
```

開發到這邊就已經可以先執行查看狀況：



偵測使用者的操作行為

處理手法與 OptionMenu 的處理手法類似即可。

```
@Override  
public boolean onContextItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.item1:  
            Toast.makeText(this, "編輯", Toast.LENGTH_SHORT).show();  
    }  
}
```

```

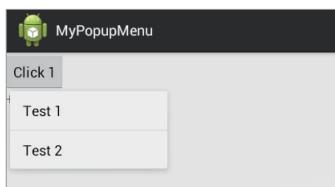
        break;
    case R.id.item2:
        Toast.makeText(this, "刪除", Toast.LENGTH_SHORT).show();
        break;
    case R.id.item3:
        Toast.makeText(this, "列表", Toast.LENGTH_SHORT).show();
        break;
    case R.id.item4:
        Toast.makeText(this, "測試", Toast.LENGTH_SHORT).show();
        break;
    }
    return super.onContextItemSelected(item);
}

```

|| Popup menu 彈出式選單 ||

彈出式選單用在 View 元件上面，使用者按下該 View 元件之後，就會出現一個彈出的選單。不過要特別注意的是，該項功能適用於 API Level 11+，也就是 Android 3.0+ 之後的版本。

先看到其基本的處理結果如下圖：



處理程序

- 在 res/menu/ 子目錄下建立選單 XML 選單內容檔案
- 設定特定的 View 的按下事件
- 建構 PopupMenu
- 指定 res/menu/ 下的 XML 選單內容檔案
- 設定特定選項的事件

在 res/menu/ 子目錄下建立選單 XML 選單內容檔案：res/menu/pmenu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@+id/item1" android:title="Test 1"></item>
    <item android:id="@+id/item2" android:title="Test 2"></item>
</menu>
```

主要版面 activity_main.xml，指定 Button 的按下 onClick 為呼叫 show PopupMenu 方法

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >

    <Button
        android:id="@+id/click1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click 1"
        android:onClick="showPopupMenu"
        />

    <TextView
        android:id="@+id/info"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        />

</LinearLayout>
```

6

回到主要程式中：

```
public void showPopupMenu(View v) {
    PopupMenu popup = new PopupMenu(this, v);
    MenuInflater inflater = popup.getMenuInflater();
    inflater.inflate(R.menu.pmenu, popup.getMenu());

    popup.show();
}
```

再來進行按下偵測，處理手法與之前 OptionMenu 類似：

```
public void showPopupMenu(View v) {
    PopupMenu popup = new PopupMenu(this, v);
    MenuInflater inflater = popup.getMenuInflater();
    inflater.inflate(R.menu.pmenu, popup.getMenu());

    popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
        @Override
        public boolean onMenuItemClick(MenuItem item) {
            switch (item.getItemId()) {
                case R.id.item1:
                    Toast.makeText(MainActivity.this, "編輯",
Toast.LENGTH_SHORT).show();
                    break;
                case R.id.item2:
                    Toast.makeText(MainActivity.this, "刪除",
Toast.LENGTH_SHORT).show();
                    break;
            }

            return false;
        }
    });
}

popup.show();
}
```

6-2 動作列 Action Bar

在一般建立專案精靈對話框之後，其實已經含進了在上一小節中所介紹的 OptionMenu，只需要再略作些修改就是 Action Bar。

先將 res/menu/ 子目錄下的 main.xml 進行修改如下：

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

<item
    android:id="@+id/action_share"
    android:icon="@drawable/share"
```

```
        android:showAsAction="ifRoom|withText"
        android:title=" 分享 "
    />
<item
    android:id="@+id/action_gift"
    android:icon="@drawable/gift"
    android:showAsAction="ifRoom|withText"
    android:title=" 禮物 "
    />
<item
    android:id="@+id/action_delete"
    android:icon="@drawable/delete"
    android:showAsAction="ifRoom|withText"
    android:title=" 刪除 "
    />
<item
    android:id="@+id/action_eraser"
    android:icon="@drawable/eraser"
    android:showAsAction="ifRoom|collapseActionView"
    android:title=" 清除 "
    />
<item
    android:id="@+id/action_save"
    android:icon="@drawable/save"
    android:showAsAction="ifRoom|collapseActionView"
    android:title=" 存檔 "
    />
<item
    android:id="@+id/action_open"
    android:icon="@drawable/open"
    android:showAsAction="collapseActionView|ifRoom"
    android:title=" 開啟 "
    />
<item
    android:id="@+id/action_help"
    android:icon="@drawable/help"
    android:showAsAction="collapseActionView"
    android:title=" 輔助 "
    />
</menu>
```

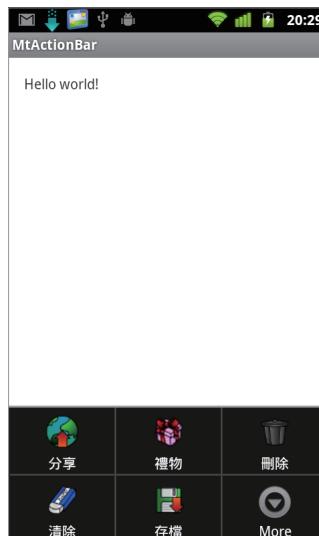
6

差別在於 showAsAction 的設定值：

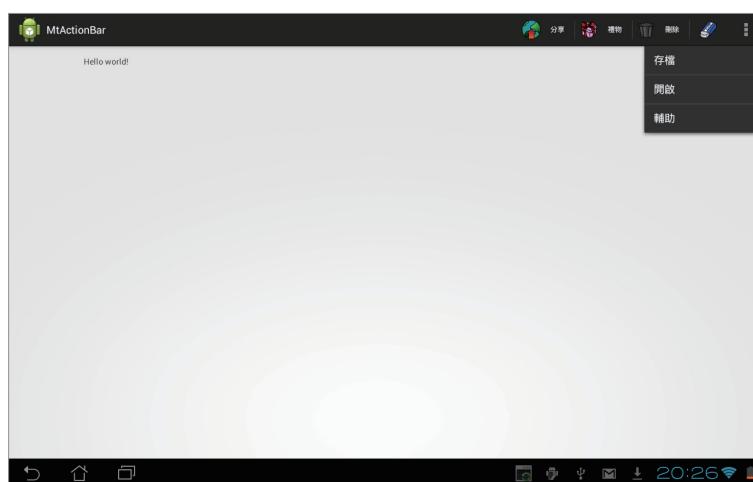
- never：不顯示在 Action Bar
- always：盡量顯示在 Action Bar，空間不夠仍然要用硬體選單鍵

- ifRoom：如果 Action Bar 的空間還夠就顯示
- with Text：也顯示文字內容（通常在平板電腦上有作用）
- collapseActionView：折疊顯示，折疊展開後顯示文字內容

在 API Level 3 以前的裝置，仍然視為 OptionMenu 處理，畫面如下：



在平板電腦的狀況如下圖：



在 API Level 3+ 的手機的狀況如下圖：



而使用者選取項目的判斷方式，也與 OptionMenu 相同。

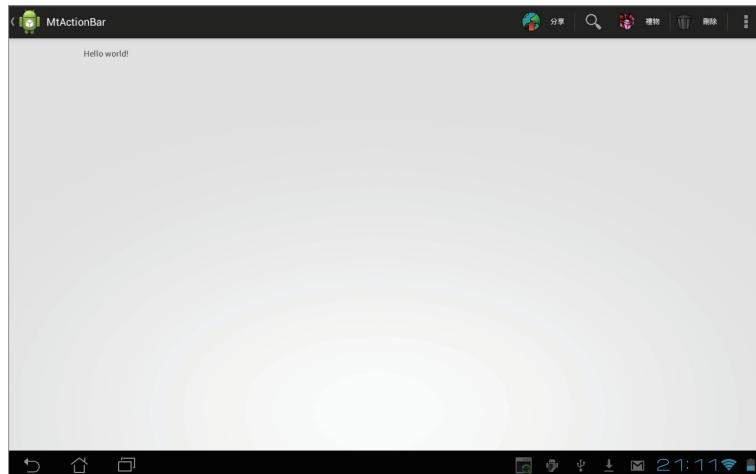
6

```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    if (item.getItemId() == R.id.action_gift){  
        Log.i("brad", "Gift");  
    }else {  
        Log.i("brad", "Other");  
    }  
    return super.onOptionsItemSelected(item);  
}
```

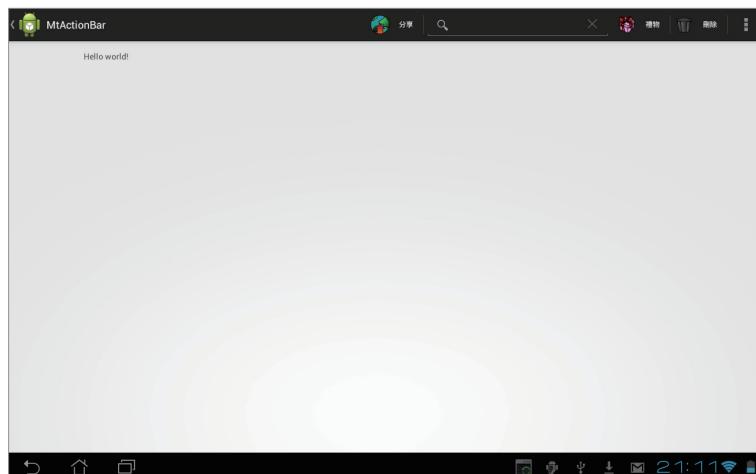
如果在 menu 中，要加上搜尋的功能，可以在 res/menu/main.xml 中增加一個 item 如下：

```
<item  
    android:id="@+id/action_search"  
    android:showAsAction="ifRoom|withText"  
    android:actionViewClass="android.widget.SearchView"  
    android:title="搜尋"  
/>
```

就會有以下的效果呈現：



當使用者按下搜尋圖示後，將會展開輸入搜尋文字。



此時應該在程式中針對 SearchView 進行設定，先透過 menu 物件實體呼叫 `findItem()` 方法找出 id 為 `action_search` 的 item，再呼叫 `getActionView()` 方法後將傳回值強制轉型回 SearchView 物件實體。

```
SearchView searchView = (SearchView)menu.findItem(R.id.action_search).  
getActionView();
```

設定帶有送出查詢按鈕。

```
searchView.setSubmitButtonEnabled(true);
```

設定當按下送出按鈕後，接收使用者輸入的字串。

```
searchView.setOnQueryTextListener(new OnQueryTextListener() {  
    @Override  
    public boolean onQueryTextSubmit(String query) {  
        Log.i("brad", "查詢字串：" + query);  
        return false;  
    }  
  
    @Override  
    public boolean onQueryTextChange(String newText) {  
        return false;  
    }  
});
```

再來增加設定一個共用提供者，可以將資料由使用者決定給特定的應用程式來處理。先在 res/menu/main.xml 中增設一個 item。

6

```
<item android:id="@+id/menu_share"  
      android:title="Share"  
      android:showAsAction="ifRoom"  
      android:actionProviderClass="android.widget.ShareActionProvider" />
```

回到程式的處理手法與 SearchView 類似。(API Level 14+ 適用)

```
// API Level 14+  
ShareActionProvider sap = (ShareActionProvider)menu.findItem(R.id.menu_share).  
.getActionProvider();  
sap.setShareIntent(getDefaultShareIntent());
```

而 getDefaultShareIntent() 方法是自訂設定共用提供者。

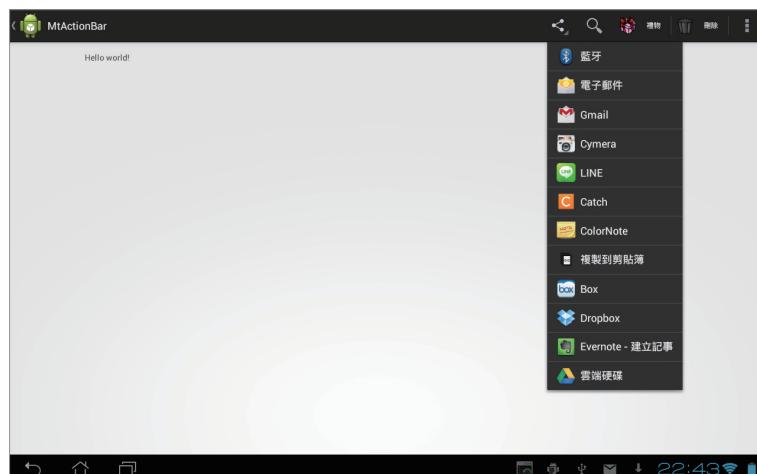
```
private Intent getDefaultShareIntent(){  
    Intent intent = new Intent(Intent.ACTION_SEND);  
    intent.setType("text/plain");
```

```

        intent.putExtra(Intent.EXTRA_SUBJECT, "SUBJECT");
        intent.putExtra(Intent.EXTRA_TEXT,"Extra Text");
        return intent;
    }
}

```

就會依照使用者在行動裝置上的 App 安裝而訂，例如下圖。



不要有 Action Bar

```
<activity android:theme="@android:style/Theme.Holo.NoActionBar">
```

或是在程式中隱藏：

```

ActionBar actionBar = getActionBar();
actionBar.hide();

```

07

Chapter

自訂 View 與 SurfaceView

- 7-1 自訂 View: 繼承 View
- 7-2 自訂 View 與觸控手勢事件處理
- 7-3 自訂 SurfaceView: 繼承 SurfaceView
- 7-4 以自訂 View 來實現手寫簽名 App 範例實作





7-1 自訂 View: 繼承 View

在 Android SDK 中的 `android.view.View` 是用來呈現給使用者的視覺介面元件，而在 API 中雖然提供非常豐富的 View 元件，包含 `TextView`、`Button` 等等，但是開發者經常可能會發現想要的外觀，操作互動等等許多方面是現有 API 無法可以替代使用的狀況，此時可以對 View 元件進行繼承及自訂內容。

首先先開發自訂類別 `extends android.view.View`：

```
package tw.brad.book.myviewtest;

import android.content.Context;
import android.util.AttributeSet;
import android.view.View;

public class PaintView extends View {

}
```

其建構有三種擇其一進行 `Override`，如果是採用只有傳遞一個 `Context` 型別的參數建構式：

```
public PaintView(Context context) {
    super(context);
}
```

則該自訂 View 類別無法在版面配置的 XML 檔案中進行配置，只能在程式執行階段加進 `ViewGroup` 之中；如果想要設計出可以在版面配置的 XML 中進行規劃，也可以在程式執行階段處理，則至少要 `Override` 的建構式是含有 `AttributeSet` 型別的參數。

```
public PaintView(Context context, AttributeSet attrs) {
    super(context, attrs);
}
```

或是

```
public PaintView(Context context, AttributeSet attrs, int defStyle) {
    super(context, attrs, defStyle);
}
```

此時可以先來進行版面配置處理。假設想要規劃設計以橫向為主的畫面，左邊有一排 Button，用來進行觸發使用者的操作，而右邊一整塊區域用來放置自訂 View。如下圖所示的初始狀態：



自訂 View 為 PaintView.java

```
package tw.brad.book.myviewtest;

import android.content.Context;
import android.util.AttributeSet;
import android.view.View;

public class PaintView extends View {

    public PaintView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }
}
```

7

版面配置 activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    android:orientation="horizontal" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="4"
        android:background="#1100ff00"
        android:orientation="vertical" >

        <Button
            android:id="@+id/clear"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Clear" />

        <Button
            android:id="@+id/undo"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Undo" />

        <Button
            android:id="@+id/redo"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Redo" />

        <Button
            android:id="@+id/chcolor"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Color" />

        <Button
            android:id="@+id/chsize"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Size" />
    </LinearLayout>

</LinearLayout
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:orientation="vertical" >

    <tw.brad.book.myviewtest.PaintView
        android:id="@+id/pview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#44ffff00" />
</LinearLayout>

</LinearLayout>
```

為了避免因為行動裝置的手持方式改變，而切換到直式顯示，可以先在 AndroidManifest.xml 中進行設定。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="tw.brad.book.myviewtest"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="tw.brad.book.myviewtest.MainActivity"
            android:label="@string/app_name" android:screenOrientation="l
            andscape">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

接下來的開發重點會先放在 PaintView 的處理。

先來了解如何取得目前的自訂 View 的寬高值為何？對於 View 而言，本來就有 getWidth() 與 getHeight() 兩個方法可以使用，就先在建構式中抓取這兩個值。

```
public PaintView(Context context, AttributeSet attrs) {
    super(context, attrs);

    int viewW = getWidth();
    int viewH = getHeight();
    Log.i("brad", viewW + " x " + viewH);

}
```

執行之後，竟然取得的兩個值皆為 0。

而 View 的顯示程序主要是在 onDraw() 方法中進行，因此來 Override 該方法，並將取得寬高值在該方法中進行看看。

```
@Override
protected void onDraw(Canvas canvas) {
    int viewW = getWidth();
    int viewH = getHeight();
    Log.i("brad", viewW + " x " + viewH);
}
```

終於得到了正確的值，也就是說在其呈現內容時才開始有了具體的寬高。如果自訂 View 每次進行呈現內容異動，還要重複抓取寬高值，在邏輯上應該不是如此。所以可以用一個 boolean 變數來判斷是否為首次執行，如果是就當時抓取寬高值，若否就不再執行抓取寬高值。

結構更改如下：

```
package tw.brad.book.myviewtest;

import android.content.Context;
import android.graphics.Canvas;
import android.util.AttributeSet;
import android.util.Log;
```

```
import android.view.View;

public class PaintView extends View {
    private boolean isInited;
    private int viewW, viewH;

    public PaintView(Context context, AttributeSet attrs) {
        super(context, attrs);

    }

    private void init(){
        viewW = getWidth();
        viewH = getHeight();

        isInited = true;
    }

    @Override
    protected void onDraw(Canvas canvas) {
        if (!isInited) init();

    }
}
```

透過傳遞開始在 onDraw() 方法中進行繪製，原理就是利用傳遞進來的
Cancas 物件進行繪製。

7

畫個圓／文字／影像資源

```
@Override
protected void onDraw(Canvas canvas) {
    if (!isInited) init();

    // 畫出一個幾何圖
    Paint paintCircle = new Paint();
    paintCircle.setColor(Color.GREEN);
    canvas.drawCircle(viewW/2, viewH/2, 40, paintCircle);

    // 畫出文字內容
    Paint paintText = new Paint();
    paintText.setColor(Color.BLACK);
    paintText.setTextSize(36);
    paintText.setTextScaleX(1.5f);
```

```

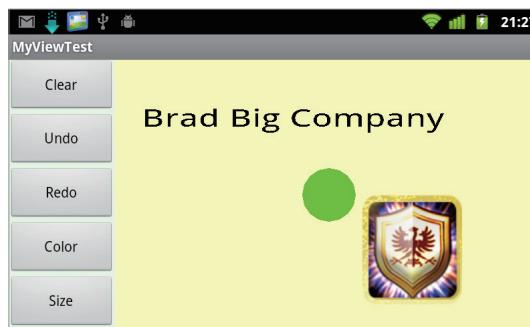
        canvas.drawText("Brad Big Company", 40, 100, paintText);

        // 畫出影像圖檔
        Bitmap shield = BitmapFactory.decodeResource(getResources(),
R.drawable.shield);
        canvas.drawBitmap(shield, viewW/2+ 50, viewH/2, null);

    }

```

結果如下：



這樣的寫法也不太好，因為在 `onDraw()` 方法中盡量能簡化到只做繪製的工作，而其他的物件建構及設定等程序，可以事先定義好即可。稍微更改結構如下：

```

package tw.brad.book.myviewtest;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.view.View;

public class PaintView extends View {
    private boolean isInited;
    private int viewW, viewH;
    private Paint paintCircle, paintText;
    private Bitmap shield;

```

```
public PaintView(Context context, AttributeSet attrs) {  
    super(context, attrs);  
}  
  
private void init() {  
    viewW = getWidth();  
    viewH = getHeight();  
  
    paintCircle = new Paint();  
    paintCircle.setColor(Color.GREEN);  
  
    paintText = new Paint();  
    paintText.setColor(Color.BLACK);  
    paintText.setTextSize(36);  
    paintText.setTextScaleX(1.5f);  
  
    shield = BitmapFactory.decodeResource(getResources(), R.drawable.shield);  
  
    isInitied = true;  
}  
  
@Override  
protected void onDraw(Canvas canvas) {  
    if (!isInitied) init();  
  
    // 畫出一個幾何圓  
    canvas.drawCircle(viewW / 2, viewH / 2, 40, paintCircle);  
  
    // 畫出文字內容  
    canvas.drawText("Brad Big Company", 40, 100, paintText);  
  
    // 畫出影像圖檔  
    canvas.drawBitmap(shield, viewW / 2 + 50, viewH / 2, null);  
}
```

7

再來練習透過執行緒來不斷重新繪製 Canvas 物件，而能夠達成視覺上動態的效果。不斷重新繪製就是呼叫 `postInvalidate()` 方法，就會再度執行 `onDraw()` 方法。週期任務用來改變其顯示位置，以上例中的圓而言，就是改變其圓心位置。

```
package tw.brad.book.myviewtest;
```

```
import java.util.Timer;
import java.util.TimerTask;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.view.View;

public class PaintView extends View {
    private boolean isInited;
    private int viewW, viewH, cx, cy;
    private Paint paintCircle, paintText;
    private Bitmap shield;
    private Timer timer;
    private MyTask task;

    public PaintView(Context context, AttributeSet attrs) {
        super(context, attrs);

        timer = new Timer();
        task = new MyTask();
    }

    private void init() {
        viewW = getWidth();
        viewH = getHeight();

        paintCircle = new Paint();
        paintCircle.setColor(Color.GREEN);

        paintText = new Paint();
        paintText.setColor(Color.BLACK);
        paintText.setTextSize(36);
        paintText.setTextScaleX(1.5f);

        shield = BitmapFactory.decodeResource(getResources(),
R.drawable.shield);

        cx = cy = 50;

        timer.scheduleAtFixedRate(task, 0, 80);
    }
}
```

```
        isInited = true;
    }

    @Override
    protected void onDraw(Canvas canvas) {
        if (!isInited) init();

        // 畫出一個幾何圓
        canvas.drawCircle(cx, cy, 40, paintCircle);

        // 畫出文字內容
        canvas.drawText("Brad Big Company", 40, 100, paintText);

        // 畫出影像圖檔
        canvas.drawBitmap(shield, viewW / 2 + 50, viewH / 2, null);
    }

    private class MyTask extends TimerTask {
        private int i;
        @Override
        public void run() {
            cx += 4;
            postInvalidate();
            if (i++ > 100) cancel();
        }
    }
}
```

7

重點整理如下：

- `postInvalidate()` 用來整個重新繪製全部呈現內容
- 預設上繪製原理依照呼叫 `canvas.drawXxx()` 的優先順序，先畫者較底層。例如背景內容先繪製，遊戲主角通常最後繪製
- 執行緒以內部類別方式處理較為容易開發，因為可以直接存取外部類別的成員



7-2 自訂 View 與觸控手勢事件處理

一般觸控事件偵測處理

再來針對使用者的觸發事件處理，最常見的就是 `onTouchEvent()`。所以就開始進行 Override 其 `onTouchEvent()` 方法：

```
@Override
public boolean onTouchEvent(MotionEvent event) {
    return super.onTouchEvent(event);
}
```

重點在於該方法被觸發後傳遞進來的 `MotionEvent` 物件實體，包含了使用者所觸摸螢幕的像素座標位置。

假設當使用者觸摸自訂 View 的位置，將繪製的圓心移至此處。

```
@Override
public boolean onTouchEvent(MotionEvent event) {
    cx = (int)event.getX();
    cy = (int)event.getY();
    postInvalidate();
    return super.onTouchEvent(event);
}
```

但是當使用者摸著螢幕沒有離開，直接滑動遊走，則沒有任何反應發生。那是因為其 `return` 值 `super.onTouchEvent(event)` 為 `false`，表示這次觸摸行為不再重複觸發。若將其 `return` 值設定為 `true`，則將會不斷地觸發該事件，輕鬆的使圓跟著手指頭移動。

手勢偵測事件處理

而另外還有一種常見的情境，就是想要設計出使用者在自訂 View 物件實體上向上下左右劃過，判斷為方向控制處理。此時就需要利用到 `android.view.GestureDetector` 物件實體進行進一步的偵測。

先進行宣告物件實體變數：

```
private GestureDetector gd;
```

開發一個自訂類別來實作 GestureDetector.OnGestureListener 介面，不過這樣處理略為麻煩，如果只是要 Override 其方法而已，那就直接繼承 android.view.GestureDetector.SimpleOnGestureListener 類別即可。(光看名稱就覺得簡單容易)

```
private class MyGDLListener extends SimpleOnGestureListener {  
    @Override  
    public boolean onDown(MotionEvent e) {  
        return true;  
    }  
  
    @Override  
    public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,  
                          float velocityY) {  
        if (Math.abs(velocityX) > Math.abs(velocityY)) {  
            if (velocityX > 10){  
                cx += 10;  
            }else if (velocityX < -10){  
                cx -= 10;  
            }  
        }else{  
            if (velocityY > 10){  
                cy += 10;  
            }else if(velocityY < -10){  
                cy -= 10;  
            }  
        }  
        postInvalidate();  
        return super.onFling(e1, e2, velocityX, velocityY);  
    }  
}
```

7

重點說明：

- 主要目的在於 Override 其 onFling() 方法
- 但是會先被 onDown() 方法偵測到，而其預設傳回 boolean 值是表示不再往下偵測，若設定傳回值為 true，才會往下偵測動作

- `onFling()` 中的第三個參數表示座標 X 軸方向移動速度，單位為每秒移動像素，向右為正值，向左為負值；第四個參數表示座標 Y 軸方向移動速度，單位為每秒移動像素，向下為正值，向上為負值

而該手勢偵測事件目前為止，完全沒有觸發點。所以，需要剛剛的 `onTouchEvent()` 方法來啟動觸發事件，因此，將其 `return` 值改為 `gd.onTouchEvent(event)` 即可。

整個架構如下：

```
package tw.brad.book.myviewtest;

import java.util.Timer;
import java.util.TimerTask;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.view.GestureDetector;
import android.view.GestureDetector.SimpleOnGestureListener;
import android.view.MotionEvent;
import android.view.View;

public class PaintView extends View {
    private boolean isInited;
    private int viewW, viewH, cx, cy;
    private Paint paintCircle, paintText;
    private Bitmap shield;
    private Timer timer;
    private MyTask task;
    private GestureDetector gd;

    public PaintView(Context context, AttributeSet attrs) {
        super(context, attrs);

        timer = new Timer();
        task = new MyTask();

        gd = new GestureDetector(context, new MyGDLListener());
    }

    private void init() {
        viewW = getWidth();
```

```
viewH = getHeight();

paintCircle = new Paint();
paintCircle.setColor(Color.GREEN);

paintText = new Paint();
paintText.setColor(Color.BLACK);
paintText.setTextSize(36);
paintText.setTextScaleX(1.5f);

shield = BitmapFactory
    .decodeResource(getResources(), R.drawable.shield);

cx = cy = 50;

timer.scheduleAtFixedRate(task, 0, 80);

isInitiated = true;
}

@Override
protected void onDraw(Canvas canvas) {
    if (!isInitiated)
        init();

    // 畫出一個幾何圓
    canvas.drawCircle(cx, cy, 40, paintCircle);

    // 畫出文字內容
    canvas.drawText("Brad Big Company", 40, 100, paintText);

    // 畫出影像圖檔
    canvas.drawBitmap(shield, viewW / 2 + 50, viewH / 2, null);
}

@Override
public boolean onTouchEvent(MotionEvent event) {
    return gd.onTouchEvent(event);
}

private class MyGDLListener extends SimpleOnGestureListener {
    @Override
    public boolean onDown(MotionEvent e) {
        return true;
    }

    @Override
    public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
        float velocityY) {
        if (Math.abs(velocityX) > Math.abs(velocityY)) {
```

```

        if (velocityX > 10) {
            cx += 10;
        } else if (velocityX < -10) {
            cx -= 10;
        }
    } else{
        if (velocityY > 10){
            cy += 10;
        } else if(velocityY < -10){
            cy -= 10;
        }
    }
    postInvalidate();
    return super.onFling(e1, e2, velocityX, velocityY);
}
}

private class MyTask extends TimerTask {
    private int i;

    @Override
    public void run() {
        cx += 4;
        postInvalidate();
        if (i++ > 100)
            cancel();
    }
}

}

```



7-3 自訂 SurfaceView：繼承 SurfaceView

上兩小節中介紹了自訂 View 的處理方式，其產生動態效果來自於週期更新繪製整個呈現的內容。對於簡單不複雜的處理上來說，算是相當容易開發。但是當要呈現的內容較為複雜，可能會有多個動態物件同時進行移動，則非常容易發生記憶體不足，或是效能不佳。

而本小節所要介紹的 `android.view.SurfaceView` 則可以針對單次要異動更新的區域進行鎖定，才開始針對該鎖定區域進行更新繪製，最後再進行解鎖。因為只需要更新異動需要異動的區域，所以在處理效能上較佳。

開發的架構與自訂 View 非常類似，此小節就不以版面配置來處理。所以直接繼承囉。

```
package tw.brad.book.mysvtest1;

import android.content.Context;
import android.view.SurfaceView;

public class MySV extends SurfaceView {

    public MySV(Context context) {
        super(context);
    }

}
```

Surface 呈現畫面的控制是由類別 SurfaceHolder 物件實體來進行處理，可以由呼叫 getHolder() 取得之。

```
public class MySV extends SurfaceView {
    private SurfaceHolder holder;

    public MySV(Context context) {
        super(context);

        holder = getHolder();
    }

}
```

7

再由 SurfaceHolder 物件實體呼叫 addCallback() 方法，指定實作 Surface Holder.Callback 介面的物件實體。下例中直接由自訂 SurfaceView 來實作即可：

```
public class MySV extends SurfaceView implements Callback {
    private SurfaceHolder holder;

    public MySV(Context context) {
        super(context);

        holder = getHolder();
        holder.addCallback(this);
    }

}
```

```

@Override
public void surfaceChanged(SurfaceHolder holder, int format, int width,
    int height) {
    // 呈現畫面異動
}

@Override
public void surfaceCreated(SurfaceHolder holder) {
    // 建立呈現的畫面
}

@Override
public void surfaceDestroyed(SurfaceHolder holder) {
    // 結束呈現的畫面
}
}

```

其處理模式繪製畫面方式，是透過 SurfaceHolder 物件實體呼叫 lockCanvas() 方法傳回被鎖定凍結的 Canvas 物件實體，再以 Canvas 物件實體進行繪製，最後再由 SurfaceHolder 物件實體呼叫 unlockCanvasAndPost() 方法，傳入已經繪製完成的 Canvas 物件實體為參數即可。

```

Canvas canvas = holder.lockCanvas(null);
canvas.drawBitmap(bg, 0, 0, null);
canvas.drawBitmap(shield, 350, 350, null);
canvas.drawCircle(x, y, 50, paint);
holder.unlockCanvasAndPost(canvas);

```

重點是當 lockCanvas 傳入參數為 null，則將回傳整個繪製區域。

而通常首次繪製的時機點是在 surfaceCreated() 方法的呼叫。

下面範例中以一個執行緒來重複繪製在特定區域：

```

package tw.brad.book.mysvtest1;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Rect;
import android.view.SurfaceHolder;
import android.view.SurfaceHolder.Callback;

```

```
import android.view.SurfaceView;

public class MySurfaceView extends SurfaceView implements Callback {
    private SurfaceHolder holder;
    private Paint paint;
    private int x, y, dy;
    private boolean isInit;
    private Bitmap bg, shield;

    public MySurfaceView(Context context) {
        super(context);

        holder = getHolder();
        holder.addCallback(this);

        paint = new Paint();
        paint.setColor(Color.YELLOW);

        x = y = 100;
        dy = 12;

        bg = BitmapFactory.decodeResource(getResources(), R.drawable.bg0);
        shield = BitmapFactory
            .decodeResource(getResources(), R.drawable.shield);

    }

    void drawCanvas() {
        Canvas canvas;

        if (!isInit) {
            canvas = holder.lockCanvas(null);
            canvas.drawBitmap(bg, 0, 0, null);
            canvas.drawBitmap(shield, 350, 350, null);
            isInit = true;
        } else {
            canvas = holder.lockCanvas(new Rect(x - 50, y - 50 -
                dy, x + 50, y + 50));
            canvas.drawBitmap(bg, 0, 0, null);
        }
        canvas.drawCircle(x, y, 50, paint);
        holder.unlockCanvasAndPost(canvas);
    }

    private class MyThread extends Thread {
        @Override
        public void run() {
            for (int i = 0; i < 40; i++) {
                y += dy;
                drawCanvas();
                try {

```

```
        Thread.sleep(40);
    } catch (InterruptedException e) {
    }
}
}

@Override
public void surfaceChanged(SurfaceHolder holder, int format, int width,
    int height) {

}

@Override
public void surfaceCreated(SurfaceHolder holder) {
    // drawCanvas();
    new MyThread().start();
}

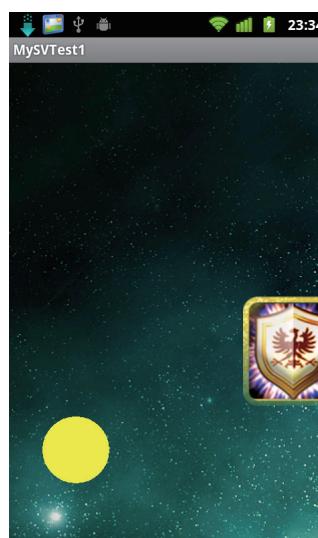
@Override
public void surfaceDestroyed(SurfaceHolder holder) {

}

}
```

其效果是一個幾何圓會從上至下落下，每次只異動更新 Rect 類別物件的區域而已。

如下圖：





7-4 以自訂 View 來實現手寫簽名 App 範例實作

直接就在這個單元來開發一個像樣的 App，也就是提供使用者可以以手勢來畫出簽名檔，並且可以進行存檔，主要是使用自訂 View 的方式來處理，外觀長相就和 7-1 小節中一開始所規劃的版面一樣，先很快說明將前置工作準備到位，就可以開始進行開發。

前置準備

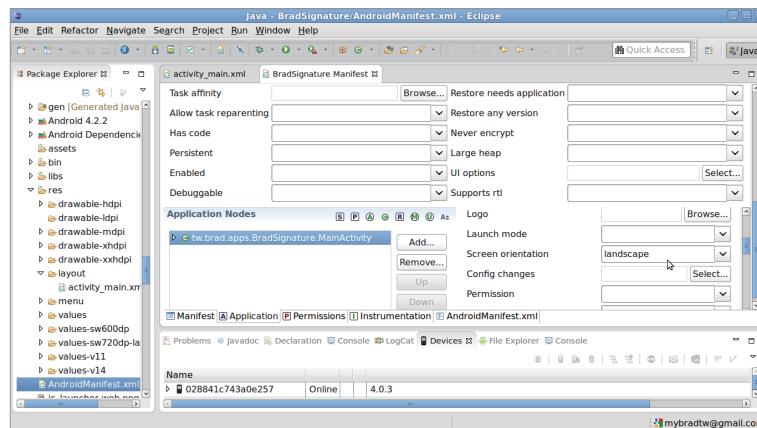
Project Name: BradSignature

Package Name: tw.brad.apps.BradSignature

Main Activity: MainActivity.java

Layout: activity_main.xml

整體操作風格是以橫向握持為主，所以直接針對該 Application 設定為橫向。



版面配置檔案：activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="4"
    android:background="#1100ff00"
    android:orientation="vertical" >

    <Button
        android:id="@+id/clear"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Clear" />

    <Button
        android:id="@+id/undo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Undo" />

    <Button
        android:id="@+id/redo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Redo" />

    <Button
        android:id="@+id/chcolor"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Color" />

    <Button
        android:id="@+id/chsize"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Size" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:orientation="vertical" >
```

```
<tw.brad.apps.BradSignature.PaintView  
    android:id="@+id/pview"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:background="#44ffff00" />  
</LinearLayout>  
</LinearLayout>
```

目前顯示結果如 7-1 小節的呈現效果。

開始處理簽名的手勢偵測處理

要畫出線條的動作是以呼叫 `drawLine()` 方法，因為需要一個 `Paint` 物件實體來處理畫筆，先在建構式中事先設定好。

再來就是 Override 自訂 View 的 `onTouchEvent()`，並將其傳回值設定為 `true`，目的是希望當使用者的一次觸摸後滑動的所有點都可以觸發取得。

目前的 `MainActivity.java` 如下：

```
package tw.brad.apps.BradSignature;  
  
import android.content.Context;  
import android.graphics.Canvas;  
import android.graphics.Color;  
import android.graphics.Paint;  
import android.util.AttributeSet;  
import android.view.GestureDetector;  
import android.view.MotionEvent;  
import android.view.View;  
  
public class PaintView extends View {  
    private Paint paintLine;  
  
    public PaintView(Context context, AttributeSet attrs) {  
        super(context, attrs);  
  
        // 設定畫筆  
        paintLine = new Paint();  
        paintLine.setColor(Color.GREEN);  
        paintLine.setStrokeWidth(4);  
    }
```

```
@Override  
public boolean onTouchEvent(MotionEvent event) {  
    return true;  
}  
  
@Override  
protected void onDraw(Canvas canvas) {  
}  
}
```

接著準備一個資料結構來存放所有經過觸摸點的線條物件，通常可以使用 `java.util.LinkedList` 類別物件實體來處理。而其元素泛型為 `java.util.HashMap` 的物件實體，以 `String` 為 Key，其 value 為 `Float`。

而使用者的觸摸動作：

- 開始觸摸：首次執行 `onTouchEvent()`，要建構出線條的資料結構物件實體，放入第一觸摸點。
- 開始滑動：放入第二及之後的觸摸點，開始針對目前線條資料重新繪製。
- 離開螢幕：`onTouchEvent()` 方法偵測到離開螢幕，也就是 `MotionEvent` 的物件方法 `getAction()` 的傳回值，如果等於 `MotionEvent.ACTION_UP`，就是離開螢幕的偵測。

修改成如下：

```
package tw.brad.apps.BradSignature;  
  
import java.util.HashMap;  
import java.util.LinkedList;  
  
import android.content.Context;  
import android.graphics.Canvas;  
import android.graphics.Color;  
import android.graphics.Paint;  
import android.util.AttributeSet;  
import android.view.MotionEvent;  
import android.view.View;  
  
public class PaintView extends View {  
    private Paint paintLine;
```

```
private LinkedList<HashMap<String,Float>> line;

public PaintView(Context context, AttributeSet attrs) {
    super(context, attrs);

    // 設定畫筆
    paintLine = new Paint();
    paintLine.setColor(Color.GREEN);
    paintLine.setStrokeWidth(4);

    // 初始建構線條資料結構物件
    line = new LinkedList<HashMap<String,Float>>();
}

@Override
public boolean onTouchEvent(MotionEvent event) {
    if (event.getAction() == MotionEvent.ACTION_UP) {
        // 離開螢幕
        line.clear();
    } else {
        // 觸摸開始
        HashMap<String,Float> point = new HashMap<String, Float>();
        point.put("x", event.getX());
        point.put("y", event.getY());
        line.add(point);
        postInvalidate();
    }
    return true;
}

@Override
protected void onDraw(Canvas canvas) {
    if (line.size() > 1) {
        // 至少有兩點才需要畫出線段
        for (int i=1; i < line.size(); i++) {
            // i = 1 表示從第二點開始處理
            HashMap<String,Float> point1 = line.get(i-1);
            // 前一點
            HashMap<String,Float> point2 = line.get(i);
            // 目前點
            canvas.drawLine(point1.get("x"), point1.get("y"),
                point2.get("x"), point2.get("y"),
                paintLine);
        }
    }
}
```

測試執行之後發現，可以畫出一條線，但是當使用者離開觸摸螢幕，該線條就消失了，但是又可以繼續畫出下一條線。原因就是從頭到尾只有使用一個 `LinkedList` 來放置一條線的資料，而觸摸離開螢幕時為了要劃下一條線，而將之前的 `LinkdeList` 的資料清除，才會有上述狀況發生。

再來就準備一個多條線的資料結構，也是 `LinkedList`，但是其存放元素為剛剛的線條的 `LinkedList`，因此會是如下的宣告處理：

```
private LinkedList<LinkedList<HashMap<String,Float>>> lines;
```

而在建構式中：

```
lines = new LinkedList<LinkedList<HashMap<String,Float>>>();
```

再來就是調整邏輯結構。

設定一個 `boolean` 變數來判斷是否處在觸摸螢幕狀態，一開始其值為 `false`，當觸摸開始，將其值設定為 `true`，也同時建構一條線的資料結構物件，並加進 `lines` 的資料結構之中；而開始滑動的判斷就是 `isTouching` 值為 `true` 的狀態，則是從 `lines` 中取出最後的線條繼續增加觸摸點資料。

```
package tw.brad.apps.BradSignature;

import java.util.HashMap;
import java.util.LinkedList;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.view.View;

public class PaintView extends View {
    private Paint paintLine;
    private LinkedList<HashMap<String,Float>> line;
    private LinkedList<LinkedList<HashMap<String,Float>>> lines;
    private boolean isTouching;
```

```
public PaintView(Context context, AttributeSet attrs) {
    super(context, attrs);

    // 設定畫筆
    paintLine = new Paint();
    paintLine.setColor(Color.GREEN);
    paintLine.setStrokeWidth(4);

    // 初始建構線條資料結構物件
    lines = new LinkedList<LinkedList<HashMap<String,Float>>>();

    isTouching = false;
}

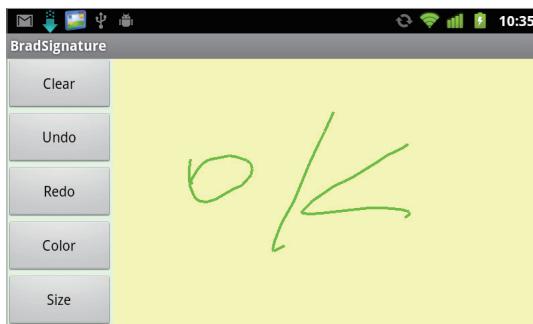
@Override
public boolean onTouchEvent(MotionEvent event) {
    if (event.getAction() == MotionEvent.ACTION_UP) {
        // 離開螢幕
        isTouching = false;
    } else {
        LinkedList<HashMap<String,Float>> line;
        if (!isTouching) {
            // 觸摸開始
            line = new LinkedList<HashMap<String,Float>>();
            lines.add(line);
            isTouching = true;
        } else {
            // 開始滑動
            line = lines.getLast();
        }
        HashMap<String,Float> point = new HashMap<String, Float>();
        point.put("x", event.getX());
        point.put("y", event.getY());
        line.add(point);
        postInvalidate();
    }

    return true;
}

@Override
protected void onDraw(Canvas canvas) {
    for (LinkedList<HashMap<String,Float>> line: lines){
        if (line.size()>1){
            // 至少有兩點才需要畫出線段
            for (int i=1; i<line.size(); i++){
                // i = 1 表示從第二點開始處理
                HashMap<String,Float> point1 = line.get(i-1);
```

```
// 前一點  
HashMap<String,Float> point2 = line.get(i);  
// 目前點  
canvas.drawLine(point1.get("x"), point1.get("y"),  
    point2.get("x"), point2.get("y"),  
    paintLine);  
}  
}  
}  
}  
}
```

終於可以畫出多條線段，如下圖：



處理外部功能

進行到要處理左側功能鍵得功能，先從清除功能下手。清除的功能對應到程式中，就是將 `lines` 中所有線段的資料清除即可，並要求重新繪製即可。此時開發一個自訂 View 的方法來負責這件事。

```
void clearDraw(){  
    lines.clear();  
    postInvalidate();  
}
```

回到 `MainActivity.java` 中設定 Button 呼叫自訂 View 的 `clearDraw()` 方法。

MainActivity.java

```

package tw.brad.apps.BradSignature;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;

public class MainActivity extends Activity {
    private PaintView pview;
    private View clear;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        pview = (PaintView)findViewById(R.id.pview);

        clear = findViewById(R.id.clear);
        clear.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                pview.clearDraw();
            }
        });
    }
}

```

7

再來處理 Undo/Redo 功能，Undo 相對 PaintView 程式中增加一個物件方法，將 lines 中的最後一條 line 元素進行移除，然後記得重新繪製即可；而 redo 是要將剛剛移除的 line 再度叫回來。所以配合這兩個功能，增設一個與 lines 一樣的物件變數 recycle，視為移除 line 物件的資源回收桶。當從 lines 移除的 line，放在 recycle；而需要 Redo 功能時，就從 recycle 中的 line 物件移除，放回 lines 中即可。

PaintView.java 中先宣告定義：

```
private LinkedList<LinkedList<HashMap<String, Float>>> recycle;
```

建構時機與 lines 同即可：

```
recycle = new LinkedList<LinkedList<HashMap<String, Float>>>();
```

開發 Undo/Redo 的物件方法：

```
// Undo 功能
void undoDraw(){
    if (lines.size()>0){
        recycle.add(lines.removeLast());
        postInvalidate();
    }
}

// Redo 功能
void redoDraw(){
    if (recycle.size()>0){
        lines.add(recycle.removeLast());
        postInvalidate();
    }
}
```

回到 MainActivity.java 中使用功能：

```
undo = findViewById(R.id.undo);
undo.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        pview.undoDraw();
    }
});

redo = findViewById(R.id.redo);
redo.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        pview.redoDraw();
    }
});
```

到此的練習，都是相當簡單的，看到如何處理自訂 View 的模式，後面其他功能在後續的章節中繼續完成。

08

Chapter

資料存取

- 8-1 偏好設定
- 8-2 內部檔案存取機制
- 8-3 外部檔案存取
- 8-4 行動裝置資料庫處理機制 SQLite
- 8-5 應用 App 資源中的資料存取資料：
遊戲關卡資料處理為例





8-1 偏好設定

在應用程式中，通常會將使用者個人化的相關設定進行儲存，以利下次執行時直接呼叫使用，使用者不會感覺到每次執行都和首次執行一樣，還要進行相關設定。例如使用者名稱，音樂音效開啟狀態或是紀錄遊戲關卡狀態等等。

`android.content.SharedPreferences` 類別是用來針對基本型別的 key-value 成對的名稱資料進行存取機制的架構，利用該類別所建構的物件可以存放的基本型別有 `boolean`、`float`、`int`、`long` 及字串物件型別的資料。

| 處理方式 |

通常在應用程式中取得 `android.content.SharedPreferences` 類別物件的方式有兩種：

- 呼叫 `getSharedPreferences(String name, int mode)`

通常適用於應用程式中會使用多個資料儲存檔案。呼叫該方法必須傳入兩個參數：

- 第一個參數項為自訂的儲存資料檔案名稱，如果指定的檔案名稱不存在，則會在實際進行存取的動作同時自動建立。
- 第二個參數為指定的操作模式，通常預設為 `MODE_PRIVATE(0)`。

- 呼叫 `getPreferences(int mode)`

通常適用於應用程式中會使用多個資料儲存檔案。所以與上個方法不同的地方就是不需要指定自訂的儲存資料檔案名稱，只需傳入一個參數項，參數為指定的操作模式，通常預設為 `MODE_PRIVATE(0)`。

兩種方法都可以在繼承 `Activity` 類別的子類別中直接呼叫，就可以取得 `SharedPreferences` 的類別物件。

■ 基本處理程序 ■

儲存資料的基本程序如下：

1. 呼叫 SharedPreferences 的類別物件的 edit() 方法取得傳回的 SharedPreferences.Editor 物件。
2. 呼叫 sharedpreferences.Editor 物件的 putXxx(String key,Xxx value) 方法將資料進行儲存。Xxx 表示基本型別的 Boolean、Float、Int、Long 及 String。
3. 最後呼叫 SharedPreferences.Editor 物件的 commit() 方法將資料存放。
4. 如果呼叫 SharedPreferences.Editor 物件的 clear() 方法將會清除資料。

■ 範例說明 ■

先宣告 SharedPreferences 和 SharedPreferences.Editor 兩個物件變數

```
private SharedPreferences sp;  
private SharedPreferences.Editor editor;
```

8

在程式中建構

```
sp = getSharedPreferences("games", MODE_PRIVATE);  
editor = sp.edit();
```

指定讀取檔案名稱為 games

讀取偏好設定資料

```
private void readPreferences(){  
    String user = sp.getString("user", "nobody");  
    boolean sound = sp.getBoolean("sound", false);  
    int stage = sp.getInt("stage", 0);  
  
    info.setText("User:" + user + "\n" +  
                "Sound:" + (sound?"On":"Off") + "\n" +  
                "Stage:" + stage);  
}
```

上述程式碼中：

- 指定讀取 Key 值為 "user" 的字串資料，呼叫 getString()，第二個參數項為當讀不到指定 Key 值時使用的預設值 "nobody"。
- 指定讀取 Key 值為 "sound" 的 boolean 資料，呼叫 getBoolean()，第二個參數項為當讀不到指定 Key 值時使用的預設值 false。
- 指定讀取 Key 值為 "stage" 的字串資料，呼叫 getInt()，第二個參數項為當讀不到指定 Key 值時使用的預設值 0。

儲存偏好設定資料

```
private void savePreferences() {
    // 清除資料
    //editor.clear();

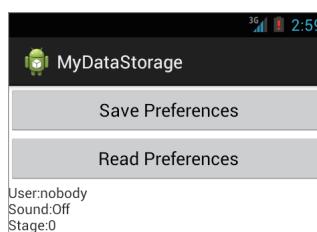
    editor.putBoolean("sound", true);
    editor.putInt("stage", 4);
    editor.putString("user", "brad");
    editor.commit();

    Toast.makeText(this, "Save OK", Toast.LENGTH_SHORT).show();
}
```

上述程式碼中，由 editor 物件實體進行資料儲存：

- 呼叫 putBoolean()，指定 Key 字串值存放 boolean 資料
- 呼叫.putInt()，指定 Key 字串值存放 int 資料
- 呼叫.putString()，指定 Key 字串值存放 String 資料
- 最後呼叫 commit() 方法進行實體檔案寫出

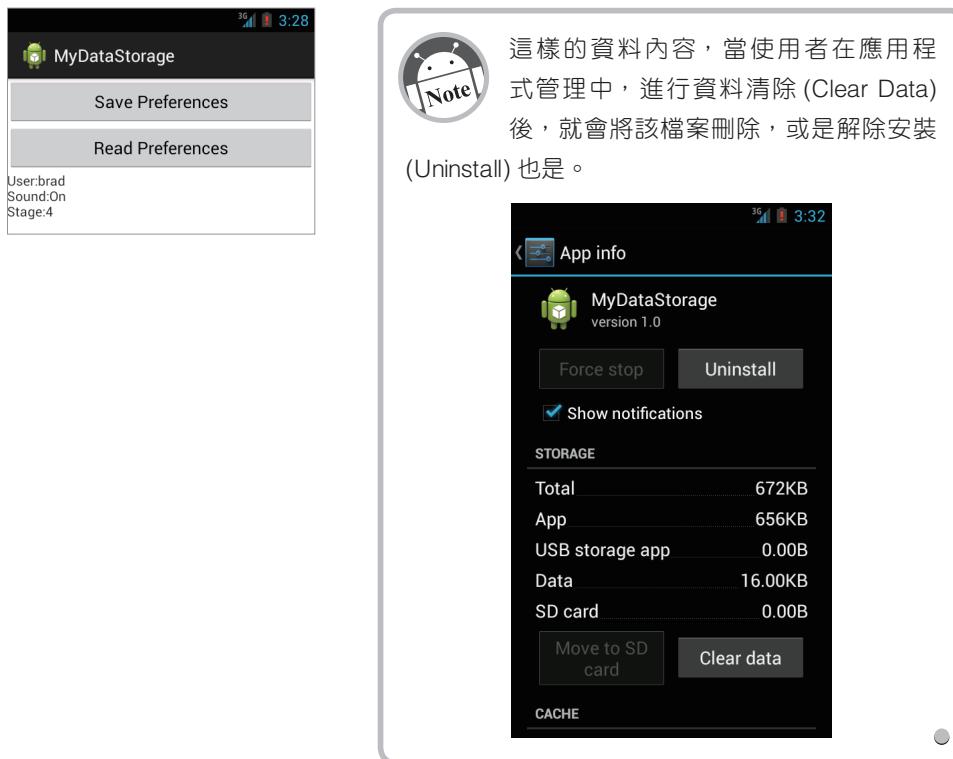
當尚未進行儲存設定資料，而先執行讀取偏好設定資料，則將會以預設資料進行給值，也就是所謂的應用程式初始預設值。呈現畫面如下圖：



而當進行資料設定儲存之後，將可以在 File Explorer 的視窗中觀察，資料檔案被放在 /data/data/<Package-Name>/shared_prefs/ 目錄下的 games.xml，也就是說，程式中只需要指定資料檔名，將會被自動附加副檔名為 .xml，其內容將會如下：

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<int name="stage" value="4" />
<string name="user">brad</string>
<boolean name="sound" value="true" />
</map>
```

再度讀取資料，將會正確讀到如下圖：



8

完整範例

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >

    <Button
        android:id="@+id/save1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Save Preferences"
        />
    <Button
        android:id="@+id/read1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Read Preferences"
        />

    <TextView
        android:id="@+id/info"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />

</LinearLayout>
```

MainActivity.java

```
package tw.brad.android.book.MyDataStorage;

import android.app.Activity;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends Activity {
    private View save1, read1;
```

```
private TextView info;
private SharedPreferences sp;
private SharedPreferences.Editor editor;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    sp = getSharedPreferences("games", MODE_PRIVATE);
    editor = sp.edit();

    info = (TextView) findViewById(R.id.info);
    save1 = findViewById(R.id.save1);
    save1.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            savePreferences();
        }
    });
    read1 = findViewById(R.id.read1);
    read1.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            readPreferences();
        }
    });
}

private void savePreferences() {
    // 清除資料
    // editor.clear();

    editor.putBoolean("sound", true);
    editor.putInt("stage", 4);
    editor.putString("user", "brad");
    editor.commit();

    Toast.makeText(this, "Save OK", Toast.LENGTH_SHORT).show();
}

private void readPreferences() {
    String user = sp.getString("user", "nobody");
    boolean sound = sp.getBoolean("sound", false);
    int stage = sp.getInt("stage", 0);

    info.setText("User:" + user + "\n" + "Sound:" + (sound ? "On" : "Off"))
}
```

```
        + "\n" + "Stage:" + stage);  
    }  
  
}
```



8-2 內部檔案存取機制

使用觀念

在應用程式執行中，當需要針對使用者相關的檔案進行存取，而這種類型的檔案的特性，通常是僅針對應用程式使用而已，例如程式中的物件序列化檔案，當下次執行該應用程式的時候，或是使用者手動暫停後繼續執行，可以讀取該物件解序列化檔案繼續之前的執行過程等等。主要的觀念在於該類型的檔案是配合應用程式才有作用，當使用者解除安裝之後，這類型的檔案就沒有存在的意義，那就非常適用使用內部檔案存取機制。千萬別與應用程式的照相功能或是錄音功能所產生的檔案存取機制搞混，因為相片檔案或是其他類似型態檔案，使用者利用應用程式產生之後，還是可以透過其他應用程式進行存取，那種檔案存取機制就是屬於外部檔案存取，而檔案存放在SDCard。本小節的檔案與上一節的偏好設定一樣，是存放在內存空間。

寫出基本程序

建立及寫出內部檔案到裝置的儲存空間之基本程序：

- 呼叫 Activity 的 openFileOutput() 方法，傳回一個 java.io.FileOutputStream 物件實體。
- 以 java.io.FileOutputStream 物件實體的 write() 方法進行資料寫入裝置的儲存空間。
- 寫入完成之後，呼叫 java.io.FileOutputStream 物件實體的 close() 方法以關閉檔案輸出串流。

建立一個檔案輸出串流

```
private void saveInnerFile(){
    FileOutputStream fout;
    try {
        fout = openFileOutput("MyData.txt", MODE_PRIVATE);

        fout.flush(); // 清除記憶體緩衝
        fout.close(); // 關閉串流
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

呼叫 `openFileOutput()` 方法，傳遞兩個參數：

1. 檔案名稱
2. 開啟模式
 - `MODE_PRIVATE`：每次寫出會刪除原來內容。
 - `MODE_APPEND`：每次寫出會保留原內容，而從檔案尾端開始寫出。

8

以上方法雖然沒有寫入任何資料，一旦執行之後，就已經會產生指定的檔案 `/data/data/<Package-Name>/files/MyData.txt`，檔案大小為 0。

接著進行文字資料寫出，呼叫 `FileOutputStream` 物件實體的 `write()` 方法，將欲寫出的資料轉成 `byte` 陣列型態，傳遞參數寫出即可。

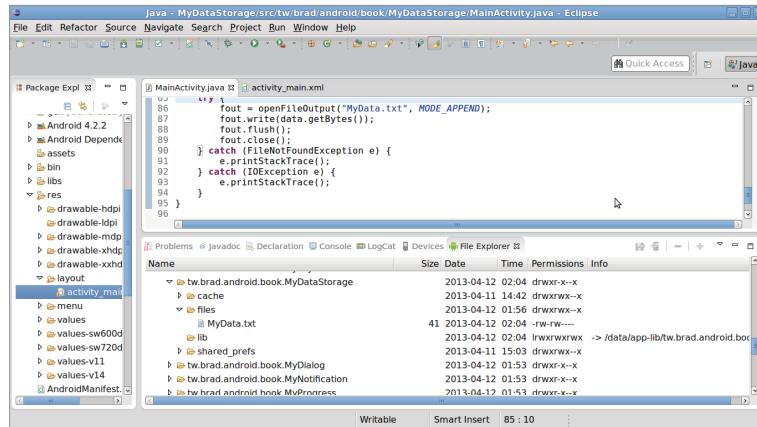
```
private void saveInnerFile() {
    FileOutputStream fout;
    String data = "Hello, World!\n我是趙令文\n";
    try {
        fout = openFileOutput("MyData.txt", MODE_PRIVATE);
        fout.write(data.getBytes());
        fout.flush();
        fout.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
```

```

        e.printStackTrace();
    }
}

```

可以從 File Explorer 觀察到檔案的變化，如下圖：



讀入基本程序

從裝置儲存空間讀取內部檔案之基本程序：

1. 呼叫 Activity 的 openFileInput() 方法，傳回一個 java.io.FileInputStream 物件實體。
2. 以 java.io.FileInputStream 物件實體的 read() 方法從裝置的儲存空間讀入一個位元組的資料。
3. 寫入完成之後，呼叫 java.io.FileInputStream 物件實體的 close() 方法以關閉檔案輸入串流。

建立一個輸入串流：

```

private void readInnerFile() {
    FileInputStream fin;

    try {
        fin = openFileInput("MyData.txt");
    }
}

```

```
        fin.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

呼叫 `openFileInput()`，傳入指定檔案名稱字串即可。檔案名稱直接就是相對 `/data/data/<Package-Name>/files` 資料夾進行指定。

以下範例將檔案內容顯示在 `TextView` 中：

```
private void readInnerFile() {
    FileInputStream fin;

    try {
        fin = openFileInput("MyData.txt");

        BufferedReader reader =
            new BufferedReader(new InputStreamReader(fin));
        String line;
        info.setText("");
        while ((line = reader.readLine()) != null){
            info.append(line + "\n");
        }

        reader.close();

        fin.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

8

因為檔案內容為一般文字檔案，所以先以 `InputStreamReader` 物件將其轉換為 `Reader` 串流物件，再將其串接在 `BufferedReader` 串流物件，以方便呼叫 `readLine()` 方法，一次讀入一列文字資料。

執行後顯示於行動裝置如下圖：



8-3 外部檔案存取

外部檔案存取就是針對 SDCard 的檔案系統進行存取。

SDCard 檔案系統基本觀念

- 與 Linux 檔案系統相同的單根檔案系統
- 檔案目錄名稱大小寫嚴格區分，B 與 b 是不同的
- SDCard 通常是掛載在 /mnt/sdcard 目錄下，最好以 API 來進行判斷
- 資料檔案寫出必須開啟特定寫出權限

判斷使用者的行動裝置是否有 SDCard

呼叫 android.os.Environment 的 static 方法 isExternalStorageRemovable() 即可判斷是否有掛載 SDCard。傳回 true 表示沒有掛載；傳回 false 則表示已經掛載。

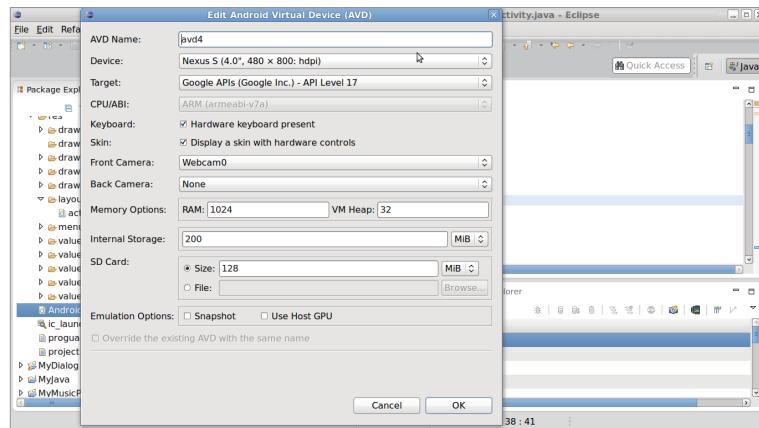
```
if (!Environment.isExternalStorageRemovable()){
    info.setText("SDCard Mounted");
} else {
    info.setText("SDCard Removed");
}
```



上述方式至少要在 API Level 9+，若是 API Level 8 以前的行動裝置，則可以呼叫 `Environment.getExternalStorageState()` 傳回字串判斷：

- "removed"：已經移除 SDCard
- "mounted"：已經掛載 SDCard

如果以模擬器進行測試，記得將模擬器的 SDCard 設定適當的大小，通常不需要設定太大，筆者大約設定成 128MB 就相當足夠了。



8

判斷 SDCard 的掛載點 (Mount Point)

呼叫 `Environment.getExternalStorageDirectory()` 傳回 SDCard 所在的目錄之 `File` 物件實體，可以在呼叫 `File` 物件實體的 `getAbsolutePath()` 方法傳回字串內容。

```
info.setText("SDCard Mounted:" +
    Environment.getExternalStorageDirectory().getAbsolutePath());
```

其他特定目錄都是針對 SDCard 所在目錄的子目錄：

- `Environment.DIRECTORY_ALARMS`
- `Environment.DIRECTORY_DCIM`
- `Environment.DIRECTORY_DOWNLOADS`

- Environment.DIRECTORY_MOVIES
- Environment.DIRECTORY_MUSIC
- Environment.DIRECTORY_NOTIFICATIONS
- Environment.DIRECTORY_PICTURES
- Environment.DIRECTORY_PODCASTS
- Environment.DIRECTORY_RINGTONES

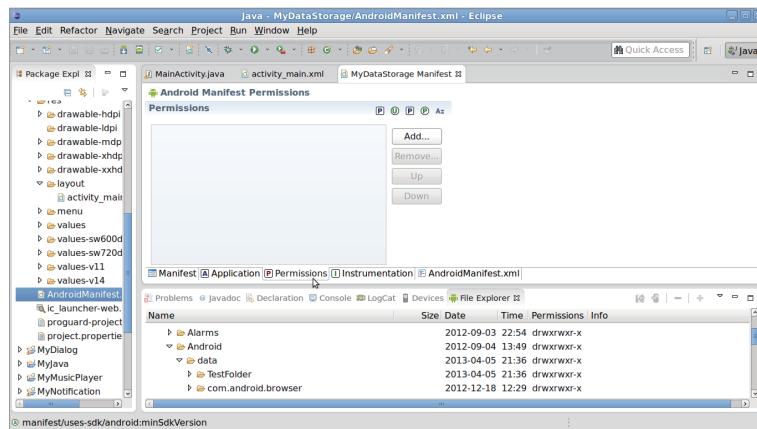
■ 應用程式檔案應該在哪裡

應用程式檔案存取原則應該是放在適當地專屬子目錄底下，如果開發者認定檔案內容特性，是跟著應用程式一起存亡的話，則應該放在`:/mnt/sdcard/Android/data/<Package-Name>/`的子目錄下。如此一來，當使用者解除安裝應用程式之後，該子目錄下的所有內容也將一併刪除，不留痕跡。而放在其他子目錄下的檔案則不受解除安裝應用程式的影響。

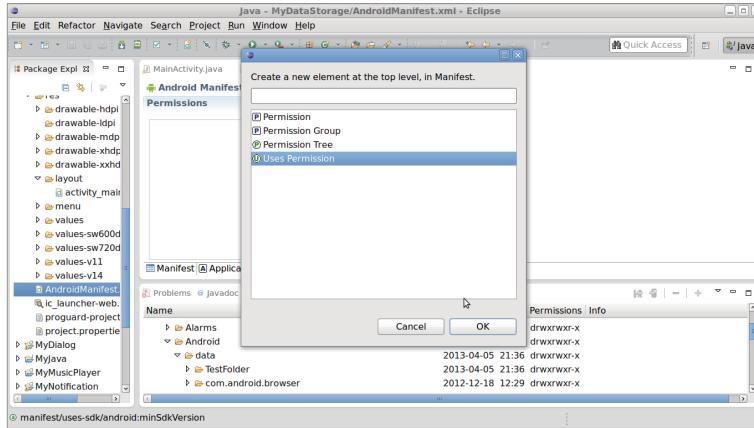
■ 開啟寫出資料的權限

如果沒有開啟寫出權限，即使程式沒有任何邏輯錯誤，也將會在使用者的執行階段拋出例外異常而中止執行。以下就進行使用者權限開啟：

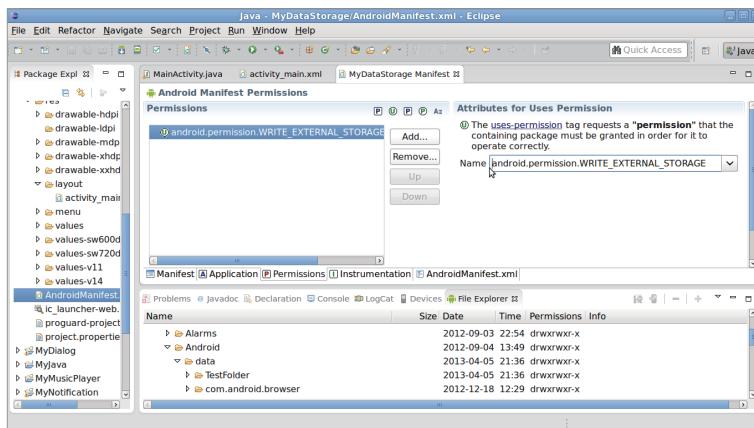
開啟專案目錄下的 `AndroidManifest.xml` 檔案，點選 `Permissions` 的頁籤。



按下「Add」之後，如下圖點選「User Permission」

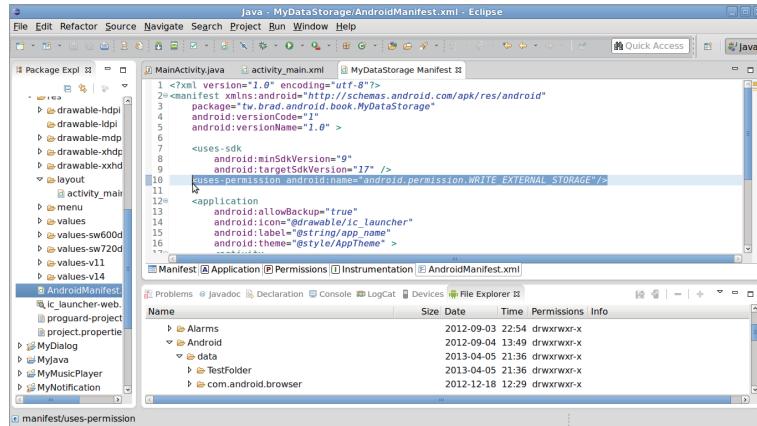


在下圖中右側以下拉式選單點選出「`android.permission.WRITE_EXTERNAL_STORAGE`」並按下「`Ctrl+s`」儲存即可。



或是直接在 `AndroidManifest.xml` 頁籤中直接輸入：

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```



開始進行程式開發

宣告 SDCard 與應用程式的子目錄 File 物件變數

```
private File sdroot, approot;
```

取得 SDCard 的 File 物件實體，再由其建構應用程式的專屬子目錄的 File 物件實體。

```
sdroot = Environment.getExternalStorageDirectory();
approot = new File(sdroot, "Android/data/" + getPackageName());
if (!approot.exists()){
    approot.mkdirs();
}
```

因為一開始使用者首次執行時，可能並不存在應用程式的專屬子目錄，因此先以 exists() 判斷是否存在，若否，則馬上呼叫 mkdirs() 進行建立。如果呼叫 mkdir()，則可能會因為不存在父目錄而無法建立出來。

接下來的處理模式就與一般 Java 檔案存取完全相同。

寫出資料檔案

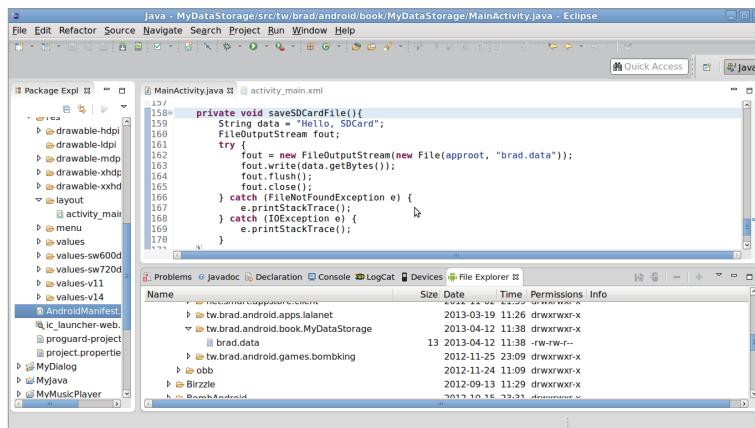
```
private void saveSDCardFile(){
    String data = "Hello, SDCard";
```

```

FileOutputStream fout;
try {
    fout = new FileOutputStream(new File(appproot, "brad.data"));
    fout.write(data.getBytes());
    fout.flush();
    fout.close();
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
}

```

則將會寫出資料於 /mnt/sdcard/Android/data/<Package-Name>/brad.data
如下圖：



8

■ 讀入資料檔案 ■

```

private void readSDCardFile(){
    FileInputStream fin;
    try {
        info.setText("");
        fin = new FileInputStream(new File(appproot, "brad.data"));
        BufferedReader reader =
            new BufferedReader(
                new InputStreamReader(fin));
        String line;
    }
}

```

```
        info.setText("");
        while ((line = reader.readLine()) != null) {
            info.append(line + "\n");
        }

        reader.close();
        fin.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```



8-4 行動裝置資料庫處理機制 SQLite

開發應用程式運用的資料庫系統來處理大量的資料，使用 SQL 查詢語法可以迅速的過濾出想要的資料，這應該是資料庫系統有別於一般檔案輸入輸出的優勢。一樣是存放資料，但是放在資料庫中的資料比較具有使用上的意義，而以一般檔案存放資料的缺點就是查詢過濾的複雜性與效能性遠低於資料庫。Android 完全支援 SQLite 資料庫系統。

建立資料庫的輔助類別物件

自行開發一個子類別繼承 SQLiteOpenHelper 類別，用來建立資料庫及資料表。實做方式如下：

- Override: onCreate(SQLiteDatabase db)
- Override: onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
- 定義建構式：MyDBHelper(Context context, String dbname, CursorFactory factory,int version)

預先處理模式

以實際範例來說明處理模式，假設 App 中會使用到資料庫來存放客戶相關資料，因此會有一個資料表存在於資料庫中，該資料表的結構大致如下：

- `_id`：整數型態；主鍵；自動遞增
- `name`：文字型態
- `tel`：文字型態
- `birthday`：日期型態

依據 SQLite 的語法來建立該資料表 (`cust`)：

```
CREATE TABLE cust (_id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, tel
TEXT, birthday DATE)
```

因此開發出一個自訂 `SQLiteOpenHelper` 類別如下 (`MyDBHelper`)：

```
package tw.brad.android.book.MyDataStorage;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.CursorFactory;
import android.database.SQLiteOpenHelper;

public class MyDBHelper extends SQLiteOpenHelper {
    private static final String createCustTable =
        "CREATE TABLE cust (_id INTEGER PRIMARY KEY AUTO_INCREMENT," +
        "name TEXT, tel TEXT, birthday DATE)";

    public MyDBHelper(Context context, String dbname, CursorFactory factory,
                      int version) {
        super(context, dbname, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(createCustTable);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    }
}
```

8

此時就可以開發到主程式進行開發。先進行宣告：

```
private MyDBHelper dbhelper;
private SQLiteDatabase db;
```

接著呼叫 dbhelper 物件方法 getReadableDatabase() 或是 getWritableDatabase() 回傳 SQLiteDatabase 物件存放在 db 變數。兩者的差異在於使用者的儲存空間不足的情況之下，getReadableDatabase() 仍然可以開啟使用，只是處在唯讀的模式，一旦使用者釋放出足夠儲存空間，則又可以開始進行讀寫模式；而 getWritableDatabase() 則在使用者的儲存空間不足的情況之下，直接拋出 Exception，開發者將會針對該 Exception 進行 try...catch 的開發結構來處理。

以下建立出資料庫名稱為 "brad":

```
dbhelper = new MyDBHelper(this, "brad", null, 1);
db = dbhelper.getReadableDatabase();
//db = dbhelper.getWritableDatabase();
```

而資料庫的關閉時機，以下放在該 Activity 結束時：

```
@Override
public void finish() {
    if (db.isOpen()){
        db.close();
    }
    super.finish();
}
```

|| 簡單查詢資料 ||

先來開發撰寫最簡單也最常用的查詢語法：SELECT * FROM cust

```
private void querySQLiteData() {
    Cursor c = db.query("cust", null, null, null, null, null, null);
    info.setText("筆數：" + c.getCount() + "\n");
    while (c.moveToNext()){
        info.append(c.getString(c.getColumnIndex("_id")) + ":" +
                    c.getString(c.getColumnIndex("name")) + ":" +
                    c.getString(c.getColumnIndex("tel")) + ":" +
                    c.getString(c.getColumnIndex("birthday")) + "\n");
    }
}
```

- 呼叫 SQLiteDatabase 物件 db 的 query()，傳遞第一個參數為資料表名稱，其餘先暫時設定為 null，共有六個 null。
- query() 方法傳回 Cursor 物件實體。
- 呼叫 Cursor 物件實體的 moveToNext()，使查詢指標往下一筆資料移動，當沒有任何資料時，將會傳回 false(boolean)。
- 透過 Cursor 物件實體的 getCount() 方法傳回查詢結果的資料筆數。
- 透過 Cursor 物件實體的 getColumnIndex() 指定查詢欄位名稱，傳回其傳回結果欄位的 index。
- 再呼叫 Cursor 物件實體的 getString()，指定欄位 index，傳回該筆資料內容。

新增資料

直接呼叫 SQLiteDatabase 物件實體的 insert() 方法，傳遞三個參數：

- 資料表名稱
- null: 通常設定為 null
- 資料內容

8

```
private void insertSQLiteData(){  
    ContentValues values = new ContentValues();  
    values.put("name", "test" + (int)(Math.random()*100));  
    values.put("tel", "0999-123456");  
    values.put("birthday", "1999-12-12");  
    db.insert("cust", null, values);  
}
```

資料內容以 ContentValues 物件來實作，呼叫其 put(Key, Value)，將欄位字串當作其 Key，而其資料值放在第二的參數傳遞。上例中建立了 name 欄位為 "test" + 0-99 的數字資料。

刪除資料

直接呼叫 SQLiteDatabase 物件實體的 delete() 方法，傳遞三個參數：

- 資料表名稱
- 查詢條件式
- 查詢條件值字串陣列

以下列的 SQL 語法為例：

```
DELETE FROM cust WHERE name like 'test%'
```

則寫成如下：

```
private void deleteSQLiteData() {
    // db.delete("cust", "name like ?", new String[]{"test_"});
    db.delete("cust", "name like ?", new String[] { "test%" });
}
```

註解列用來刪除 test0 – test9 而已。查詢條件式中的問號可以多個，而多個問號依照由左至右的順序，一對一的對應到查詢條件值字串陣列的值。

修改資料

直接呼叫 SQLiteDatabase 物件實體的 update() 方法，傳遞四個參數：

- 資料表名稱
- 修改的資料內容
- 查詢條件式
- 查詢條件值字串陣列

以下列的 SQL 語法為例：

```
UPDATE cust SET name='趙令文',birthday='1999-01-01' WHERE name='Brad'
```

實作如下：

```
private void updateSQLiteData() {
    ContentValues values = new ContentValues();
    values.put("name", "趙令文");
    values.put("birthday", "1999-01-01");
    db.update("cust", values, "name = ?", new String[] { "Brad" });
}
```

進一步了解查詢方式

再來回頭看看查詢的參數項：

1. 資料表名稱
2. 查詢欄位字串陣列
3. 查詢條件式
4. 查詢條件值字串陣列
5. GROUP BY 字串語法
6. Having 字串語法
7. ORDER By 字串語法

以下列 SQL 為例：

```
SELECT _id, name, birthday WHERE _id>4 ORDER BY _id
```

實作如下：

```
Cursor c = db.query("cust",
    new String[]{"_id", "name", "birthday"}
    , "_id > ?",
    new String[]{"4"},
    null,
    null,
    "_id");
```

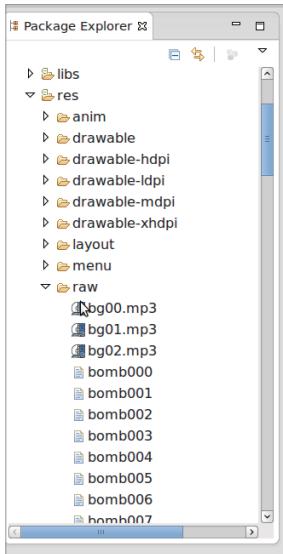
8



8-5 應用 App 資源中的資料存取資料：遊戲關卡資料處理為例

應用程式中事先設定好的資料內容，提供應用程式在執行中存取使用，給使用者在不同的情境執行，可以使用本小節的處理模式。例如遊戲 App 中的關卡地圖資料，或是應用程式中給使用者不同的操作環境相關資料。

資料存放在專案架構下的 res/ 目錄，自行建立出 res/raw/ 的子目錄，將資料相關檔案放置於該子目錄下，如下圖所示：



提醒：子目錄名稱為全小寫 raw，不得任意修改，修改原則式依照資源名稱規則。

而在程式中將會事先定義一個 int[] 陣列存放所需要用到的部份：

關卡資料：

```
private int[] leveldata = {
    R.raw.bomb000, R.raw.bomb001, R.raw.bomb002, R.raw.bomb003,
    R.raw.bomb004, R.raw.bomb005, R.raw.bomb006, R.raw.bomb007};
```

定義資料

而資料內容為自行定義，本例中定義如下：

- 一列就是一個關卡地圖中的顯示內容一列
- 0: 不顯示
- 1: 顯示磚塊
- 2: 顯示岩石
- 7: 顯示敵人
- 8: 顯示主角
- 9: 顯示目標物

預計顯示如下遊戲畫面：



右大半邊就是整個關卡的地圖，目前主角在左下方，敵人在右下方，有三個目標物要去取得，而散布在磚塊中的寶物是在程式中以亂數產生，所以不在關卡中設定，以免玩家玩出密技，增加遊戲變化的趣味性。

■ 讀取資料檔案 ■

8

產生一般簡單的文字檔內容如下：

bomb001

```
0,0,0,0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2  
0,0,0,0,2,9,0,0,0,1,0,0,0,0,0,1,1,0,2  
0,0,0,0,2,1,1,1,0,1,0,1,1,1,1,0,0,1,9,2  
0,0,0,0,2,0,0,0,0,1,0,1,0,0,0,0,0,1,1,2  
0,0,0,0,2,0,1,1,1,1,0,1,0,1,0,0,0,0,0,2  
0,0,0,0,2,0,0,0,0,0,1,0,1,0,1,1,1,1,2  
0,0,0,0,2,0,1,0,0,0,0,0,1,0,0,0,0,0,1,2  
0,0,0,0,2,0,1,0,1,1,1,0,1,0,1,1,0,1,2  
0,0,0,0,2,0,1,0,0,0,0,0,1,0,0,0,0,0,1,2  
0,0,0,0,2,0,1,1,1,1,0,1,0,1,1,1,0,1,2  
0,0,0,0,2,0,1,1,1,1,0,1,1,1,1,1,9,0,2  
0,0,0,0,2,8,0,0,0,0,0,1,0,0,0,1,1,0,7,2  
0,0,0,0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
```

程式中讀取方式

透過呼叫 Context 的 getResources() 方法傳回 Resources 物件實體，也就是用來存取資源資料的物件。再呼叫其 openRawResource()，傳遞資源變數整數值，則將傳回一個 InputStream 物件實體。

```
BufferedReader reader = new BufferedReader(  
    new InputStreamReader(res.openRawResource(leveldata[n])));
```

之後進行資料解析，寫入一個事先定義的二維陣列之中 gamemap[][]

```
String temp1;  
String[] temp3;  
int i = 0, j = 0, v;  
while ((temp1 = reader.readLine()) != null) {  
    temp3 = temp1.split(",");  
    for (String temp4 : temp3) {  
        v = Integer.parseInt(temp4);  
        gamemap[i][j] = v;  
    }  
}
```

當然，以上僅用來說明讀取專案資源的檔案，解析資料方式非常多，讀者可以考慮以 XML 方式作為資料格式。

詳細的原始碼如本書所附的光碟中。

09

Chapter

網際網路相關

- 9-1 網路介面及 IP Address
- 9-2 UDP 通訊協定的資料存取
- 9-3 TCP 通訊協定的資料存取
- 9-4 Http 通訊協定的資料存取
- 9-5 WebView 使用





9-1 網路介面及 IP Address

行動裝置中網路裝置的相關狀態取得，是有助於存取網路相關服務的重要資訊。使用者的行動裝置的網路連線狀態，使用的網路服務型態，以及目前所使用的 IP Address 等等。都會是在往後的網路服務中使用。

因此，本小節先來進行行動裝置上的網路相關資訊的取得。

裝置的網路狀態

android.net.NetworkInfo 類別物件可以用來取得目前網路的連線狀態，以及提供服務的型態。而該物件的取得是透過 android.net.ConnectivityManager 物件實體呼叫其 `getActiveNetworkInfo()` 方法傳回。ConnectivityManager 是由 `getSystemService()` 方法傳入 `Context.CONNECTIVITY_SERVICE` 得到。

```
ConnectivityManager connMgr =  
    (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);  
  
NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
```

接著就可以由 `networkInfo` 物件實體呼叫 `isConnected()` 方法傳回 boolean 值來判斷目前是否已經連線；呼叫其 `gettypeName()` 方法傳回連線服務為 WIFI 或是 MOBILE。

```
if (networkInfo != null && networkInfo.isConnected()) {  
    sb.append("Type: " + networkInfo.getTypeName() + "\n");  
} else {  
    sb.append("Not Connect");  
}
```

以上部份的使用呼叫，必須要開啟 `android.permission.ACCESS_NETWORK_STATE` 的使用權限。在 `androidManifest.xml` 中加上：

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

■ 網路介面的 IP Address ■

如果要繼續以下的網路介面卡資訊的取得，則必須再開啟 android.permission.INTERNET 的使用權限。

```
<uses-permission android:name="android.permission.INTERNET"/>
```

呼叫 `java.net.NetworkInterface` 的 `getNetworkInterfaces()` 方法 將 傳 回 `Enumeration<NetworkInterface>` 的物件實體，代表行動裝置上目前所偵測到的網路介面。而每個網路介面上都可被綁定一組以上的 IP Address，所以可以用以下方式進行查找：

```
try {
    Enumeration<NetworkInterface> ifcs = NetworkInterface
        .getNetworkInterfaces();
    while (ifcs.hasMoreElements()) {
        NetworkInterface ifc = ifcs.nextElement();
        sb.append("Interface: " + ifc.getDisplayName() + "\n");
        List<InterfaceAddress> ips = ifc.getInterfaceAddresses();
        for (InterfaceAddress ip : ips) {
            sb.append("\tIP: " + ip.getAddress().getHostAddress()
                + "\n");
        }
    }
} catch (SocketException e) {
    sb.append("XXX");
}
```

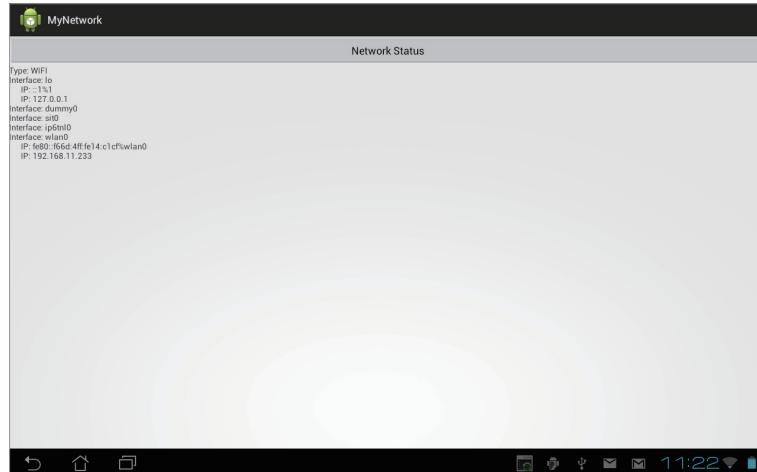
9

實際測試後的結果如下：

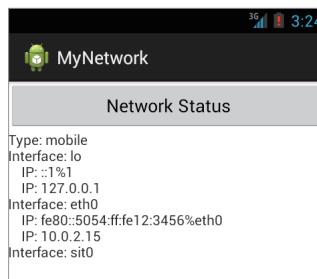
Android 2.3 手機裝置



平板電腦 Android 4+



模擬器 Android 4+



在模擬器中可以發現到其取得的 IP Address 為 10.0.2.15. 另外，在 Android 4+ 以後提供支援 IPv6.

模擬器中的相關網路位置如下表：

網路位址	說明
10.0.2.1	閘道位址
10.0.2.2	原本作業系統的 127.0.0.1 的別名位址
10.0.2.3	第一優先 DNS
10.0.2.4/10.0.2.5/10.0.2.6	第二三四順位 DNS
10.0.2.15	AVD 本機網路位址
127.0.0.1	AVD 本機迴路位址

其中 10.0.2.2 是對應到實際執行 AVD 的作業系統的 127.0.0.1，所以當以 AVD 的觀點與 10.0.2.2 的主機進行網路連線，就是相當於對作業系統的 127.0.0.1 進行網路連線。

取得裝置連線 IP Address

只要判斷網路介面中，非 loopback 的網路介面即為目前對外部連線的 IP Address：

```
private void getMyIPAddress() {  
    try {  
        for (Enumeration<NetworkInterface> en = NetworkInterface  
             .getNetworkInterfaces(); en.hasMoreElements();) {  
            NetworkInterface intf = en.nextElement();  
            for (Enumeration<InetAddress> enumIpAddr = intf  
                 .getInetAddresses(); enumIpAddr.  
                 hasMoreElements();) {  
                InetAddress inetAddress = enumIpAddr.nextElement();  
                if (!inetAddress.isLoopbackAddress()) {  
                    mesg.setText(inetAddress.getHostAddress());  
                }  
            }  
        }  
    } catch (SocketException ex) {}  
}
```

9

建構 IP Address 物件實體

在後面相關網際網路單元中，都會應用到 IP Address 物件實體，也就是上例中的 java.net.InetAddress 類別物件實體。該物件實體並非以 new 方式建構出來，通常會透過 InetAddress 類別的 static 方法，呼叫 getXxx() 系列方法取得。例如：

```
private void testIP(){  
    try {  
        InetAddress ip1 = InetAddress.getByName("www.brad.tw");  
        InetAddress ip2 = InetAddress.getByName("192.168.12.34");  
    }
```

```
        } catch (UnknownHostException e) {
            Log.i("brad", "不明主機");
        }
    }
```

呼叫 `getByName()` 方法，傳入字串參數，可以是主機名稱或是 IP Address。傳入主機名稱之後，會依照使用者行動裝置的 DNS 進行主機名稱解析，如果無法解析，則會拋出 `UnknownHostException`；如果是輸入 IP Address 字串，只要輸入格式不正確，也將會拋出 `UnknownHostException` 的例外異常，但是並非表示該 IP Address 的主機於當時是存在的。



9-2 UDP 通訊協定的資料存取

UDP 的連線方式是屬於非連接導向，當 Client 發送出 UDP 的資料封包給遠端 Server 時，如果 Server 已經準備好接收該 UDP 封包資料，當然就馬上進行接收；但是當 Server 端並未處在接收狀態下，該資料封包就會被自動丟棄，連 Client 端也不會有任何通知重送的相關訊息。這樣的連線模式重點在於資料傳送的速度較快，相對的是並不重視資料的完整性。

處理模式

以 `java.net DatagramSocket` 與 `java.net DatagramPacket` 兩個類別物件為主。

建立接收端：

- 建構 `DatagramSocket` 物件實體，並綁定在特定的通訊埠
- 建構 `DatagramPacket` 物件實體，並以特定長度的 byte 陣列來接收資料
- `DatagramSocket` 物件實體呼叫 `receive()` 方法，並傳入 `DatagramPacket` 物件實體為參數
- 接收資料後，相關封包資料內容存放在 `DatagramPacket` 物件實體中

而接收完成的 DatagramPacket 物件實體相關資料如下：

- getAddress()，傳回傳送端的 InetAddress 物件，可以取得其 IP Address
- getLength()，傳回傳送資料的長度
- getData()，傳回資料內容 byte 陣列

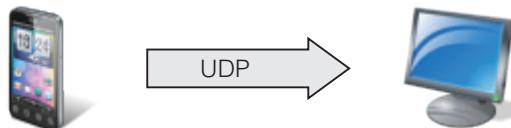
建立傳送端：

- 建構 DatagramSocket 物件實體，無須指定通訊埠
- 建構 DatagramPacket 物件實體，將傳遞的資料內容轉換成 byte 陣列來傳送資料，並指定接收端的 InetAddress 物件，及接收端的通訊埠
- DatagramSocket 物件實體呼叫 send() 方法，並傳入 DatagramPacket 物件實體為參數

實作測試

以下直接以 UDP 通訊協定進行 Android 行動裝置發送資料給一般 PC 電腦。

9



先進行 PC 接收端，開發以下的 Java 程式，以執行緒方式啟動 UDP 等候在特定的通訊埠 8888，如果讀者 PC 端有防火牆相關設定，記得先將 UDP Port:8888 先行進行允許通過。

MyUDPREceiver.java

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;

public class MyUDPREceiver {
    public static void main(String[] args) {
```

```
        new UDPReceiveThread().start();
    }

    private static class UDPReceiveThread extends Thread {
        private DatagramSocket socket = null;
        private DatagramPacket packet = null;
        public UDPReceiveThread() {
            try {
                socket = new DatagramSocket(8888);
            } catch (SocketException e) {
            }
        }
        @Override
        public void run() {
            while (socket != null) {
                byte[] buf = new byte[1024];
                packet = new DatagramPacket(buf, buf.length);
                try {
                    socket.receive(packet);
                    String data = new String(packet.getData(),
0, packet.getLength(), "UTF-8");
                    System.out.println(data);
                    if (data.equals("quit")){
                        break;
                    }
                } catch (IOException e) {
                    break;
                }
            }
            if (socket != null) {
                socket.close();
                socket = null;
            }
        }
    }
}
```

重點說明：

- while 判斷 DatagramSocket 是否不為 null，就以一個 1024bytes 的陣列為接收陣列
- socket 物件實體呼叫 receiver() 方法傳入 packet 物件實體開始等候接收
- 收到之後呼叫 packet 物件實體相關方法，取出所需要的資料
- 如果資料內容為 "quit" 字串，則結束等候的執行緒

而在 Android 行動裝置上開發一個測試專案。規劃畫面如下：



版面規劃：res/layout/activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <Button
        android:id="@+id/up"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Up" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <Button
            android:id="@+id/left"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Left" />

        <Button
            android:id="@+id/stop"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Stop" />

    </LinearLayout>
</LinearLayout>
```

```
<Button  
    android:id="@+id/right"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_weight="1"  
    android:text="Right" />  
  
</LinearLayout>  
  
<Button  
    android:id="@+id/down"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="Down" />  
  
</LinearLayout>
```

回到程式開發：

```
package tw.brad.android.book.myudpsender;  
  
import java.io.IOException;  
import java.net.DatagramPacket;  
import java.net DatagramSocket;  
import java.net.InetAddress;  
import java.net.SocketException;  
  
import android.app.Activity;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.View;  
import android.view.View.OnClickListener;  
  
public class MainActivity extends Activity {  
    private View up, down, left, right, stop;  
    private DatagramSocket socket;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        try {  
            socket = new DatagramSocket();  
        } catch (SocketException e) {  
        }
```

```
}

up = findViewById(R.id.up);
down = findViewById(R.id.down);
left = findViewById(R.id.left);
right = findViewById(R.id.right);
stop = findViewById(R.id.stop);

up.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        new MyUDPSend("往上").start();
    }
});
down.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        new MyUDPSend("往下").start();
    }
});
left.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        new MyUDPSend("往左").start();
    }
});
right.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        new MyUDPSend("往右").start();
    }
});
stop.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        new MyUDPSend("quit").start();
    }
});

@Override
public void finish() {
    if (socket != null) socket.close();
    super.finish();
}
```

9

```
private class MyUDPSend extends Thread {  
    private byte data[];  
    private DatagramPacket packet;  
  
    MyUDPSend(String dd){  
        data = dd.getBytes();  
    }  
    @Override  
    public void run() {  
        if (socket != null){  
            try {  
                packet = new DatagramPacket(data, data.length,  
                    InetAddress.getByName("192.168.2.104"),  
                    8888);  
                socket.send(packet);  
            } catch (IOException e) {  
            }  
        }  
    }  
}
```

重點說明：

- 開啟使用權限：INTERNET
- PC 接收端假設其 IP Address 為 192.168.2.104
- 以上此段測試實做，也可以延伸成為行動裝置控制器



9-3 TCP 通訊協定的資料存取

TCP 通訊協定是屬於連接導向，Client 端與 Server 端在進行資料傳遞之前，必須經過三方交握確認連線狀態之後，才開始進行資料的傳遞。因此，可以確保資料傳遞的完整性。TCP 就好像撥電話一樣，一定會確認接通的對方是要找的人之後，才會開始進行訊息的傳遞。

處理模式

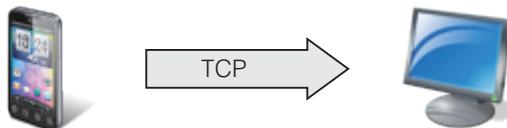
伺服端（Server）先建構出 `java.net.ServerSocket` 物件實體即可，傳遞參數指定其監聽在特定的通訊埠後，呼叫其 `accept()` 方法就開始進行監聽（Listening）。一旦用戶端（Client）開始進行連線，則將傳回一個 `java.net.Socket` 物件實體。可以透過 `Socket` 物件實體呼叫 `getInputStream()` 傳回 `java.io.InputStream` 物件實體，而開始進行資料的傳遞。

用戶端（Client）只需要建構出 `java.net.Socket` 物件實體，指定傳送封包資料的目的端及通訊埠，一旦完成連線，則呼叫 `Socket` 物件實體之 `getOutputStream()` 方法傳回 `OutputStream` 物件實體，開始進行資料傳送。

實做測試

以下直接以 TCP 通訊協定進行 Android 行動裝置發送資料給一般 PC 電腦。

9



先進行 PC 伺服端，開發以下的 Java 程式，以執行緒方式啟動 TCP 監聽在特定的通訊埠 9999，如果讀者 PC 端有防火牆相關設定，記得先將 TCP Port:9999 先行進行允許通過。

MyTCPServer.java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.DatagramPacket;
import java.net.ServerSocket;
import java.net.Socket;
```

```
public class MyTCPServer {  
    public static void main(String[] args) {  
        new TCPServerThread().start();  
    }  
  
    private static class TCPServerThread extends Thread {  
        private ServerSocket server = null;  
        private Socket socket = null;  
        private InputStream is = null;  
        private BufferedReader reader = null;  
  
        public TCPServerThread() {  
            try {  
                server = new ServerSocket(9999);  
            } catch (IOException e) {  
            }  
        }  
        @Override  
        public void run() {  
            stop:  
            while (server != null){  
                try {  
                    socket = server.accept();  
                    is = socket.getInputStream();  
                    reader = new BufferedReader(new  
                        InputStreamReader(is));  
                    String line;  
                    while ( (line = reader.readLine()) != null){  
                        System.out.println(line);  
                        if (line.equals("quit")) break stop;  
                    }  
                    reader.close();  
  
                } catch (IOException e) {  
                }  
            }  
            if (server != null) {  
                try {  
                    server.close();  
                } catch (IOException e) {  
                }  
            }  
        }  
    }  
}
```

而用戶端的版面規劃與 UDP 的實作測試一樣，僅列出程式開發實作：

```
package tw.brad.android.book.mytcpclient;

import java.io.IOException;
import java.io.OutputStream;
import java.net.InetAddress;
import java.net.Socket;
import java.net.UnknownHostException;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;

public class MainActivity extends Activity {
    private View up, down, left, right, stop;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        up = findViewById(R.id.up);
        down = findViewById(R.id.down);
        left = findViewById(R.id.left);
        right = findViewById(R.id.right);
        stop = findViewById(R.id.stop);

        up.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                new MyTCPClient("往上").start();
            }
        });
        down.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                new MyTCPClient("往下").start();
            }
        });
        left.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                new MyTCPClient("往左").start();
            }
        });
    }
}
```

```
        }
    });
    right.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            new MyTCPClient("往右").start();
        }
    });
    stop.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            new MyTCPClient("quit").start();
        }
    });
}

@Override
public void finish() {
    super.finish();
}

private class MyTCPClient extends Thread {
    private byte data[];
    private Socket socket = null;

    MyTCPClient(String dd) {
        data = dd.getBytes();
    }

    @Override
    public void run() {
        try {
            socket = new Socket(InetAddress.getByName
("192.168.2.104"),
                    9999);
            OutputStream os = socket.getOutputStream();
            os.write(data);
            os.flush();
            socket.close();
        } catch (IOException e) {
            Log.i("brad", "exception");
        }
    }
}
}
```

重點說明：

- 記得開啟使用權限：INTERNET
- 所有網路相關物件，必須放在執行緒中進行，不可以在主執行緒中進行操作。(Android 4+)



9-4 Http 通訊協定的資料存取

其實 Http 通訊協定是架構在 TCP 的通訊協定，只需要將相關協定的資料傳送即可，但是實作上較為複雜，因此在 Android 上可以較為簡便的 API 來進行開發。

常見的兩種 API 的使用：

- 實作 Apache 的 HttpClient 的 android.net.http.AndroidHttpClient 的類別物件，而在官方 API 文件是說 DefaultHttpClient
- 以 java.net.URL 物件實體來開啟出 HttpURLConnection 物件實體

9

而以上兩個物件實體的觀念，就是一般在瀏覽網頁的瀏覽器的角色地位，所以會直接影響到網頁伺服端的 Session 處理，一旦該物件實體呼叫 close() 方法或是重新取得建構出來，相當於重新開啟一個瀏覽器，此時之前用戶端的 Session 就自動取消。

以 AndroidHttpClient 及 DefaultHttpClient 實做

而第一種 API 方式處理，既然與 Apache 的 DefaultHttpClient 是類似行為，筆者就先以 AndroidHttpClient 來實作說明。先來處理 GET method 方式傳遞要求。

```
AndroidHttpClient client =
    AndroidHttpClient.newInstance("輸入傳遞給伺服器 Agent 字串");
```

再來建構出 `HttpGet` 物件實體，其觀念相當於瀏覽器上輸入的網址列：

```
HttpGet get = new HttpGet("http://android.ez2test.com/bombking.png");
```

上例假設想要取得遠端伺服器的 `bombking.png` 的網路資源，事實上是一個 `png` 的影像檔案。

接著就可以要求 `client` 呼叫執行 `get` 物件實體。

```
HttpResponse response = client.execute(get);
```

執行之後會傳回一個 `HttpResponse` 的物件實體，代表遠端伺服器的回應物件。

想要取得其回應內容，可以先呼叫 `getEntity()` 方法傳回 `HttpEntity` 物件實體，再呼叫 `getContent()` 方法就可以傳回 `InputStream` 的輸入串流物件，進而接收資料內容。

```
InputStream is = response.getEntity().getContent();
```

以下實作出一個執行緒來進行這段處理程序如下：

```
private class MyGetHttp extends Thread {
    String url;

    MyGetHttp(String uu) {
        url = uu;
    }

    @Override
    public void run() {
        AndroidHttpClient client = AndroidHttpClient.newInstance
        ("Android");
        HttpGet get = new HttpGet(url);

        try {
            HttpResponse response = client.execute(get);
            InputStream is = response.getEntity().getContent();

            bmp = BitmapFactory.decodeStream(is);
            handler.sendMessage(1);
        }
    }
}
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
}
```

重點說明：

- 宣告一個字串屬性用來存放 URL
- 傳回來的串流適用於影像檔案內容，所以直接呼叫 BitmapFactory.decodeStream() 方法，將資料轉為 Bitmap 物件實體。讀者可以轉換成其他內容來彈性應用。
- 因為 Thread 中無法直接進行 View 元件的存取，所以透過 Handler 類別的自訂子類別來處理後續的影像呈現在 ImageView 中。

```
private class MyHandler extends Handler {
    @Override
    public void handleMessage(Message msg) {
        if (msg.what == 1) {
            img.setImageBitmap(bmp);
        }
    }
}
```

9

上例中，img 為在版面配置中的 ImageView。

再來以 DefaultHttpClient 來處理 POST method 的傳遞模式，操作原理非常類似，只是物件實體取得有些許差異，另一方面，既然使用 POST method，也順便來練習傳遞參數。(GET method 的參數直接以 URL 方式即可)

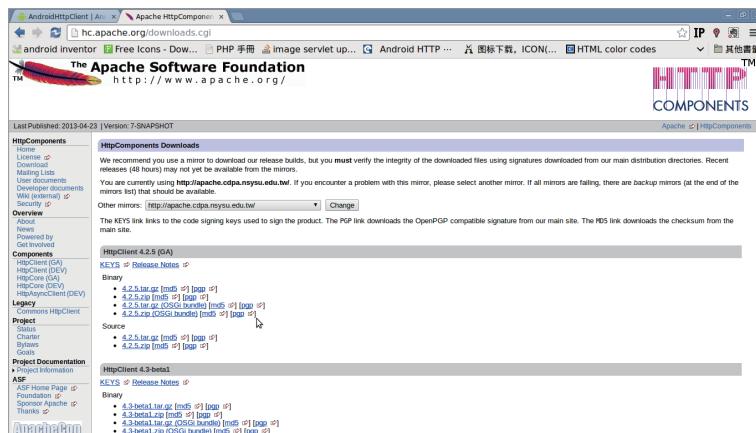
建構 DefaultHttpClient 物件實體：

```
DefaultHttpClient client = new DefaultHttpClient();
```

建構 HttpPost 物件實體：

```
HttpPost post = new HttpPost("http://android.ez2test.com/posttest.php");
```

之後的參數透過 Apache 的 HttpClient 的 API 來進行，因此可以至其官方網站下載：(Google 搜尋關鍵字 apache httpclient 即可)：



將壓縮檔解壓縮至特定目錄後，回到 Eclipse 專案下，如果是下載 OSGi bundle 的版本，就是將解壓縮後的一個 org.apache.httpcomponents.httpclient_ 版本數 .jar 檔案，或是非 OSGi bundle 的版本，將解壓縮後的 libs/ 子目錄下的所有 xxx.jar 檔案複製到專案目錄下的 libs/ 子目錄即可。

開始處理傳遞的參數，假設有一個需要傳遞 username 與 password 兩個參數，而用戶端打算使其值為 brad 與 123456，則先以 org.apache.entity.mime.content.StringBody 類別物件將參數值進行編碼。

```
StringBody sb1 = new StringBody("brad");
StringBody sb2 = new StringBody("123456");
```

再來建構出 org.apache.entity.mime.MultipartEntity 物件實體，用來將傳遞參數建立關係。

```
MultipartEntity entity = new MultipartEntity();
entity.addPart("username", sb1);
entity.addPart("passwd", sb2);
```

終於要以 `HttpPost` 物件實體將其設定為 Entity，就可以請 `DefaultHttpClient` 物件實體來執行 `HttpPost` 了。

```
post.setEntity(entity);
HttpResponse response = client.execute(post);
```

之後的處理動作，從取得 `InputStream` 開始都與上述方法一樣。

以 `java.net.HttpURLConnection` 實作

另一種方式是透過 `HttpURLConnection` 來處理，通常會先建構出 `java.net.URL` 物件實體。

```
URL url = new URL("http://android.ez2test.com/bombking.png");
```

9

再來就是透過 `URL` 物件實體呼叫其 `openConnection()` 方法，傳回 `URLConnection` 物件實體，在將其強制轉型為 `HttpURLConnection`。

```
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
```

若是採用 GET method 的傳輸，則先設定 `setRequestMethod()` 方法。

```
conn.setRequestMethod("GET");
```

或其他相關設定。

```
conn.setReadTimeout(3000);
conn.setConnectTimeout(5000);
```

最後呼叫 `connect()` 方法開始進行連線。

```
conn.connect();
```

沒有 Exception 拋出狀況下，就可以接著呼叫 `getInputStream()` 方法來接收資料。

```
InputStream is = conn.getInputStream();
```

當取得到 `InputStream` 之後，處理的原則和模式就與上述完全相同。



9-5 WebView 使用

`android.webkit.WebView` 是用來呈現網頁內容的 View 元件，其提供了一般瀏覽器常見的瀏覽歷程（上一頁或是下一頁）、放大縮小或是文字搜尋功能等等。網頁內容的提供可能是遠端的網頁伺服器，也可能是附加在專案 App 中的檔案。對於網頁內容的提供者而言，同時能夠符合一般的網頁瀏覽器與行動裝置而言，就不是那麼容易。如果將網頁內容定位在整個 App 的版面編排處理，也可以與 Android 的其他 View 元件及程式開發控制，這樣反而會有不同的開發架構模式。

預設狀況之下，`WebView` 並非類似瀏覽器，沒有提供支援 `JavaScript` 的能力，也就是說定位在以 `HTML` 內容來呈現內容為主要目的。如果需要與使用者互動的介面，則必須再進一步的設定處理。事實上，如果將與使用者互動的層面處理完整的情況下，`WebView` 不失為一個強而有力的使用者介面元件。



如果 `WebView` 所在的 Activity 會存取網路上的網頁內容，則該 Activity 必須將 `INTERNET` 的使用權限打開。

專案名稱→ `AndroidManifest.xml` 中

```
<uses-permission  
    android:name="android.permission.INTERNET" />
```

■ 基本的處理方式 - 直接放進 Activity 中 ■

這種方式處理 WebView 是將 Activity 呼叫 setContentView() 中設定為一個 WebView 物件，而該 WebView 物件設定載入指定的 Url 字串的 loadUrl() 方法，則將會使其載入指定網址的網頁內容呈現整個 WebView。

```
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        WebView webView = new WebView(this);
        webView.loadUrl("http://www.pcuser.com.tw/");
        setContentView(webView);
    }
}
```

9

常見載入網頁內容三種方式：

1. 直接輸入 Http 網址

```
webView.loadUrl("http://www.pcuser.com.tw/");
```

2. 將網頁內容檔案放在 assets/ 目錄下

```
webView.loadUrl("file:///android_asset/index.html");
```

3. 直接將網頁內容以字串方式載入

```
String data = "<h1>Brad Big Company</h1><hr />歡迎您光臨布萊德大公司";
webView.loadData(data, "text/html; charset=utf-8", null);
```

或

```
String data = "<h1>Brad Big Company</h1><hr />歡迎您光臨布萊德大公司 " ;  
webview.loadDataWithBaseUrl(null, data, null, "utf8", null);
```

■ 基本的處理方式 - 以版面配置方式處理 ■

通常也常見搭配 layout 的 XML 進行配置，增加 Button 元件來處理前後歷程，EditText 元件來負責接收使用者輸入的網址。如果只是將 WebView 元件進行設定 loadUrl()，則將會相當於以 Intent 方式呼叫出行動裝置使用者的預設瀏覽器進行載入網址網頁內容，效果與以下方式類似：(不建議使用)

```
Uri uri = Uri.parse("http://www.pcuser.com.tw");  
Intent intent = new Intent(Intent.ACTION_VIEW, uri);  
startActivity(intent);
```

因此可以透過 WebView 元件呼叫設定 setWebViewClient() 方法，傳遞一個 android.webkit.WebViewClient 物件即可，也可以自行開發撰寫一個 WebViewClient 的子類別物件，更細部的處理網頁內容載入相關程序方法。則 WebView 元件就可以類似瀏覽器相關的行為提供給使用者。

```
WebViewClient client = new WebViewClient();  
webview.setWebViewClient(client)
```

如下範例，以 WebView 處理一般瀏覽器得基本功能：

版面配置

```
res/layout/activity_main.xml  
  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical"  
    >  
    <RelativeLayout  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"
```

```
>
    <Button
        android:id="@+id/back"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:text="向後"
    />
    <Button
        android:id="@+id/forward"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@+id/back"
        android:text="向前"
    />
    <Button
        android:id="@+id/reload"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:text="重載"
    />
</RelativeLayout>
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
>
    <Button
        android:id="@+id/go"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:text="Go"
    />
    <EditText
        android:id="@+id/url"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/go"
        android:layout_alignTop="@+id/go"
        android:layout_alignBottom="@+id/go"
    />
</RelativeLayout>
<WebView
    android:id="@+id/webview"
```

9

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
/>
</LinearLayout>
```

程式處理

```
MainActivity.java package tw.brad.android.books.MyWebViewTest1;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends Activity {
    private Button back, forward, reload, go;
    private EditText url;
    private WebView webview;
    private WebViewClient client;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        webview = (WebView)findViewById(R.id.webview);
        url = (EditText)findViewById(R.id.url);
        back = (Button)findViewById(R.id.back);
        forward = (Button)findViewById(R.id.forward);
        reload = (Button)findViewById(R.id.reload);
        go = (Button)findViewById(R.id.go);

        // 設定按下 [ 向後 ] 的按鈕功能
        back.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                backWebPage();
            }
        });
        // 設定按下 [ 向前 ] 的按鈕功能
```

```
forward.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        forwardWebPage();
    }
});

// 設定按下 [ 重載 ] 的按鈕功能
reload.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        reloadWebPage();
    }
});

// 設定按下 [Go] 的按鈕功能
go.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        goWebPage();
    }
});

// 進行 WebView 元件初始設定
initWebView();

}

// 負責 WebView 初始設定
private void initWebView() {
    client = new WebViewClient();
    webview.setWebViewClient(client);
}

// 執行網頁瀏覽歷程的上一頁
private void backWebPage() {
    webview.goBack();
}

// 執行網頁瀏覽歷程的下一頁
private void forwardWebPage() {
    webview.goForward();
}

// 執行網頁瀏覽的重新載入
private void reloadWebPage() {
    webview.reload();
```

```
    }

    // 執行前往指定網頁網址
    private void goWebPage() {
        String gourl = url.getText().toString();
        gourl = "http://" + gourl;
        webview.loadUrl(gourl);
    }

}
```

清單內容

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="tw.brad.android.books.MyWebViewTest1"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />
    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="tw.brad.android.books.MyWebViewTest1.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

操作使用畫面如下圖：



至於 WebClient 元件的細部處理就是開發撰寫其子類別，例如提供當頁面開始載入時呈現一個 ProgressDialog，而載入完畢後關閉 ProgressDialog。

9

```
// 自訂 WebClient 子類別
private class MyWebClient extends WebClient {
    // 當剛開始載入網頁內容的事件
    @Override
    public void onPageStarted(WebView view, String url, Bitmap favicon) {
        super.onPageStarted(view, url, favicon);
        progress.show();
    }

    // 當已經載入網頁內容完畢的事件
    @Override
    public void onPageFinished(WebView view, String url) {
        super.onPageFinished(view, url);
        progress.dismiss();
    }
}
```

就會有以下的效果呈現



進一步設定 WebView 功能

呼叫 WebView 元件的 `getSettings()` 方法傳回一個 `android.webkit.WebSettings` 物件，經由該物件的設定可以使 WebView 的功能加強，而其中的 JavaScript 功能更是後續的延伸使用。

使 WebView 支援內建縮放尺寸功能

```
settings = webview.getSettings();
settings.setBuiltInZoomControls(true);
```

使 WebView 支援 JavaScript 功能

```
settings.setJavaScriptEnabled(true);
```

使 WebView 不支援密碼儲存機制（使用者安全性考量）

```
settings.setSavePassword(false);
```

JavaScript 的作用是在網頁內容，WebView 是 Android 中的呈現元件，兩者之間的關係就好像是在 PC 上的瀏覽器觀看具有 JavaScript 支援的網頁，但是 WebView 可以使這層關係更進化到 JavaScript 可以傳遞資料給 Android 的

App，而 Android 的 App 其他元件也可以控制到 JavaScript 可以處理的部份，如此一來就可以用網頁來開發 Android App 的版面配置及使用者互動的層面。

將網頁資料傳遞 Android App

首先先來處理簡單的網頁版面如下：(這次的範例將網頁檔案放在專案下的 assets，讀者也可以放在遠端網頁伺服器，載入方式如上文所述。)

```
assests/myweb.html
<script>
    function showData(){
        var data = document.getElementById('data');
        document.getElementById('showhere').innerHTML = data.value;
    }
</script>
<h1>Brad Big Company</h1>
<hr />
<input type="text" id="data" />
<input type="button" onclick="showData()" value="Click" />
<hr />
<div id="showhere"></div>
```

9

這段網頁的處理機制非常簡單，當使用者在 id 為 data 中輸入文字資料後，按下 Click 的按鈕後，將會觸發 onclick 事件，進而執行 showData() 函式功能，而將輸入的文字資料成現在 id 為 showhere 的 div 標籤元素中。而這樣的動作要能夠運作在 Android 中，則必須將 WebSettings 物件的 setJavaScriptEnabled(true) 方法設定，才會生效。

以下解說範例因為網頁是放在專案中，也沒有進行網際網路相關的連接，所以不需要特別去設定 INTERNET 的使用權限。

進行版面配置處理

```
res/layout/activity_main.xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >
```

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    >
    <Button
        android:id="@+id/click1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:text="Click 1"
        />
    <Button
        android:id="@+id/click2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@+id/click1"
        android:text="Click 2"
        />
</RelativeLayout>
<WebView
    android:id="@+id/webview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    />
</LinearLayout>
```

兩個 Button 先預留給後面開發使用，目前先來進行 WebView 的處理。

先來最基本的處理設定：

MainActivity.java

```
package tw.brad.android.book.MyWebViewTest2;

import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class MainActivity extends Activity {
    private WebView webview;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);

        webview = (WebView) findViewById(R.id.webview);

        initWebView();
    }

    private void initWebView() {
        webview.setWebViewClient(new WebViewClient());

        webview.loadUrl("file:///android_asset/myweb.html");
    }

}
```

此時的狀況是可以顯示出指定的網頁內容，但是按下網頁中的 Click 按鈕卻沒有任何效果產生，原因就出在沒有啟用支援 JavaScript 的功能。接下來就改寫 initWebView() 中的程式如下：

```
private void initWebView(){
    webview.setWebViewClient(new WebViewClient());

    WebSettings settings = webview.getSettings();
    settings.setJavaScriptEnabled(true);

    webview.loadUrl("file:///android_asset/myweb.html");
}
```

9

到此處應該可以正常的看到輸入資料後按下 Click 的效果。

但是這只是單純的在網頁中與使用者互動，而接下來希望進一步能將使用者輸入的文字資料傳遞給 Android App 處理。

先來進行 Android 端的接收機制，只需要撰寫非常簡單的類別及方法。但是提供給 JavaScript 呼叫的方法必須是 public 的存取修飾字，切記！

目前放在 MainActivity 類別中的內部類別：

```
private class MyJavaScript {
    private Context c;

    MyJavaScript(Context c){this.c = c;}
```

```
    public void alert(String data){  
        Toast.makeText(c, data, Toast.LENGTH_SHORT).show();  
    }  
}
```

最主要就是提供了 `public void alert()` 來負責接收 JavaScript 的傳遞資料。該類別的物件將會被 `WebView` 物件以呼叫 `addJavascriptInterface()` 方法傳遞參數，並設定其字串名稱。

```
webView.addJavascriptInterface(new MyJavaScript(), "Brad");
```

以上是指定 `MyJavaScript` 物件實體的名稱為 Brad，該名稱將會被 JavaScript 給呼叫使用。

再回到網頁來處理傳送：

assets/myweb.html 中的 JavaScript

```
<script>  
    function showData(){  
        var data = document.getElementById('data');  
        document.getElementById('showhere').innerHTML = data.value;  
  
        window.Brad.alert(data.value);  
        // 或是直接 Brad.alert(data.value); 亦可  
    }  
</script>
```

重點就是呼叫 Brad 的物件之 `alert()` 方法，就相當於呼叫 Android 中 `MyJavaScript` 物件的 `alert()` 方法。

以上的應用面非常廣泛，例如以網頁版面設計輸入表單，而由 Android App 來處理使用者輸入的資料；或是呈現 GoogleMap 的 `MapView` 中，抓到使用者點按位置的經緯度資料，而由 Android App 來儲存到資料庫等等。

由 Android App 傳遞資料給網頁

當使用者在其他不是 `WebView` 元件中，或是存在於程式中的資料，透過 `JavaScript` 來傳遞網頁進行使用者互動。

延續上文的範例，先來處理 JavaScript 的部份，就是撰寫一段方法來處理接收自 Android 的資料：

assets/myweb.html 中的 JavaScript

```
<script>
    // 這段函式負責處理接收資料
    function fromAndroid(msg) {
        document.getElementById("showhere").innerHTML = msg;
    }
</script>
```

回到 Android 程式中，當要傳送給網頁呼叫 JavaScript 函式，一樣呼叫 WebView 物件的 loadUrl() 方法傳遞 "javascript: 函式名稱"。

因此修改如下：

MainActivity.java 中的 onCreate() 方法

```
click1.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        sendToJavaScript1();
    }
});

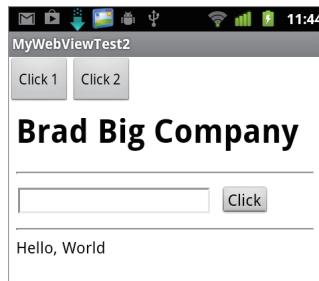
click2.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        sendToJavaScript2();
    }
});
```

9

MainActivity.java 中物件方法

```
private void sendToJavaScript1() {
    webview.loadUrl("javascript:fromAndroid('Hello, World')");
}

private void sendToJavaScript2() {
    webview.loadUrl("javascript:fromAndroid('" + (int)(Math.
random()*49+1)+ "')");
```



分別按下 Click1 按鈕會出現 "Hello,World"；按下 Click2 會出現 1-49 之間的亂數。

可以應用在行動裝置的資料傳遞到網頁內容來處理，例如將使用者目前 GPS 所偵測到的地理位置傳給網頁。

10

Chapter

影音多媒體與相機

- 10-1 播放音樂
- 10-2 音效處理
- 10-3 錄音處理
- 10-4 錄影放影
- 10-5 相機



10-1 播放音樂

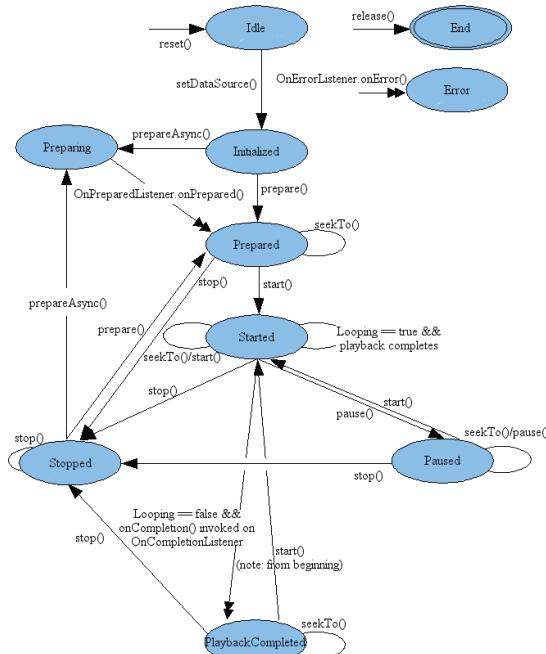
基本概念

在行動裝置上面播放音樂應該算是使用頻率相當高的項目。而在一般應用程式的 App 中可能沒有那麼常用，除非該應用程式就是以播放音樂為主，但是在遊戲 App 中幾乎是不可或缺的功能。

播放音樂的類別主要是透過 `android.media.MediaPlayer` 類別物件實體來進行，通常會有以下幾種播放狀況：

- 播放 SDCard 上面的音樂檔案
- 播放專案資源中的音樂檔案（例如：遊戲中的背景音樂）
- 播放網際網路遠端的音樂檔案

`MediaPlayer` 物件實體播放音樂的狀態圖：



重點說明：

- 圖中的橢圓形表示狀態
- 呼叫 `setDataSource()` 會使其進入初始完成狀態
- 完成初始狀態繼續呼叫 `prepare()` 進入完成準備狀態
- 開始播放音樂是從 `start()` 開始
- 呼叫 `stop()` 會進入停止狀態
- 播放狀態中呼叫 `pause()` 會進入暫停狀態
- 暫停狀態下呼叫 `start()` 後又繼續開始播放
- 播放及暫停狀態下可以呼叫 `seekTo()` 進行跳躍
- 呼叫 `reset()` 會使其進入閒置狀態
- 呼叫 `release()` 會使其進入結束狀態

10

| SDCard 上的音樂播放 |

先假設要播放的音樂檔案放在 SDCard 下的 `brad.mp3`，所以在程式中：

```
Environment.getExternalStorageDirectory().getPath() +  
        "/brad.mp3"
```

開始建構 `MediaPlayer` 物件實體。

```
player = new MediaPlayer();
```

並設定其資料來源。

```
player.setDataSource(Environment.getExternalStorageDirectory()  
.getPath() + "/bg01.mp3");
```

使其進入準備狀態後直接播放。

```
try {  
    player.setDataSource(Environment  
        .getExternalStorageDirectory()  
        .getPath() + "/brad.mp3");
```

```

        player.prepare();
        player.start();

    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

其他常見的控制方法呼叫：

- `setVolume` (左邊喇叭音量 float 值，右邊喇叭音量 float 值)：設定播放音量
- `getDuration()`：傳回目前播放時間 (1/1000 秒單位)
- `seekTo` (跳躍到的時間 1/1000 秒單位)

可以搭配顯示播放進度，利用 `SeekBar` 類別物件實體來實作。先在 `MediaPlayer` 物件實體呼叫 `start()` 方法之後，開始將目前的 `Duration` 值設定給 `SeekBar` 物件實體。

```
seekbar.setMax(player.getDuration());
```

並且透過 `Timer/TimerTask` 來週期將目前播放的進度傳回。

```

private class SeekBarTask extends TimerTask {
    @Override
    public void run() {
        seekbar.setProgress(player.getCurrentPosition());
    }
}

```

而在一開始播放時啟動週期任務。

```
SBTask = new SeekBarTask();
timer.scheduleAtFixedRate(SBTask, 0, 200);
```

接著在 `SeekBar` 物件實體處理 `setOnSeekBarChangeListener()`。

```

seekbar.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {
    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {

```

```
}

@Override
public void onStartTrackingTouch(SeekBar seekBar) {
}

@Override
public void onProgressChanged(SeekBar seekBar, int progress,
    boolean fromUser) {
    if (fromUser && player != null) {
        player.seekTo(progress);

    } else if (!fromUser) {
        info.setText(progress + "/" + seekbar.getMax());
    }
}
});
```

10

播放專案資源中音樂檔案

先將要播放的音樂檔案放進專案下 res/raw/ 子目錄中。假設放了一個 res/raw/brad.mp3，這種類型的播放並不需要以 new 方式建構出 MediaPlayer 物件實體，而是呼叫 MediaPlayer 類別的 static 方法 create()，將會傳回指定資源的 MediaPlayer 物件實體。

```
player = MediaPlayer.create(this, R.raw.brad);
```

如果是遊戲的背景音樂，通常會先呼叫 setLooping(true)，使音樂會不斷地重複播放。當然這種狀況下應該去設計的音樂結尾與開頭會有連續性，才不會使玩家感覺明顯的開始與結束。

```
player.setLooping(true);
player.start();
```

播放 URL 的音樂檔案

播放模式與 SDCard 相同，但是因為遠端音樂檔案相較於 SDCard 上面的檔案，可能會需要較多的準備工作，因此不是呼叫 prepare() 方法，而是非同

步的 `prepareAsync()` 方法。而要等到準備完成之後才能開始 `start()`，但是何時會準備好呢？可以透過 `MediaPlayer` 物件實體設定 `setOnPreparedListener()` 來進行監聽。

```
try {
    player.setDataSource("http://www.ez2test.com/brad.mp3");

    // player.prepare();

    player.prepareAsync();
    player.setOnPreparedListener(new OnPreparedListener() {

        @Override
        public void onPrepared(MediaPlayer mp) {
            player.start();

            seekbar.setMax(player.getDuration());
            SBTTask = new SeekBarTask();
            timer.scheduleAtFixedRate(SBTTask, 0, 200);
        }
    });
} catch (IOException e) {
    e.printStackTrace();
}
```

支援的通訊協定如下：

- RTSP (RTP, SDP)
- HTTP/HTTPS progressive streaming
- HTTP/HTTPS live streaming:
 - MPEG-2 TS media files only
 - Protocol version 3 (Android 4.0 and above)
 - Protocol version 2 (Android 3.x)
 - Not supported before Android 3.0



別忘記要開啟
INTERNET 的
使用權限。

■暫停繼續播放■

應該會先呼叫 `MediaPlayer` 物件實體的 `isPlaying()` 方法傳回 `boolean` 值判斷目前是否播放狀態中。

```

if (player.isPlaying()) {
    // 當作暫停處理
    player.pause();
} else if (!player.isPlaying()) {
    // 當作繼續處理
    player.start();
}

```

停止播放

只是呼叫 stop() 方法，但是如果是應用程式結束的話，則應該呼叫 reset()。另外，如果有使用到 Timer/TimerTask 來處理 SeekBar 物件，也別忘記將 Timer 物件實體取消結束。

```

public void finish() {
    if (player != null && player.isPlaying()) {
        player.reset();
        player = null;
    }
    timer.cancel();
    timer = null;
    super.finish();
}

private void stopMusic() {
    if (player != null) {
        player.stop();
        player = null;
        SBTTask.cancel();
        seekbar.setProgress(0);
    }
}

```

10



10-2 音效處理

音效與音樂不同之處在於音效有較高的即時性，當發射子彈後就要馬上發出咻的音效，稍微延遲就不合遊戲情境。因此處理的觀念就是事先載

入，當要播放音效時直接呼叫，如果還要作準備工作就太遲了。通常會以 SoundPool 來實作。

■ 建構 SoundPool 物件實體 ■

```
spool = new SoundPool(10, AudioManager.STREAM_MUSIC, 0);
```

傳遞以下參數項：

- 最大串流數
- 串流類型
- 取樣轉換值，預設值為 0，目前尚無效果。

將要播放的檔案放在 res/raw/ 子目錄下。先定義播放音效。

```
private int[] sounds = new int[4];
private static final int SOUND_HIT = 0;
private static final int SOUND_GO = 1;
private static final int SOUND_KILL = 2;
private static final int SOUND_OK = 3;
```

開始載入音效檔案，呼叫 load() 方法，傳回特定的整數音效 ID

```
sounds[SOUND_HIT] = spool.load(this, R.raw.bang_1, 1);
sounds[SOUND_GO] = spool.load(this, R.raw.bang_3, 1);
sounds[SOUND_KILL] = spool.load(this, R.raw.cling_steele, 1);
sounds[SOUND_OK] = spool.load(this, R.raw.freebar, 1);
```

■ 即時播放音效 ■

呼叫 SoundPool 物件實體的 play() 方法，傳遞以下參數：

- 事先載入的音效 ID
- 左聲道音量
- 右聲道音量
- 優先順序

- 是否重複
- 取樣值

```
spool.play(sounds[SOUND_HIT], rvol * 0.1f, rvol * 0.1f, 1, 0, 1);
```



10-3 錄音處理

透過行動裝置的麥克風可以進行錄音工作。處理模式大致上有兩種模式：

- 透過 Intent 呼叫其他錄音程式
- 自訂處理錄音程序

錄音工作有兩個權限必須開啟：(Uses Prmission)

10

- 錄音：RECORD_AUDIO
- 寫檔：WRITE_EXTERNAL_STORAGE

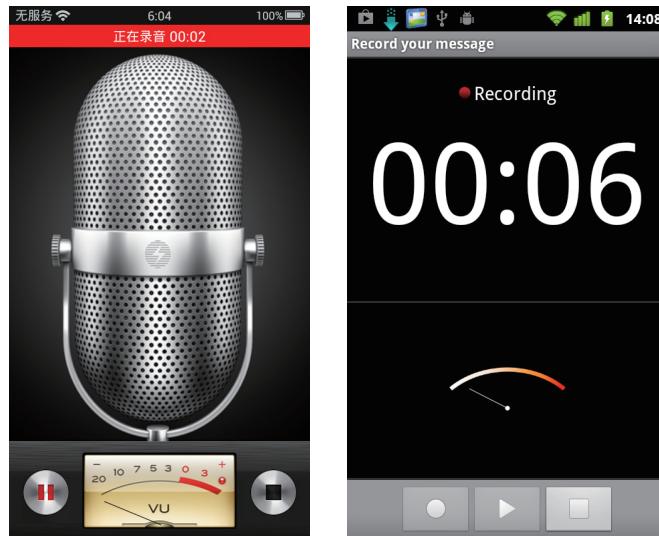
```
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

呼叫其他錄音程式

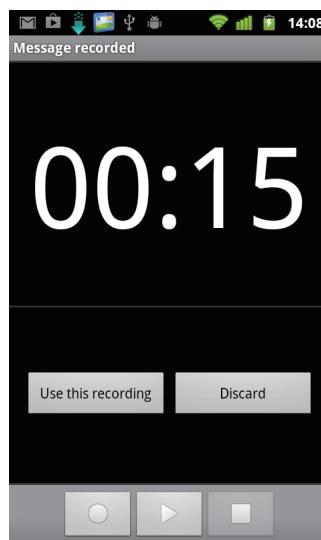
這種模式來處理是比較簡單，只需要透過 Intent 傳遞呼叫即可。

```
Intent intent =
    new Intent(MediaStore.Audio.Media.RECORD_SOUND_ACTION);
startActivityForResult(intent, 0);
```

在不同的行動裝置上的不同呼叫。



錄音完畢之後的存檔對話框：



而錄音完畢之後，會傳回一個 Recult Code 給原本呼叫的 Activity，因此，原本呼叫的 Activity 必須要改寫 onActivityResult() 才能接收其值。錄音存檔完畢，將會回傳值為 RESULT_OK(=1)。並將其資料放在 Intent 物件實體中，透過呼叫 getData() 傳回 android.net.Uri 的物件實體，如果想要進一步取得實體檔案，則必須另外開發撰寫轉換方法。

```
@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    Log.i("brad", "Result: " + resultCode);  
    if (resultCode == RESULT_OK){  
        Uri uri = data.getData();  
        Log.i("brad", uri.getPath());  
        Log.i("brad", getRealPathFromURI(uri));  
        realFile = getRealPathFromURI(uri);  
    }else {  
        Log.i("brad", "Cancel Record");  
    }  
}
```

10

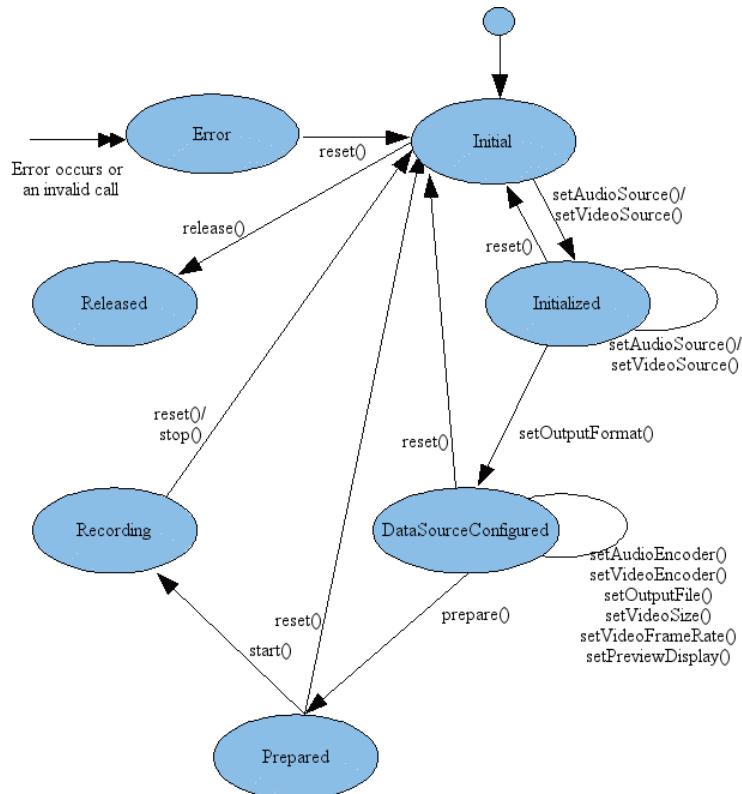
附上一段自訂的轉換方法

```
public String getRealPathFromURI(Uri contentUri) {  
    String[] proj = { MediaStore.Audio.Media.DATA };  
    Cursor cursor = getContentResolver().query(contentUri, proj, null,  
null, null);  
    int column_index = cursor  
        .getColumnIndexOrThrow(MediaStore.Audio.Media.DATA);  
    cursor.moveToFirst();  
    return cursor.getString(column_index);  
}
```

自訂錄音處理程序

自訂錄音處理程序就略為複雜些，要把握住錄音的狀態控制即可。整個運作的主角為 android.media.MediaRecorder 的物件實體，以下先了解其狀態程序圖：

錄音狀態圖



MediaRecorder state diagram

重點說明：

- 進行準備之前的呼叫程序的順序性非常重要
- 先呼叫 set AudioSource() 完成初始化狀態
- 再呼叫 set Output Format() 設定輸出格式，完成資料來源組態
- 再來設定 set Audio Encoder() 編碼器，及 set Ouput File() 輸出檔案
- 呼叫 prepare() 進入準備完成狀態
- 終於可以開始呼叫 start() 進行錄音
- 錄音期間呼叫 stop() 則停止錄音

建構 MediaRecorder 物件實體

```
recorder = new MediaRecorder();
```

開始準播程序及錄音：

```
private void startRecorder() {  
    recorder.setAudioSource(MediaRecorder.AudioSource.MIC);  
    recorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);  
    recorder.setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);  
    recorder.setOutputFile("/mnt/sdcard/bradrecorder.mp4");  
    try {  
        recorder.prepare();  
        recorder.start();  
    } catch (IllegalStateException e) {  
        e.printStackTrace();  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```

10

停止錄音

```
private void stopRecorder() {  
    try {  
        recorder.stop();  
    } catch (IllegalStateException ee) {  
    }  
}
```



10-4 錄影放映

| 錄影 |

錄影的處理模式與錄音非常類似，也是有兩種方式可以處理。

- 呼叫其他錄影程式
- 自訂錄影程序

錄影的處理有三個權限需要開啟：(Uses Prmission)

- 相機：CAMERA (呼叫其他錄影程式時不需要)
- 錄音：RECORD_AUDIO (呼叫其他錄影程式時不需要)
- 寫檔：WRITE_EXTERNAL_STORAGE (兩種處理模式都需要)

呼叫其他錄影程式

原理與錄音類似，但是可以事先設定錄影的檔案，這樣就可以不必在事後進行 Uri 的實體檔名轉換。

先來了解與錄音類似的手法。

```
Intent intent = new Intent(
    android.provider.MediaStore.ACTION_VIDEO_CAPTURE);

startActivityForResult(intent, 1);
```

改寫 onActivityResult() 方法

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == RESULT_OK && requestCode == 1) {
        Uri uri = data.getData();
        Log.i("brad", uri.getPath());
        Log.i("brad", getRealPathFromURI(uri));
    }
}

public String getRealPathFromURI(Uri contentUri) {
    String[] proj = { MediaStore.Audio.Media.DATA };
    Cursor cursor = getContentResolver().query(contentUri, proj, null,
        null, null);
    int column_index = cursor
        .getColumnIndexOrThrow(MediaStore.Audio.Media.DATA);
    cursor.moveToFirst();
    return cursor.getString(column_index);
}
```

也可以事先設定錄影檔案，如下處理模式：

```
private void doVR2() {  
    Intent intent = new Intent(  
        android.provider.MediaStore.ACTION_VIDEO_CAPTURE);  
  
    intent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(  
new File("/mnt/sdcard/ok.3gp")));  
    startActivityForResult(intent, 2);  
}
```

自訂錄影程序

透過相機裝置進行錄影，此時要使用 SurfaceView 來處理螢幕異動更新較為頻繁的狀態。也是與錄音相同的使用 MediaRecorder 物件實體進行。

先來設計錄影的視景，在版面配置檔中，以 SurfaceView 來處理。

```
<SurfaceView  
    android:id="@+id/sv"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" />
```

10

進行重要物件變數宣告：

```
private SurfaceView sv;  
private SurfaceHolder holder;  
private Camera camera;  
private MediaRecorder mr;
```

進行建構物件實體及設定：

```
SetRequestedOrientation(  
ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);  
  
sv = (SurfaceView) findViewById(R.id.sv);  
holder = sv.getHolder();  
holder.addCallback(this);  
holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
```

此時因為 addCallback(this)，所以 Activity 必須宣告 implements android.view.SurfaceHolder.Callback，並且實作以下方法：

```

@Override
public void surfaceChanged(SurfaceHolder holder, int format, int width,
    int height) {
}

@Override
public void surfaceCreated(SurfaceHolder holder) {

}

@Override
public void surfaceDestroyed(SurfaceHolder holder) {
}

```

繼續針對相機及 MediaRecorder 物件進行初始設定。

```

private void initRecorder(Surface surface) {
    if (camera == null) {
        camera = Camera.open();
        camera.unlock();
    }

    if (mr == null) {
        mr = new MediaRecorder();
    }

    mr.setPreviewDisplay(surface);
    mr.setCamera(camera);

    mr.setVideoSource(MediaRecorder.VideoSource.CAMERA);
    mr.setOutputFormat(MediaRecorder.OutputFormat.DEFAULT);
    mr.setOutputFile("/mnt/sdcard/test.mp4");
    mr.setMaxDuration(-1); // 沒有限制錄影時限
    mr.setVideoFrameRate(15);
    mr.setVideoEncoder(MediaRecorder.VideoEncoder.DEFAULT);

    try {
        mr.prepare();
    } catch (IllegalStateException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

```

假設設計一個 Button 第一次按下去開始錄影，第二次按下去結束錄影：

```
private void doVR() {  
    if (isRecording) {  
        mr.stop();  
        mr.reset();  
    } else {  
        initRecorder(holder.getSurface());  
        mr.start();  
    }  
    isRecording = !isRecording;  
}
```

最後，很重要的處理動作，就是針對 App 結束之後的釋放裝置，否則使用者很可能離開程式之後，就無法使用相機，直到下次重新開啟行動裝置。

```
@Override  
public void surfaceDestroyed(SurfaceHolder holder) {  
    if (mr != null) {  
        mr.reset();  
        mr.release();  
        mr = null;  
    }  
    if (camera != null) {  
        camera.release();  
        camera = null;  
    }  
}
```

10

播放影片

可以透過 VideoView 來進行影片播放，先來處理版面配置。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    >  
  
<VideoView  
    android:id="@+id/vv"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    />
```

```
</RelativeLayout>
```

操作方式更為單純簡單，找出 VideoView 物件實體的參考。

```
vv = (VideoView)findViewById(R.id.vv);
```

播放 SDCard：

```
vv.setVideoPath("/mnt/sdcard/test.mp4");
```

播放 res/raw/ 目錄下的資源：

```
vv.setVideoURI(Uri.parse("android.resource://" + getPackageName() + "/" + R.raw.test));
```

播放遠端檔案：

```
vv.setVideoURI(Uri.parse("http://www.ez2test.com/brad.mp4"));
```

開始播放：

```
vv.requestFocus();
vv.start();
```



10-5 相機

相機拍照的處理方式，也可以和錄音錄影一樣，也是有兩種方式處理。

- 呼叫其他照相程式
- 自訂照相程序

照相的處理有兩個權限必須開啟：(Uses Prmission):

- 相機：CAMERA（呼叫其他照相程式時不需要）
- 寫檔：WRITE_EXTERNAL_STORAGE（兩種處理模式都需要）

|| 呼叫其他照相程式 ||

處理模式與錄影類似，但是因為是照相處理，筆者再區分出縮圖和原圖，因為往往在許多應用上，會使用到照相之後的縮圖，而不需要另外在程式中處理。而縮圖與原圖也可以同時一次照相取得，只是為了講解方便而區分說明。

直接以 Intent 物件實體交給系統來呼叫使用者已經安裝的照相程式選單。

```
private void takePic1(){
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    startActivityForResult(intent, 1);
}
```

如下圖：

10



如果使用者之前已經設定了預設相機，或是只有一個相機程式，那就不會出現如上圖的對話框。

當照相完成並在該照相程式中確定，則會回傳 `resultCode` 為 `RESULT_OK`。

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode == RESULT_OK){
        switch (requestCode){
            case 1:
                afterTakePic1(data);
                break;
            case 2:
                afterTakePic2();
                break;
            case 3:
                afterTakePic3();
                break;
        }
    }
}
```

以下分別處理 `afterTakePicX()`。

只取得縮圖：`afterTakePic1()`：

```
private void afterTakePic1(Intent data){
    Bitmap bmp = (Bitmap)data.getExtras().get("data");           // 縮圖
    img.setImageBitmap(bmp);

    // 如果喜歡的話，將該縮圖另存檔案
    try {
        bmp.compress(CompressFormat.JPEG, 85, new FileOutputStream("/mnt/sdcard/camera1.jpg"));
    } catch (FileNotFoundException e) {
    }
}
```

若要直接在照相之前，就先設定照相原檔案的放置，則會在 `Intent` 物件實體中設定。

```
private void takePic2(){
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

    outputFileUri = Uri.fromFile(new File("/mnt/sdcard/camera2.jpg"));
```

```

        intent.putExtra(MediaStore.EXTRA_OUTPUT, outputFileUri);
        startActivityForResult(intent, 2);
    }
}

```

接著處理 afterTakePic2() :

```

private void afterTakePic2(){
    Bitmap bmp = BitmapFactory.decodeFile("/mnt/sdcard/camera2.jpg");
    img.setImageBitmap(bmp);
}

```

自訂相機程式

當然一切就要從頭處理，所以就先從是否有相機裝置，有幾個相機裝置開始下手囉。

10

在 Activity 中呼叫 getPackageManager() 方法傳回 PackageManager 物件實體，再由 PackageManager 物件實體呼叫其 hasSystemFeature() 方法，傳遞 PackageManager.FEATURE_CAMERA 參數，傳回 boolean 值表示是否有相機裝置。其實，不妨藉此機會在這裡稍微了解更多的裝置偵測。例如許多讀者認為使用者的裝置為 API Level 9+，所以就可以使用 NFC，這是不正確的觀念。API 是軟體的支援，而硬體裝置是否有配置是兩回事，所以要透過偵測方式來判斷。

```

private void takePic3() {
    int numCamera = checkCameraHardware(this);
    msg.setText("Camera num:" + numCamera);
    if (numCamera > 0) {

    }
}

// 偵測相機硬體
private int checkCameraHardware(Context context) {
    PackageManager pkgr = getPackageManager();
    if (pkgr.hasSystemFeature(PackageManager.FEATURE_CAMERA)) {
        // 有相機裝置
        return Camera.getNumberOfCameras(); // 傳回相機裝置個數 API Level 9+
    } else {
}

```

```
// 無相機裝置
return 0;
}
}
```

可以繼續針對相機裝置來取得其支援能力。

```
// 取得相機物件實體
public static Camera getCameraInstance(int cid){
    Camera c = null;
    try {
        c = Camera.open(cid);
    }
    catch (Exception e){
    }
    return c;
}

// 偵測相機支援功能
private void checkCameraSupport(){
    Camera c = getCameraInstance(0);
    Camera.Parameters param = c.getParameters();

    c.release();
}
```

開始進行照相的動作。

自訂預覽觀景窗是自訂相機要先處理的部份，這是一個繼承 android.view.SurfaceView 的自訂類別，並且實作 android.view.SurfaceHolder.Callback 介面。

```
CameraPreview.java
package tw.brad.android.book.mycamera;

import java.io.IOException;

import android.content.Context;
import android.hardware.Camera;
import android.util.Log;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

public class CameraPreview extends SurfaceView implements SurfaceHolder.
```

```
Callback {
    private SurfaceHolder mHolder;
    private Camera mCamera;

    public CameraPreview(Context context, Camera camera) {
        super(context);
        mCamera = camera;

        mHolder = getHolder();
        mHolder.addCallback(this);

        mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    }

    public void surfaceCreated(SurfaceHolder holder) {
        try {
            mCamera.setPreviewDisplay(holder);
            mCamera.startPreview();
        } catch (IOException e) {
            Log.d("brad", "預覽觀景窗設定錯誤");
        }
    }

    public void surfaceDestroyed(SurfaceHolder holder) {
        // 結束觀景窗後，也將相機實體釋放
        mCamera.release();
        mCamera = null;
    }

    public void surfaceChanged(SurfaceHolder holder, int format, int w, int h) {
        if (mHolder.getSurface() == null){
            return;
        }
        try {
            mCamera.stopPreview();
        } catch (Exception e){
        }

        try {
            mCamera.setPreviewDisplay(mHolder);
            mCamera.startPreview();

        } catch (Exception e){
            Log.d("brad", "錯誤設定觀景窗");
        }
    }
}
```

10

接著來處理整體的預覽版面配置。

```
res/layout/activity_brad_camera.xml

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal" >

    <FrameLayout
        android:id="@+id/camera_preview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1" />

    <Button
        android:id="@+id/button_capture"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Capture" />

</LinearLayout>
```

Button 是用來觸發照相的動作，而 FrameLayout 是用來放置剛剛處理好的 CameraPreview 用的。

終於可以來寫照相的程序。

```
BradCamera.java

package tw.brad.android.book.mycamera;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;

import android.app.Activity;
import android.hardware.Camera;
import android.hardware.Camera.PictureCallback;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
```

10

```
import android.widget.Button;
import android.widget.FrameLayout;

public class BradCamera extends Activity {

    private Camera mCamera;
    private CameraPreview mPreview;
    private Button capture;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity Brad_camera);

        mCamera = getCameraInstance();

        mPreview = new CameraPreview(this, mCamera);
        FrameLayout preview = (FrameLayout) findViewById(R.id.camera_
            preview);
        preview.addView(mPreview);

        capture = (Button)findViewById(R.id.button_capture);
        capture.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                mCamera.takePicture(null, null, new MyPicCallback());
            }
        });
    }

    private class MyPicCallback implements PictureCallback {

        @Override
        public void onPictureTaken(byte[] data, Camera camera) {
            File pictureFile = new File("/mnt/sdcard/camera3.jpg");
            if (pictureFile == null){
                Log.d("brad", "檔案寫出權限失敗");
                return;
            }

            try {
                FileOutputStream fos = new FileOutputStream(pictureFile);
                fos.write(data);
                fos.close();
            } catch (FileNotFoundException e) {

```

```
        } catch (IOException e) {
    }
}
}

// 取得相機物件實體
public static Camera getCameraInstance(int cid) {
    Camera c = null;
    try {
        c = Camera.open(cid);
    } catch (Exception e) {
    }
    return c;
}
}
```

11

Chapter

地圖與衛星定位系統

- 11-1 GPS 定位
- 11-2 基本 Google Map
- 11-3 進階 Google Map





11-1 GPS 定位

透過 Android 行動裝置的衛星定位系統，可以取得目前所在的位置資訊，通常就是使用經緯度來表示出定位資訊；定位資訊的取得方式通常有兩種：GPS 及網路，GPS 僅適用戶外的環境下，因此使用上大受限制；而使用行動網路方式定位，相對的比較不夠精確，優點就是比較不耗電。

比較項目 / 定位方式	GPS	Network
精確度	高	差
耗電量	高	低
高度 (海拔)	有	無
速度	有	無

透過取得的經緯度資訊，搭配使用 Google Map，可以讓使用者輕鬆的了解目前所在位置的相關地理資訊。常見的應用就是將所在位置附近的商家或是旅遊景點提供詳細資料，或是將定位路線產生路徑圖的呈現等等。

開啟使用者權限如下：

- GPS 定位 : ACCESS_FINE_LOCATION
- Network 定位 : ACCESS_COARSE_LOCATION

開始基本實作

宣告使用 LocationManager 類別物件，該物件使整個定位系統的主要管理物件。

```
private LocationManager lmgr;
```

透過呼叫 getSystemService() 方法，傳入 LOCATION_SERVICE 參數，將傳回值強制轉型為 LocationManager。

```
lmgr = (LocationManager) getSystemService(LOCATION_SERVICE);
```

而定位的提供者有兩個變數來處理：

```
locationProvider = LocationManager.NETWORK_PROVIDER;  
locationProvider = LocationManager.GPS_PROVIDER;
```

接著就開始處理定位監聽物件，只需要自訂類別 implements android.location.LocationListener 介面即可：

```
private class MyLocationListener implements LocationListener {  
  
    @Override  
    public void onLocationChanged(Location location) {}  
  
    @Override  
    public void onProviderDisabled(String provider) {}  
  
    @Override  
    public void onProviderEnabled(String provider) {}  
  
    @Override  
    public void onStatusChanged(String provider, int status, Bundle extras) {}  
  
}
```

11

其中最主要實作的方法就是 onLocationChanged()，該方法會在使用者的行動裝置的定位位置移動改變，而觸發傳入的參數 Location 物件實體，涵蓋了目前位置的相關資訊，常見如下資料：

- getAccuracy()：傳回精準度，其數值單位為公尺 (float)
- getAltitude()：傳回海拔高度，其數值單位為公尺 (double)
- getBearing()：傳回方位角度，其數值單位為角度 0.0-360.0 (float)
- getLatitude()：傳回緯度，其數值單位為度 (double)
- getLongitude()：傳回經度，其數值單位為角度 (double)

- `getSpeed()`：傳回速度，其數值單位為公尺 / 秒 (float)
- `getTime()`：傳回時間值，表示從 1970 年 1 月 1 日起算的千分之一秒

最後就透過 `LocationManager` 物件實體呼叫 `requestLocationUpdates()` 方法，傳入以下參數：

- 定位提供者
- 位移更新最小時間，單位為千分之一秒
- 位移更新最小距離，單位為公尺
- 定位監聽物件實體

```
mll = new MyLocationListener();
lmgr.requestLocationUpdates(locationProvider, 0, 0, mll);
```

當不再需要使用 GPS 定位時，記得將該定位監聽物件解除註冊。

```
lmgr.removeUpdates(mll);
```

|| 較佳位置取得 ||

此段內容參考 Google Android 開發者網站的文件，試圖在 GPS 與 Network 的定位上進行切換，取得較佳的位置資訊內容。

建立一個自訂類別，用來處理使用者的位置物件相關屬性及方法。該類別定義了幾個重要的屬性：

- 排程計時器物件 `Timer`，用來週期來處理定位程序
- 定位管理員 `LocationManager` 物件
- 自訂定位結果物件 `LocationResult`

如下簡單的程式碼 `MyLocation.java`

```
package tw.brad.android.book.mygpsandmapv3;

import java.util.Timer;
import java.util.TimerTask;
import android.content.Context;
```

```
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;

public class MyLocation {
    Timer timer1;
    LocationManager lm;
    LocationResult locationResult;
    boolean gps_enabled = false;
    boolean network_enabled = false;

    public boolean getLocation(Context context, LocationResult result) {
        locationResult = result;
        if (lm == null)
            lm = (LocationManager) context
                .getSystemService(Context.LOCATION_SERVICE);

        try {
            gps_enabled = lm.isProviderEnabled(LocationManager.GPS_
                PROVIDER);
        } catch (Exception ex) {
        }
        try {
            network_enabled = lm
                .isProviderEnabled(LocationManager.NETWORK_PROVIDER);
        } catch (Exception ex) {
        }

        if (!gps_enabled && !network_enabled)
            return false;

        if (gps_enabled)
            lm.requestLocationUpdates(LocationManager.GPS_PROVIDER,
                0, 0, locationListenerGps);
        if (network_enabled)
            lm.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0,
                locationListenerNetwork);
        timer1 = new Timer();
        timer1.schedule(new GetLastLocation(), 20000);
        return true;
    }

    LocationListener locationListenerGps = new LocationListener() {
        public void onLocationChanged(Location location) {
```

```
        timer1.cancel();
        locationResult.getLocation(location);
        lm.removeUpdates(this);
        lm.removeUpdates(locationListenerNetwork);
    }

    public void onProviderDisabled(String provider) {
    }

    public void onProviderEnabled(String provider) {
    }

    public void onStatusChanged(String provider, int status,
        Bundle extras) {
    }
};

LocationListener locationListenerNetwork = new LocationListener() {
    public void onLocationChanged(Location location) {
        timer1.cancel();
        locationResult.getLocation(location);
        lm.removeUpdates(this);
        lm.removeUpdates(locationListenerGps);
    }

    public void onProviderDisabled(String provider) {
    }

    public void onProviderEnabled(String provider) {
    }

    public void onStatusChanged(String provider, int status,
        Bundle extras) {
    }
};

class GetLastLocation extends TimerTask {
    @Override
    public void run() {
        lm.removeUpdates(locationListenerGps);
        lm.removeUpdates(locationListenerNetwork);

        Location net_loc = null, gps_loc = null;
        if (gps_enabled)
            gps_loc = lm.getLastKnownLocation(LocationManager.GPS_PROVIDER);
        if (network_enabled)
```

```
        net_loc = lm
        .getLastKnownLocation(LocationManager.NETWORK_PROVIDER);

        if (gps_loc != null && net_loc != null) {
            if (gps_loc.getTime() > net_loc.getTime())
                locationResult.gotLocation(gps_loc);
            else
                locationResult.gotLocation(net_loc);
            return;
        }

        if (gps_loc != null) {
            locationResult.gotLocation(gps_loc);
            return;
        }
        if (net_loc != null) {
            locationResult.gotLocation(net_loc);
            return;
        }
        locationResult.gotLocation(null);
    }
}

public static abstract class LocationResult {
    public abstract void gotLocation(Location location);
}
```

11

回到自行開發的專案程式中處理。

先實作出 LocationResult。

```
LocationResult locationResult = new LocationResult() {
    @Override
    public void gotLocation(Location location) {
        nowLat = location.getLatitude();
        nowLng = location.getLongitude();
    }
};
```

說明：

- nowLat 為事先定義的 double 變數，表示緯度
- nowLng 為事先定義的 double 變數，表示經度

先建構出 MyLocation 物件實體，並呼叫 getLocation() 方法。

```
MyLocation myLocation = new MyLocation();
myLocation.getLocation(this, locationResult);
```

就可以開始進行 Location 物件實體資料的取得。

另外，也提供了一個位置物件實體的最佳比較方法，以及判斷是否相同提供者的判斷。

先定義時間差值：(目前設定為兩分鐘)

```
private static final int TWO_MINUTES = 1000 * 60 * 2;
```

方法 isBetterLocation()：

```
protected boolean isBetterLocation(Location location,
        Location currentBestLocation) {
    if (currentBestLocation == null) {
        // 預設以新抓到的位置為最佳位置物件
        return true;
    }

    // 檢查修正後新位置是否較新
    long timeDelta = location.getTime() - currentBestLocation.getTime();
    boolean isSignificantlyNewer = timeDelta > TWO_MINUTES;
    boolean isSignificantlyOlder = timeDelta < -TWO_MINUTES;
    boolean isNewer = timeDelta > 0;

    // 判斷是否距離上次位置時間為兩分鐘以上，是的話就對了，因為使用者應該是在移動狀態
    if (isSignificantlyNewer) {
        return true;
    } else if (isSignificantlyOlder) {
        return false;
    }

    int accuracyDelta = (int) (location.getAccuracy() - currentBestLocation
            .getAccuracy());
    boolean isLessAccurate = accuracyDelta > 0;
    boolean isMoreAccurate = accuracyDelta < 0;
    boolean isSignificantlyLessAccurate = accuracyDelta > 200;

    // 檢查兩個位置的取得是否相同的提供者
```

```
boolean isFromSameProvider = isSameProvider(location.getProvider(),
                                             currentBestLocation.getProvider());

        if (isMoreAccurate) {
            return true;
        } else if (isNewer && !isLessAccurate) {
            return true;
        } else if (isNewer && !isSignificantlyLessAccurate
                   && isFromSameProvider) {
            return true;
        }
        return false;
    }
```

方法 isSameProvider() :

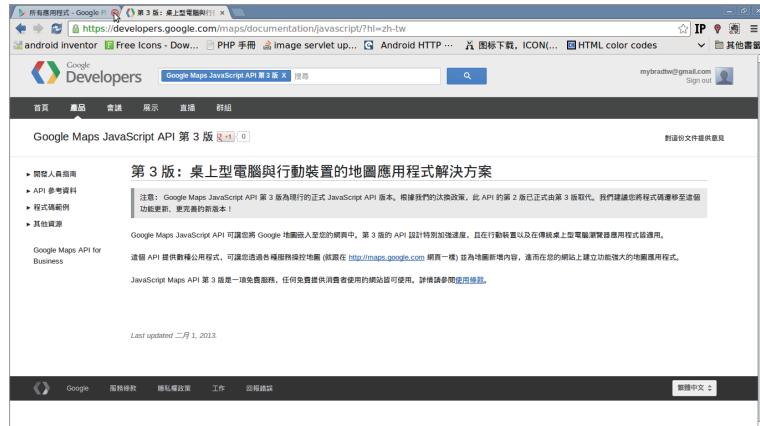
11

```
private boolean isSameProvider(String provider1, String provider2) {
    if (provider1 == null) {
        return provider2 == null;
    }
    return provider1.equals(provider2);
}
```

11-2

基本 Google Map

Google 自 2012.12.3 起不再支援 Maps API v1；而到了 2013.3.3 前仍接受申請 Maps API v1 key。Maps API v2 改用 com.google.android.gms.maps.MapFragment，仍需申請 Maps API v2 key；Google Maps API v3 則改為 Javascript API 方式，使用 WebView 來顯示 Google Map，應用程式不需再申請 Google Maps API key，但是對於地圖呈現的 API 仍要使用 API Key，其使用限制為同一頁面免費存取次數為 25,000 次 / 天。



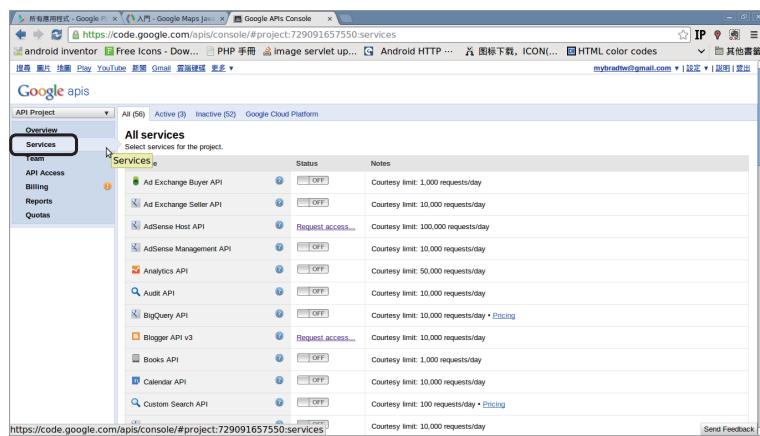
因此本書將重點直接放在 v3 的版本的開發模式。

開發前置作業

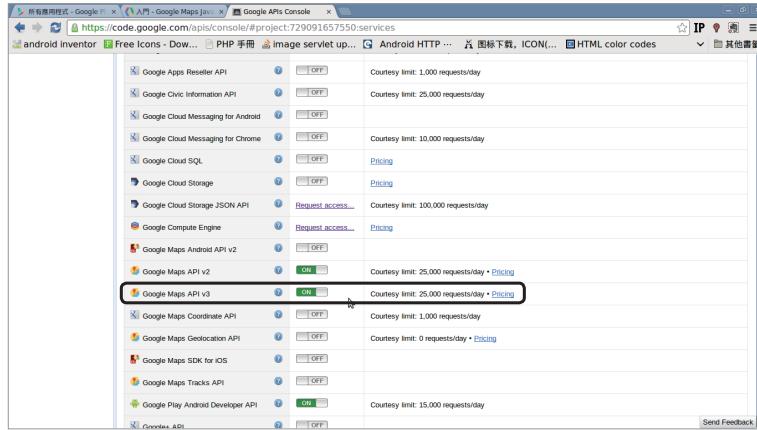
註冊申請 Google 帳號，以該帳號繼續申請開發者身份。

前往：<https://code.google.com/apis/console>

註冊之後，進入以下網頁，並點選左側的 Services。

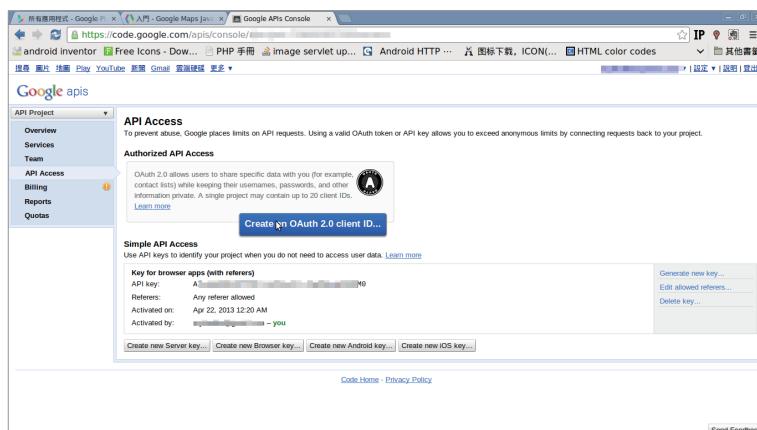


右側網頁往下移動，直到看到 Google Maps API v3：



11

將其開關切到 ON 即可，接著點選左側的 API Access 項目，將看到如下
的資料畫面：



在 Simple API Access 中的 API Key 的資料就是開發上要用的 API Key。

Hello, Map

先在網頁檔案中進行測試，利用官方開發網站的 Hello, Map 來實測。

showmap.html

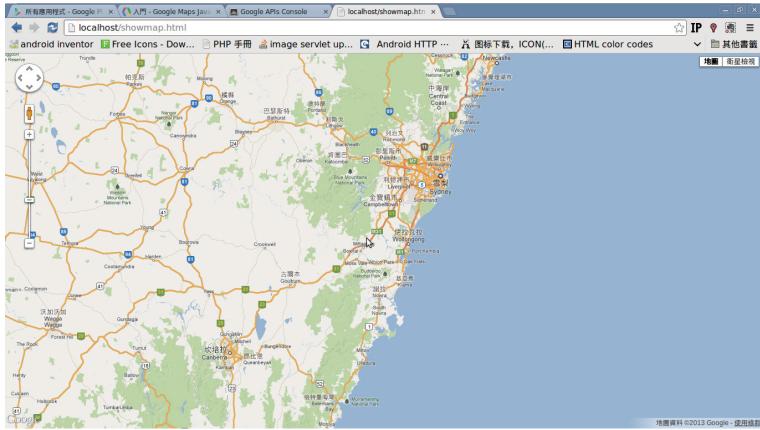
```

<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
<style type="text/css">
    html {
        height: 100%
    }
    body {
        height: 100%;
        margin: 0;
        padding: 0
    }
    #map_canvas {
        height: 100%
    }
</style>
<script type="text/javascript"
    src="http://maps.googleapis.com/maps/api/js?v=3&key=API Key 放在此處
&sensor=false">
</script>
<script type="text/javascript">
    var map;
    function initialize() {
        var mapOptions = {
            center: new google.maps.LatLng(-34.397, 150.644),
            zoom: 8,
            mapTypeId: google.maps.MapTypeId.ROADMAP
        };
        map = new google.maps.Map(document.getElementById("map_
canvas"), mapOptions);
    }
</script>
</head>
<body onload="initialize()">
    <div id="map_canvas" style="width: 100%; height: 100%"></div>
</body>
</html>

```

筆者針對 map 的物件變數稍作修改，將其定義為全域變數，因為後續還會有其他 JavaScript 的自訂函數要對該變數進行相關存取設定。

透過瀏覽器開啟，應該會看到如下澳洲雪梨的地圖：



11

至此就算是完成了基本的前置作業。

| 在 Android 開發上的應用 |

因為要取得相關的地圖資料是從網際網路通訊傳輸，所以先在開發專案上設定使用權限 INTERNET。而在版面配置上是以 WebView 元件進行處理。

可將含有 Google Map 的 html 檔案，放在遠端伺服器，或是專案資源中。

先來實作遠端伺服器的處理模式，假設放在：<http://android.ez2test.com/showmap.html>

```
private void initWebView() {
    webview.setWebViewClient(new WebViewClient());

    WebSettings settings = webview.getSettings();
    settings.setJavaScriptEnabled(true);

    webview.addJavascriptInterface(new FromJavaScript(this), "Android");

    webview.loadUrl("http://android.ez2test.com/showmap.html");
    webview.setVisibility(View.INVISIBLE);
}
```

而如果放在專案資源中的處理方式，可以將 html 檔案放在專案目錄下的 assets/ 子目錄中。

```
private void initWebView() {  
    webview.setWebViewClient(new WebViewClient());  
  
    WebSettings settings = webview.getSettings();  
    settings.setJavaScriptEnabled(true);  
  
    webview.addJavascriptInterface(new FromJavaScript(this), "Android");  
  
    webview.loadUrl("file:///android_asset/showmap.html");  
    webview.setVisibility(View.INVISIBLE);  
}
```

在行動裝置上呈現如下的畫面（範例檔案修改為取得定位資訊）：



接下來通常開發的重點分成三個部份：

- 以 JavaScript 來處理使用者與地圖物件的互動，例如點按特定位置出現標記（Marker）
- JavaScript 將使用者的地圖相關資料，傳遞給 Android 繼續處理。例如使用者的標記傳遞給 Android 的 SQLite 資料庫儲存
- Android 將訊息資料傳遞給 JavaScript，例如上一小節取得的定位經緯度資料傳給 JavaScript，並使地圖置中於定位位置

11-3

進階 Google Map

11

接著針對上一小節的重點開發進行實作。

JavaScript 處理說明

宣告及使用 Google Maps API v3

```
<script type="text/javascript"
       src="http://maps.googleapis.com/maps/api/js?v=3&key=API Key&sensor=false">
</script>
```

定義變數 map：

```
var map;
```

而 google.maps.Map 物件的建構會使用到 google.maps.MapOptions 物件，因此先建構：

```
var mapOptions = {
  center: new google.maps.LatLng(-34.397, 150.644),
  zoom: 14,
  mapTypeId: google.maps.MapTypeId.ROADMAP
};
```

設定三個基本屬性：

- center：定位中心位置
- zoom：縮放值
- mapTypeId：地圖類型

定位中心位置為 `google.maps.LatLng` 的經緯度類別物件實體，所以直接傳入緯度與精度的數值來建構出來。縮放值越大越接近地面；越小越遠離地面。

以下是 Google Maps API 提供的地圖類型：

- `MapTypeId.ROADMAP`：會顯示預設道路地圖檢視畫面
- `MapTypeId.SATELLITE`：會顯示 Google 地球衛星影像
- `MapTypeId.HYBRID`：會顯示一般檢視和衛星檢視的混合畫面
- `MapTypeId.TERRAIN`：會根據地形資訊顯示實際地圖

建立最重要的 `map` 物件實體：

```
map = new google.maps.Map(document.getElementById("map_canvas"), mapOptions);
```

試著練習如何處理地圖事件，當使用者按下地圖特定位置，使其增加一個標記（Marker）在地圖上面。

先開發一段 JavaScript 的自訂函數 `addMarker()`

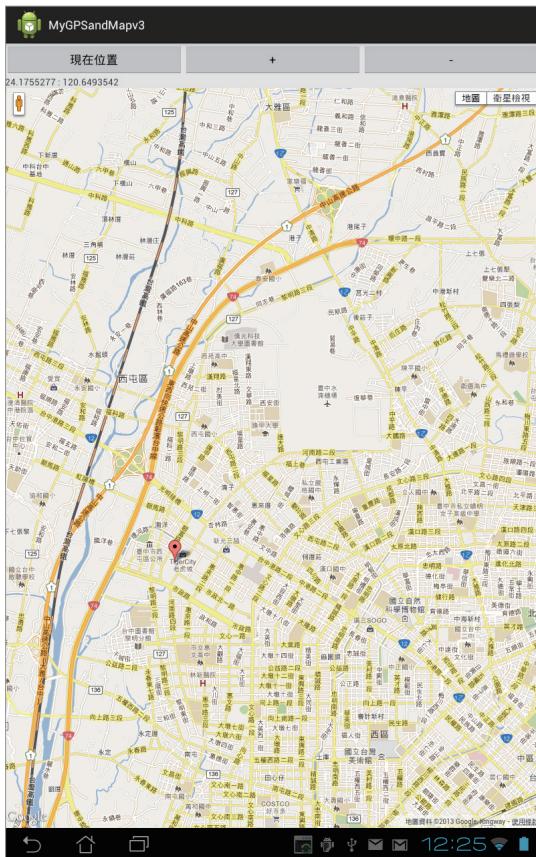
```
function addMarker(latLng) {
    new google.maps.Marker({
        map: map,
        position: latLng
    });
}
```

回到 `initialize()` 中增加上事件處理，針對 `map` 發生 `click` 事件來呼叫 `addMarker()`：

```
function initialize() {
    var mapOptions = {
```

```
center: new google.maps.LatLng(-34.397, 150.644),  
zoom: 14,  
mapTypeId: google.maps.MapTypeId.ROADMAP  
};  
map = new google.maps.Map(document.getElementById("map_canvas"),  
mapOptions);  
  
google.maps.event.addListener(map, "click", function(event){  
    addMarker(event.latLng);  
});  
}
```

如此一來使用者就可以觸摸特定位置並標記。



JavaScript 資料傳回 Android

先在 Android 程式中開發自訂類別

```
private class FromJavaScript {
    private Context c;

    FromJavaScript(Context c) {
        this.c = c;
    }

    public void getLatLng(String data) {
        Log.i("brad", data);
    }
}
```

定義一個方法 `getLatLng()`，傳遞的字串參數就是從 JavaScript 傳來的值。方法中將該字串資料以 `Log()` 顯示。

使 `webview` 增加該物件的處理介面，並定義 Android 字串為呼叫的物件名稱：

```
webview.addJavascriptInterface(new FromJavaScript(this), "Android");
```

回到 JavaScript 中的 `addMarker()`：

```
function addMarker(latLng) {
    new google.maps.Marker({
        map: map,
        position: latLng
    });
    Android.getLatLng(latLng.lat() + " : " + latLng.lng());
}
```

以物件變數名稱為 "Android" 物件來呼叫 `getLatLng()` 方法。這樣就可以將使用者點擊的特定位置，先在 `map` 上面標記，並傳回給 Android 繼續處理。

以 Android 傳遞資料給 JavaScript

這種情況很多時候發生在對 JavaScript 的自訂函數進行呼叫，所以先定義一些自訂函數出來，移到指定中心位置：

```
function moveTo(lat, lng){  
    var newpos = new google.maps.LatLng(lat, lng);  
    map.panTo(newpos);  
}
```

放大視景：

```
function zoomIn(){  
    var zoom = map.getZoom();  
    if (zoom<21)zoom++;  
    map.setZoom(zoom);  
}
```

縮小視景：

```
function zoomOut(){  
    var zoom = map.getZoom();  
    if (zoom>1)zoom--;  
    map.setZoom(zoom);  
}
```

11

回到 Android 中，只需要呼叫 webview 的 loadUrl() 即可，

改回之前以 GPS 取得定位：

```
LocationResult locationResult = new LocationResult() {  
    @Override  
    public void gotLocation(Location location) {  
        nowLat = location.getLatitude();  
        nowLng = location.getLongitude();  
        info.setText(nowLat + " : " + nowLng);  
        webview.loadUrl("javascript:moveTo(" + nowLat + "," + nowLng + ")");  
    }  
};
```

或是

```
webview.loadUrl("javascript:zoomIn()");  
webview.loadUrl("javascript:zoomOut()");
```

MEMO

12

Chapter

感應器運作原理及應用

- 12-1 感應器運作原理與應用
- 12-2 三軸加速感應器
- 12-3 重力加速度感應器
- 12-4 磁極方向感應器
- 12-5 光線 / 溫度 / 濕度 / 壓力感應器





12-1 感應器運作原理與應用

■ 基本概念 ■

事實上這個章節所要討論的感應器 Sensor，指的是透過 android.hardware.Sensor 的 API 與實際硬體的感應器，兩者之間所回應的數據資料。

在目前的行動裝置上面，或多或少都配備一些感應器設備。而硬體配備與否，先天就決定了這個行動裝置是否可以支援使用該項感應器裝置；後天上還需要軟體的支援，也就是其 Android API Level 是否支援，兩相搭配之下，才能透過該項裝置的感應器來取得相關的數據資料。

原則上可以分成三種類型的感應器：

- 位移感應器
 - 三軸加速感應器
 - 重力加速度感應器
 - 陀螺儀感應器
 - 線性加速感應器
 - 旋轉向量感應器
- 環境感應器
 - 光線感應器
 - 溫度感應器
 - 相對濕度感應器
 - 大氣壓力感應器
- 位置感應器
 - 方向感應器
 - 磁極感應器
 - 接近感應器

處理原則

處理的原則如下：

- 呼叫 `getSystemService()` 方法，傳遞 `SENSOR_SERVICE`，取得 `android.hardware.SensorManager` 的物件實體
- 再由 `SensorManager` 物件實體呼叫 `getDefaultSensor()` 方法，傳遞指定的 `Sensor` 型態的 `int` 值，傳回 `Sensor` 物件實體
- 如果傳回的 `Sensor` 物件實體為 `null`，就表示不支援該項感應器
- 建立一個自訂類別實作 `android.hardware.SensorEventListener` 介面
- Override 其 `onSensorChange()` 方法
- 將傳遞來的 `SensorEvent` 物件實體，取得該感應器的數據資料
- 最後由 `SensorManager` 物件實體呼叫 `registerListener()` 方法，設定 `SensorEventListener` 物件及週期頻率即可
- 當不再使用該感應器時，要記得 `SensorManager` 物件實體呼叫 `unregisterListener()` 方法

12

實作開發

宣告物件變數

```
static SensorManager smgr;
private Sensor sensor;
private boolean isSensorEnable;
```

偵測是否支援：

```
sensor = smgr.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
// 偵測指定的感應器是否支援
if (sensor != null) {
    isSensorEnable = true;
}
```

開發自訂類別處理感應器事件監聽器

```

private class MySensorEventListener implements SensorEventListener {

    // 精準度改變事件
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }

    // 感應事件
    @Override
    public void onSensorChanged(SensorEvent event) {
    }

}

```

註冊與解除註冊

```

@Override
protected void onResume() {
    super.onResume();
    if (isSensorEnable) {
        smgr.registerListener(listener, sensor,
            SensorManager.SENSOR_DELAY_UI);
    }
}

@Override
protected void onPause() {
    super.onPause();
    if (listener != null) {
        smgr.unregisterListener(listener);
    }
}

```

感應器監聽的頻率設定如下

- SensorManagerSENSOR_DELAY_NORMAL: 0.2 秒
- SensorManagerSENSOR_DELAY_UI: 0.06 秒
- SensorManagerSENSOR_DELAY_GAME: 0.02 秒
- SensorManagerSENSOR_DELAY_FASTEST: 0 秒
- API Level 11+ 可以自行指定微秒單位

開發的重點在於自訂類別處理感應器事件監聽器中，以 SensorEvent 物件實體呼叫其屬性 values，傳回一個 float[] 陣列的值。

■ 使用者裝置支援處理 ■

對於開發這而言，既然存在這樣的狀況，而又不想增加開發上的考量，則可以在 Androidmanifest.xml 檔案中針對特定的感應器，設定其屬性值。使得使用者如果透過 Google play 的市集下載安裝之前，可以得知使用者的行動裝置是否支援 App 中的需求。通常這樣的處理方式是 App 中對於該項感應器裝置是必要條件，也就是說如果沒有該項感應器裝置，就完全無法使用 App 的狀況。如果沒有該項特定的感應器裝置仍有其他方式可以替代，則不建議如此進行設定。

12

假設要偵測的感應器為三軸加速感應器，則如下設定在 AndroidManifest.xml：

```
<uses-feature  
    android:name="android.hardware.sensor.accelerometer"  
    android:required="true" />
```

其他：

- 三軸加速感應器：accelerometer
- 環境溫度濕度：barometer
- 方向羅盤感應器 compass
- 陀螺儀感應器：gyroscope
- 光線感應器：light
- 接近感應器：proximity



12-2 三軸加速感應器

利用行動裝置所在的空間座標而感應其移動加速度的感應器。

當行動裝置對於使用者以直向握持的狀態而言，左右移動則為 X 座標軸，前後移動移動則為 Y 座標軸，上下移動則為 Z 座標軸。其值的單位為每秒平方所移動的公尺距離。

而其各座標軸的值，是透過 SensorEvent.values 屬性取得的 float[] 陣列來的：

- SensorEvent.values[0] => X 座標軸
- SensorEvent.values[1] => Y 座標軸
- SensorEvent.values[2] => Z 座標軸

接下來直接進行開發實作。實作方式不打算將數據資料直接呈現出來，因為即使是敏感度較差的狀況，也將會是每 0.2 秒更新資料，不易判斷處理；甚至將數據資料以數學方式重新整理，也只是數字。因此筆者以自訂 View 來繪製幾何圖形呈現，應該會比較有感。

通常三軸加速感應器的 XY 座標軸較為常被應用，因此以下實作以 XY 座標為主。

先進行基本偵測，傳遞參數為 Sensor.TYPE_ACCELEROMETER：

```
sensor = smgr.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
// 偵測指定的感應器是否支援
if (sensor != null) {
    isSensorEnable = true;
    listener = new MySensorEventListener();
}
```

建立一個自訂 View，想要將三軸加速感應器反應在一個幾何圓上面。

```
private class TestView extends View {
    private int viewW, viewH;
```

```
private boolean isInited;
private Paint paintText, paintValue;
float cx, cy, cr;

public TestView(Context context) {
    super(context);
    setBackgroundColor(Color.BLACK);

    paintText = new Paint();
    paintText.setTextSize(36);
    paintText.setColor(Color.YELLOW);

    paintValue = new Paint();
    paintValue.setTextSize(36);
    paintValue.setColor(Color.RED);
}

private void init() {
    viewW = getWidth();
    viewH = getHeight();

    cx = viewW / 2;
    cy = viewH / 2;
    cr = 50;

    isInited = true;
}

@Override
protected void onDraw(Canvas canvas) {
    if (!isInited) {
        init();
        canvas.drawText(
            isSensorEnable ? "三軸加速器支援" + rotation : "裝
            置不支援",
            viewW / 4, viewH / 4, paintText);
    }
    canvas.drawCircle(cx, cy, cr, paintText);
}
}
```

重點說明：

- 背景設定為全黑
- 畫一個幾何圓，一開始的位置在正中央，半徑為 50 像素

所以在 onCreate() 方法中整理如下：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    tv = new TestView(this);
    setContentView(tv);

    rotation = getWindowManager().getDefaultDisplay().getRotation();

    smgr = MainActivity.smgr;
    sensor = smgr.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    // 偵測指定的感應器是否支援
    if (sensor != null) {
        isSensorEnable = true;
        listener = new MySensorEventListener();
    }
}
```

開發自訂感應器事件監聽器：

```
private class MySensorEventListener implements SensorEventListener {
    // 精準度改變事件
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }

    // 感應事件
    @Override
    public void onSensorChanged(SensorEvent event) {
        x = event.values[0] * 7;
        y = event.values[1] * 7;
        z = event.values[2];

        if (rotation == 0) {
            sx = x;
            sy = y;
            sz = z;
        } else if (rotation == 3) {
            sx = y;
            sy = -1 * x;
            sz = z;
        }
    }
}
```

```

        tv.addCX(-1 * sx);
        tv.addCY(sy);

        tv.postInvalidate();
    }

}

```

重點說明：

- tv 為 TestView 的物件實體，當有改變資料，則呼叫 TestView 重新繪製
處理感應器事件監聽器的註冊與解除：

12

```

@Override
protected void onResume() {
    super.onResume();
    if (isSensorEnable) {
        smgr.registerListener(listener, sensor,
            SensorManager.SENSOR_DELAY_UI);
    }
}

@Override
protected void onPause() {
    super.onPause();
    if (listener != null) {
        smgr.unregisterListener(listener);
    }
}

```

其中特別的地方是有一個 rotation 的 int 變數，其值是透過呼叫 get WindowManager().getDefaultDisplay().getRotation() 而來。在本單元的範例是以垂直方向執行，而在一般手機的狀態下，其值為 0，但是相同的設定下，在平板電腦（以 TF-101 為例）下因為預設為水平方向，所以得到其值為 3，所以要將 XY 座標軸作調整轉換，這樣可以使該應用程式適用於不同的行動裝置。

正常執行下應該會看到一個幾何圓形，會隨著使用者前後左右翻轉而移動，可以應用在控制方向的應用。



12-3 重力加速度感應器

利用行動裝置所在的空間座標而感應其重力加速度的感應器。(Sensor.TYPE_GRAVITY)

座標軸的觀念與三軸加速度感應器是相同的，而其靜置狀態下，Z座標軸的值大約在 9.8 每秒平方公尺單位，也就是一個 g 值。

一樣是以一個幾何圖形的圓來呈現其效果，當感應偵測 Z 座標軸的改變，會即時變更圓的半徑大小。當使用者晃動行動裝置的同時，就會變更圓的大小，若是整個行動裝置反面過來，則會得到負值，使得原本是黃色的圓，轉變成紅色的圓。

應用舉例：

- 數值變化為晃動手機，模擬為擲筊，或是釣魚甩竿
- 數值負數為翻面，觸發的同時暫停遊戲進行，或是來電鈴聲停止

```
package tw.brad.bookmysensor;

import android.app.Activity;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.view.View;

public class GravityTest extends Activity {
    private TestView tv;
    private SensorManager smgr;
    private Sensor sensor;
    private boolean isSensorEnable;
    private int rotation;
    private MySensorEventListener listener = null;
```

12

```
private float x, y, z, sx, sy, sz;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    tv = new TestView(this);
    setContentView(tv);

    rotation = getWindowManager().getDefaultDisplay().getRotation();

    smgr = MainActivity.smgr;
    sensor = smgr.getDefaultSensor(Sensor.TYPE_GRAVITY);
    // 偵測指定的感應器是否支援
    if (sensor != null) {
        isSensorEnable = true;
        listener = new MySensorEventListener();
    }
}

@Override
protected void onResume() {
    super.onResume();
    if (isSensorEnable) {
        smgr.registerListener(listener, sensor,
            SensorManager.SENSOR_DELAY_UI);
    }
}

@Override
protected void onPause() {
    super.onPause();
    if (listener != null) {
        smgr.unregisterListener(listener);
    }
}

private class MySensorEventListener implements SensorEventListener {
    // 精準度改變事件
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }

    // 感應事件
    @Override
    public void onSensorChanged(SensorEvent event) {
```

```
x = event.values[0] * 100;
y = event.values[1] * 100;
z = event.values[2];

if (rotation == 0){
    sx = x;
    sy = y;
    sz = z;
} else if (rotation == 3){
    sx = -1*y;
    sy = -1*x;
    sz = z;
}

tv.cr = z * 10 + 30;

tv.postInvalidate();
}

}

private class TestView extends View {
    private int viewW, viewH;
    private boolean isInited;
    private Paint paintText, paintValue;
    float cx, cy, cr;

    public TestView(Context context) {
        super(context);
        setBackgroundColor(Color.BLACK);

        paintText = new Paint();
        paintText.setTextSize(36);
        paintText.setColor(Color.YELLOW);

        paintValue = new Paint();
        paintValue.setTextSize(36);
        paintValue.setColor(Color.RED);
    }

    private void init() {
        viewW = getWidth();
        viewH = getHeight();

        cx = viewW / 2;
        cy = viewH / 2;
    }
}
```

```
        cr = 50;

        isInited = true;
    }

    @Override
    protected void onDraw(Canvas canvas) {
        if (!isInited) {
            init();
            canvas.drawText(isSensorEnable ? "重力加速度感應器支援" :
                "裝置不支援",
                viewW / 8, viewH / 8, paintText);
        }

        canvas.drawText((int) sx + ":" + (int) sy + ":" + sz,
            viewW / 8, viewH / 8, paintValue);
        canvas.drawCircle(cx, cy, Math.abs(cr), cr > 0 ?
            paintText: paintValue);
    }
}
```

12



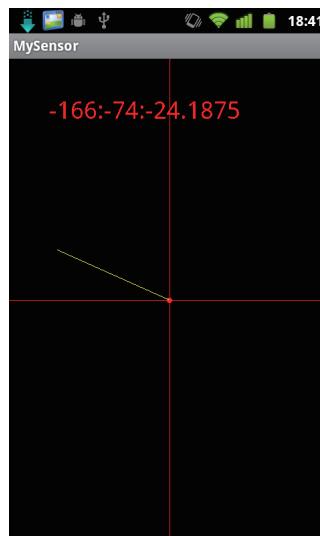
12-4

磁極方向感應器

利用行動裝置所在的空間座標而感應其各自座標的磁場強弱值的感應器。(Sensor.TYPE_MAGNETIC_FIELD)

在各座標軸的數值就是行動裝置目前偵測的磁場磁力的數據資料，磁場磁力最強的就是地球 N 極，也就是向北方向，所以若行動裝置平放的狀態下，表示 Z 座標軸為固定，而以 XY 座標軸所接收的數據資料可以呈現出一個基本的指北針。

以下範例中會將抓到的 XY 座標值放大 4 倍，以便容易處理明顯的數據資料，而數學幾何 Y 座標軸與螢幕的 Y 軸剛好相反（螢幕向上為負方向，向下為正方向），所以乘以 -4。先在螢幕中間會出數學幾何的 XY 座標軸及中心點，而黃色線條從中心點出發所繪製的就是目前磁極的北極。



```
package tw.brad.book.mysensor;

import android.app.Activity;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.view.View;

public class MagneticFieldTest extends Activity {
    private TestView tv;
    private SensorManager smgr;
    private Sensor sensor;
    private boolean isSensorEnable;
    private int rotation;
    private MySensorEventListener listener = null;
    private float x, y, z, sx, sy, sz;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        tv = new TestView(this);
    }
}
```

```
setContentView(tv);
rotation = getWindowManager().getDefaultDisplay().getRotation();

smgr = MainActivity.smgr;
sensor = smgr.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);
// 偵測指定的感應器是否支援
if (sensor != null) {
    isSensorEnable = true;
    listener = new MySensorEventListener();
}
}

@Override
protected void onResume() {
    super.onResume();
    if (isSensorEnable) {
        smgr.registerListener(listener, sensor,
            SensorManager.SENSOR_DELAY_UI);
    }
}

@Override
protected void onPause() {
    super.onPause();
    if (listener != null) {
        smgr.unregisterListener(listener);
    }
}

private class MySensorEventListener implements SensorEventListener {
    private boolean isNotFirst;
    private float zFirst;

    // 精準度改變事件
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }

    // 感應事件
    @Override
    public void onSensorChanged(SensorEvent event) {
        x = event.values[0]*4;
        y = event.values[1]*-4;
        z = event.values[2];
    }
}
```

```
if (rotation == 0) {
    sx = x;
    sy = y;
    sz = z;
} else if (rotation == 3) {
    sx = -1 * y;
    sy = -1 * x;
    sz = z;
}

tv.cr = 30;

tv.postInvalidate();
}

}

private class TestView extends View {
    private int viewW, viewH;
    private boolean isInited;
    private Paint paintText, paintValue;
    float cx, cy, cr;

    public TestView(Context context) {
        super(context);
        setBackgroundColor(Color.BLACK);

        paintText = new Paint();
        paintText.setTextSize(36);
        paintText.setColor(Color.YELLOW);

        paintValue = new Paint();
        paintValue.setTextSize(36);
        paintValue.setColor(Color.RED);
    }

    private void init() {
        viewW = getWidth();
        viewH = getHeight();

        cx = viewW / 2;
        cy = viewH / 2;
        cr = 30;

        isInited = true;
    }
}
```

```

@Override
protected void onDraw(Canvas canvas) {
    if (!isInit) {
        init();
        canvas.drawText(isSensorEnable ?
            " 磁極方位感應器支援 " : " 裝置不支援 ",
            viewW / 8, viewH / 8, paintText);
    }

    canvas.drawText((int) sx + ":" + (int) sy + ":" + sz,
        viewW / 8, viewH / 8, paintValue);

    canvas.drawCircle(cx, cy, 4, paintValue);
    canvas.drawLine(0, cy, viewW, cy, paintValue);
    canvas.drawLine(cx, 0, cx, viewH, paintValue);
    canvas.drawLine(cx, cy, cx +sx, cy + sy, paintText);
}
}
}

```

12



12-5 光線 / 溫度 / 濕度 / 壓力感應器

這部份都是用來測量環境的感應器：

- 光線的感應器（Sensor.TYPE_LIGHT）
- 溫度的感應器（Sensor.TYPE_AMBIENT_TEMPERATURE）
- 相對濕度的感應器（Sensor.TYPE_RELATIVE_HUMIDITY）
- 大氣壓力的感應器（Sensor.TYPE_RELATIVE_PRESSURE）

只有一個 SensorEvent.values[0] 值傳回，值越大光度越亮；值越小越暗。通常可以透過該數據資料調整 App 中的相關屬性。其他的感應器只是所代表的值不同而已，其他邏輯架構都非常類似。

```

package tw.brad.book.mysensor;

import android.app.Activity;
import android.content.Context;

```

```
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.view.View;

public class LightTest extends Activity {
    private TestView tv;
    private SensorManager smgr;
    private Sensor sensor;
    private boolean isSensorEnable;
    private MySensorEventListener listener = null;
    private float x, y, z, sx, sy, sz;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        tv = new TestView(this);
        setContentView(tv);

        smgr = MainActivity.smgr;
        sensor = smgr.getDefaultSensor(Sensor.TYPE_LIGHT);
        // 偵測指定的感應器是否支援
        if (sensor != null) {
            isSensorEnable = true;
            listener = new MySensorEventListener();
        }
    }

    @Override
    protected void onResume() {
        super.onResume();
        if (isSensorEnable) {
            smgr.registerListener(listener, sensor,
                SensorManager.SENSOR_DELAY_UI);
        }
    }

    @Override
    protected void onPause() {
        super.onPause();
        if (listener != null) {
```

```
        smgr.unregisterListener(listener);
    }
}

private class MySensorEventListener implements SensorEventListener {
    // 精準度改變事件
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }

    // 感應事件
    @Override
    public void onSensorChanged(SensorEvent event) {
        x = event.values[0];
        y = event.values[1];
        z = event.values[2];

        sx = x;
        sy = y;
        sz = z;

        tv.cr = z * 10 + 30;

        tv.postInvalidate();
    }
}

private class TestView extends View {
    private int viewW, viewH;
    private boolean isInited;
    private Paint paintText, paintValue;
    float cx, cy, cr;

    public TestView(Context context) {
        super(context);
        setBackgroundColor(Color.BLACK);

        paintText = new Paint();
        paintText.setTextSize(36);
        paintText.setColor(Color.YELLOW);

        paintValue = new Paint();
        paintValue.setTextSize(36);
        paintValue.setColor(Color.RED);
    }
}
```

12

```
}

private void init() {
    viewW = getWidth();
    viewH = getHeight();

    cx = viewW / 2;
    cy = viewH / 2;
    cr = 50;

    isInitiated = true;
}

@Override
protected void onDraw(Canvas canvas) {
    if (!isInitiated) {
        init();
        canvas.drawText(isSensorEnable ?
            "光線感應器支援" : "裝置不支援",
            viewW / 8, viewH / 8, paintText);
    }

    canvas.drawText(sx + " : " + sy + " : " + sz, viewW / 8,
        viewH / 8, paintValue);
    canvas.drawCircle(cx, cy, Math.abs(cr), cr > 0 ?
        paintText: paintValue);
}
}
```

13

Chapter

資源與國際化

- 13-1 提供資源內容
- 13-2 存取資源內容
- 13-3 應用程式執行中的改變
- 13-4 資源內容的區域化



先來看一段 Java 程式最熟悉不過的 Hello,World 看看再說：

```
.....
System.out.println( "Hello, World");
.....
```

對於這的 Java 程式而言，"Hello, World" 就是一種資源，算是文字訊息的資源。但是，無論在任何語系的作業系統下，通通看到的都是 "Hello, World"。如果希望可以在中文環境下看到 "您好，全世界"，甚至於因為登入帳號的使用者為 brad，當執行這隻程式時可以看到 "您好，brad"。這樣的 Hello, World 程式應該會更為廣泛的應用。因此，最簡單的方式就是將準備呈現的資源內容給獨立在程式碼以外的地方，獨立進行維護及管理。

通常在開發應用程式的時候，會將應用程式中所用的的影像或是文字訊息等相關資源獨立維護，而不會直接寫在應用程式碼之中，因為那麼做只會增加日後維護的困擾，除非你覺得這個應用程式專案只需要寫這麼一次就可以為你賺進很多錢。而將應用程式碼以外的資源獨立的處理方式，除了提高日後的維護性之外，還可以進一步的發展成為多國語系的應用程式，使該專案的應用範疇更為廣闊。Android 專案的資源內容通常是放在專案目錄之下的 res/ 子目錄，這個章節就是準備來深入了解資源子目錄。

針對任何類型的資源內容，都會有預設及其他多重的替代性資源可以選擇使用，通常有以下的原則來處理：

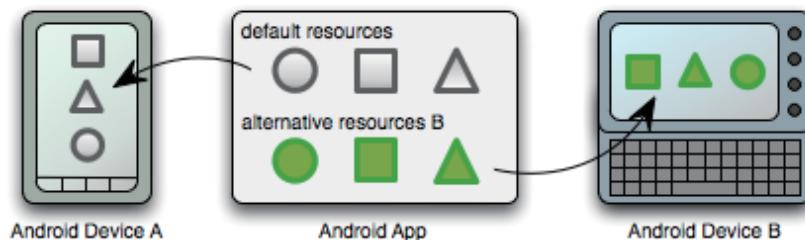
- 預設的資源內容應用在裝置的組態設定或是沒有其他替代性資源可以使用的時機。
- 替代性資源內容會是完整的一組內容，特別用在特定的場合，通常會放在目錄名稱容易辨認的子目錄下。

舉例來說，一般的使用者介面的版面配置內容會放在 res/layout/ 子目錄下，通常預設是針對行動裝置拿直的方式來操作，或是將行動裝置橫向 (landscape) 操作，如果在使用介面上想要另外處理，那可能就會另外建立出一個子目錄 res/layout-land/ 來存放當使用者橫向使用應用程式的資源內容。



13

上圖的 Android 應用程式分別提供給裝置 A 及裝置 B 使用，因為只有處理預設資源，所以無論是裝置 A 的直向操作或是裝置 B 的橫向操作，所看的使用者介面內容都是一樣的，但是呈現的整體比例卻是不同的，所產生的效果就不是很恰當。



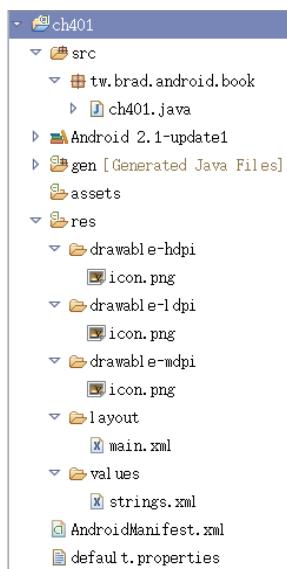
上圖在 Android 應用程式開發架構中，除了預設資源內容的提供之外，還另外提供替代性資源內容給橫向操作的裝置使用。因此，無論是裝置 A 或是裝置 B 都可以看到比例上或是視覺上較為一致的使用者介面。其實，除了視覺上之外的應用，還可以運用在訊息內容的處理。

上述說明只是讓您了解資源使用的重要性。以下將會提供您如何架構完善的資源內容，以及運用替代性資源內容的手法及時機。



13-1 提供資源內容

在 Android 專案架構下，預設建立的專案目錄結構中，就已經將資源內容獨立在 res/ 的子目錄下：



預設資源內容及架構

目前預設的專案建立的同時，產生了 res/drawable-xxxx/，res/layout/ 及 res/values/3 個子目錄。xxxx 表示解析度不同所使用不同的繪圖資源目錄，hdpi 為高解析度；ldpi 為低解析度；mdpi 則是適用於中解析度的裝置。較為完整的內容如下表：

子目錄	存放資源內容格式及型態
res/anim/	<ul style="list-style-type: none"> • 動畫資源內容 • xml 檔案格式 • Tween Animation • Frame Animation

子目錄	存放資源內容格式及型態
res/color/	<ul style="list-style-type: none"> • 定義顏色值 • xml 檔案格式
res/drawable-xxxx/	<ul style="list-style-type: none"> • 影像圖資源 • 點陣圖檔案格式或是 xml 檔案格式 • .png • .9.png • .jpg • .gif • .xml
res/layout/	<ul style="list-style-type: none"> • 使用者介面的版面配置資源 • xml 檔案格式
res/menu/	<ul style="list-style-type: none"> • 選單資源，應用在選項選單 (Options Menu)，本文選單 (Context Menu) 或是子選單 (Sub Menu)。 • xml 檔案格式
res/raw/	<ul style="list-style-type: none"> • 文字檔案資源內容 • 任何沒有經過壓縮處理的原始的文字檔案格式。 • 如果在應用程式中只是單純的讀取文字資料，而並不需要以資源型態管理，則可以放在 assets/ 子目錄之下。其實這就是資源和資產的不同之處，簡單的區分方式就是，資產總是固定的，資源是可以選擇改變的內容。
res/values/	<ul style="list-style-type: none"> • 可能是文字內容，也可能數值資料等等的值。 • xml 檔案格式
res/xml/	<ul style="list-style-type: none"> • 任何形式的 xml 資源內容 • xml 檔案格式



通常不會直接將檔案放進特定的子目錄之下，而會透過特定的操作程序存放。這樣才能夠進行資源管理的目的。影像圖檔就比較可以直接放進 res/drawable-xxxx/ 的子目錄下，放完之後，記得按下 F5 功能鍵。

■ 替代選擇性資源內容 ■

一個完善的 Android 專案應用程式都應該提供除了預設資源以外的其他替代選擇性的資源內容。至少從使用者的語系角度來看，就可以使更多的使用者使用精心所設計出來的應用程式。可能有人會舉這麼一個例子來反駁，就是寫一個卜卦（擲杯）程式，應該只有中文語系的使用者會去使用，你仍然可以針對直向或是橫向操作行動裝置的不同習性來處理不同的版面配置，使得雖然只有中文語系的使用者會使用的應用程式，仍然顧及到不同的操作習性的層面，這就是替代選擇性預設資源的重要性。

處理的程序相當簡單。首先，先建立替代選擇性的資源目錄：

1. 也是在專案的 res/ 子目錄之下，依照命名原則：" 資源項目 - 限定名稱 "，建立出類似像 res/layout-land/ 或是 res/drawable-hdpi/ 的子目錄。而限定名稱還必須更細的區分，則可以繼續描述命名下去，中間以 "-"(dash) 做區隔即可。
2. 替代選擇性資源目錄下的檔案名稱應該與預設資源目錄下的一致。

例如：

```
res/
    drawable/
        icon.png
        welcome.png
        background.png
    drawable-hdpi/
        icon.png
        welcome.png
        background.png
    drawable-ldpi/
        icon.png
        welcome.png
        background.png
    drawable-mdpi/
        icon.png
        welcome.png
        background.png
```

在以上架構中，drawable-xxxx/ 子目錄下的檔案內容及名稱都與 drawable/ 子目錄下的一樣，雖然所呈現的影像檔案內容可能完全不一樣。

以下列出 Android 目前常用的限定名稱：

區分項目	限定名稱範例	說明
MCC 及 MNC	mcc466 mcc466-mnc01 mcc466-mnc92	行動裝置國碼及網路碼 這是定義在 ITU E.212 (Land Mobile Numbering Plan)，用來識別無線電信網路的行動站。特別是 GSM 及 UMTS 網路。相關詳細對照表在附錄中。 <ul style="list-style-type: none"> • mcc466：表示台灣 • mcc460-mnc01：表示台灣遠傳電信 • mcc310-mnc92：表示台灣的中華電信
語言及區域	en en-rUS en-rCA zh-rTW	限定在語系 - 區域的資源目錄，使用兩碼的 ISO 639-1 的語系代碼，及兩碼的 ISO 3166-1-alpha-2 的區域代碼，相關詳細對照表在附錄中。大小寫沒有區分，如果有指定區域代碼，必須前置 "r" 表示區域 (region)。 <ul style="list-style-type: none"> • en 表示英文 • en-rUS 表示英文 - 美國 • en-rCA 表示英文 - 加拿大 • zh-rTW 表示中文 - 台灣
螢幕尺寸	small normal large	small 應用在低解析度的 QVGA 的螢幕上 QVGA (240x320)，2.6"-3.0" 對角線 normal 應用在傳統的中解析度 HVGA 的螢幕上 • WQVGA (240x400)，3.2"-3.5" 對角線 • FWQVGA (240x432)，3.5"-3.8" 對角線 • HVGA (320x480)，3.0"-3.5" 對角線 • WVGA (480x800)，3.3"-4.0" 對角線 • FWVGA (480x854)，3.5"-4.0" 對角線 large 應用在中解析度 VGA 的螢幕上 • WVGA (480x800)，4.8"-5.5" 對角線 • FWVGA (480x854)，5.0"-5.8" 對角線

區分項目	限定名稱範例	說明
螢幕形狀	long notlong	long(寬螢幕) : WQVGA , WVGA , FWVGA notlong : QVGA , HVGA , VGA 這個項目並非是直向或是橫向的操作螢幕，而是螢幕本體。
螢幕操作方向	port land	port 垂直操作方向 (直向) land 水平操作方向 (橫向)
基座模式	car desk	car 車用模式 desk 桌上模式 適用於 Android API Level 8 以上
夜間模式	night notnight	night 夜間模式 notnight 白天模式 適用於 Android API Level 8 以上
螢幕解析度	ldpi mdpi hdpi nodpi	ldpi 低解析度，大約在 120dpi mdpi 中解析度，大約在 160dpi hdpi 高解析度，大約在 240dpi nodpi 應用在點陣資源內容，不限制在特定的解析度
觸控螢幕模式	notouch stylus finger	notouch 沒有觸控螢幕裝置 stylus 使用觸控筆模式 finger 手指的觸控螢幕

注意事項：

- 單一資源目錄的限定名稱可以多重指定，例如 layout-zh-rTW-car 。
- 使用多重限定名稱的規則必須按照上表的順序性使用。例如要建立一個高解析度的垂直影像資源內容，則其名稱為 drawable-port-hdpi ，若寫成 drawable-hdpi-port 就是錯誤的方式。所以上表的順序性是非常重要參考依據。
- 各個多重限定名稱資源目錄不可以使用巢狀結構。
- 也不允許使用特定資源但是多重限定名稱，例如 drawable-mdpi-hdpi 想要同時支援中高解析度的影像資源內容 。



13-2 存取資源內容

一旦完成資源內容的配置及存放，接著下來就是要討論如何存取使用這些資源內容。

13

只要釋放在上一節中的資源內容，也就是存放在專案架構下的 res/ 資目錄資源內容，都可以透過其資源唯一識別碼來進行存取。所有的資源內容都會被定義在 R.java 之中，你不需要手動處理該檔案內容，甚至於說是千萬不要自行改變 R.java 中的內容，開發工具程式 -aapt 會自動配置這些資源內容並賦予一個該專案應用程式的唯一資源識別碼。

一個資源識別碼的組成內容會區分成兩部份：資源型態（resource type）及資源名稱（resource name）。

- 資源型態：例如 string,drawable 或是 layout。
- 資源名稱：例如影像檔案名稱或是自訂 View 元件的名稱。

有兩種方式來進行資源內容的存取

- 程式碼：R. 資源型態 . 資源名稱，例如："R.string.message"
- XML 定義檔：@ 資源型態 / 資源名稱，例如："@string/message"

程式碼中存取資源內容

在程式碼中使用指定資源內容時，只需要指定在 R 類別中的唯一識別碼，通常使用其靜態的成員屬性名稱的方式。假設有一個顯示文字訊息的元件 TextView, 想要呈現文字訊息的資源內容放在 res/values/strings.xml 中的 message。因此，就可以下段程式碼中的 R.string.message 來進行存取：

```
res/values/strings.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">ch401</string>
```

```
<string name="hello">Hello World, ch401!</string>
<string name="message">Hello, Brad</string>
/>
```

```
res/layout/main.xml
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:id="@+id/tv"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        />
</LinearLayout>
```

```
src/tw.brad.android.book/ch401.java
```

```
package tw.brad.android.book;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class ch401 extends Activity {
    private TextView tv = null;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        tv = (TextView) findViewById(R.id.tv);
        tv.setText(R.string.message);
    }
}
```

XML 中存取資源內容

在 xml 檔案中存取資源內容的方法與在程式中不太一樣，相當於說明該資源內容放在哪裡，也就是會有一個前置 "@" 字元，表示 "在" (at) 哪裡的

意思。其格式為 "@ 資源型態 / 資源名稱 "，例如在 main.xml 檔案中定義一個按鈕元件上的文字 android:text="@string/bt_click"，就直接延續上個範例來看：

```
res/values/strings.xml
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">ch401</string>
    <string name="hello">Hello World, ch401!</string>
    <string name="message">Hello, Brad</string>
    <string name="bt_click">按下去吧</string>
</resources>
```

```
res/layout/main.xml
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:id="@+id/tv"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        />
    <Button
        android:id="@+id/bt"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/bt_click"
        />
</LinearLayout>
```



13-3 應用程式執行中的改變

當應用程式已經啟動執行中，可能因為使用者的因素而改變的裝置的模式，最常見的狀況就是原本垂直操作模式改成水平橫向操作，或是從桌上基

座拿到車上基座使用等等執行中的狀態模式改變。無論如何都會被重新啟動 (restart)，也就是呼叫 `onDestroy()` 之後，再度重新啟動 `onCreate()`。這樣的執行過程是自動被呼叫，以便因應新的裝置模式。

常見的處理方式有兩種：(你可以任選一種恰當的方式來實作)

■ 設計一個保留及回存物件 ■

當重新啟動 Activity 的生命周期的時候，原本使用的物件狀態屬性等等資料，必須在還沒有被摧毀之前存放起來，而當再度啟動新的生命周期之後，馬上回存這個物件機制。

1. 重新改寫 `onRetainNonConfigurationInstance()` 方法來傳回想要保留的物件實體。
2. 重新啟動之後呼叫 `getLastNonConfigurationInstance()` 方法行回覆原本被保留的物件。



13-4 資源內容的區域化

一般的開發方式，都是使用特定的語系進行程式設計，往往也因此而只能限定給特定的使用者使用，無法擴大使用者範圍。這個章節就是介紹 Android 上的區域化程式設計方式。

■ 支援的區域國別 ■

以下列出 Android 2.2 版本縮支援的國別語系（排列順序依照 Android 開發者官方網站）

- Chinese, PRC (zh_CN): 中國
- Chinese, Taiwan (zh_TW): 台灣
- Czech (cs_CZ): 捷克

- Dutch, Netherlands (nl_NL): 荷蘭 , 荷蘭文
- Dutch, Belgium (nl_BE): 比利時 , 荷蘭文
- English, US (en_US): 美國 , 英文
- English, Britain (en_GB): 英國 , 英文
- English, Canada (en_CA): 加拿大 , 英文
- English, Australia (en_AU): 澳洲 , 英文
- English, New Zealand (en_NZ): 紐西蘭 , 英文
- English, Singapore(en_SG): 新加坡 , 英文
- French, France (fr_FR): 法國 , 法文
- French, Belgium (fr_BE): 比利時 , 法文
- French, Canada (fr_CA): 加拿大 , 法文
- French, Switzerland (fr_CH): 瑞士 , 法文
- German, Germany (de_DE): 德國 , 德文
- German, Austria (de_AT) : 澳洲 , 德文
- German, Switzerland (de_CH): 瑞士 , 德文
- German, Liechtenstein (de_LI): 列之敦斯登 , 德文
- Italian, Italy (it_IT): 義大利 , 義大利文
- Italian, Switzerland (it_CH): 瑞士 , 義大利文
- Japanese (ja_JP): 日本
- Korean (ko_KR): 韓國
- Polish (pl_PL): 波蘭
- Russian (ru_RU): 俄羅斯
- Spanish (es_ES): 西班牙

對於使用者而言，以上的國別語系可以透過 Android 手機上的「設定→語言與鍵盤→選取地區設定」來做適當的調整。

對於程式設計而言，如果沒有特別的處理語系問題的程式，只需要將相關的資源按照專案預設的資源目錄存放即可。可以在這些資源內容中直接使用中文或是其他語系的文字即可。(別忘記資源目錄就是在 res/)

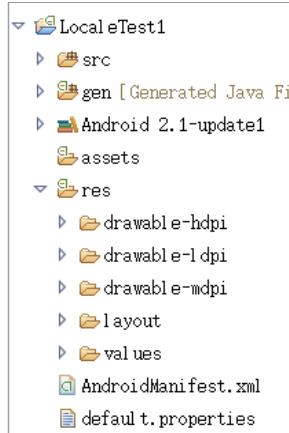
然而，相關語系的資源不只是文字內容而已，還包括了圖像，聲音，影片等等。舉例來看，想處理一個操作介面，當按下按鈕之後，會用語音回報系統當時的狀態，此時就可能會因為使用者設定中文而發生國語語音；而設定英文的使用者將會聽到英語的說明。因此，就先從資源目錄開始來介紹區域化的 Android 開發。

■ 進一步認識專案資源 ■

Android 的專案資源形式可能是一般的文字 (texts)，配置 (layouts)，聲音 (sounds)，圖像 (graphics) 及其他相關的靜態資料項。任何一個 Android 專案中可以包含依或多個資源組 (set)，每組資源內容是用來應用在不同的裝置組態之中，當使用者執行了 Android 應用程式時，Android 會自動選擇一組最佳最適合的資源組來應用，也就是說，Android 是自動選擇，並不會每次執行 Android 應用程式都會詢問使用者，但是也不會對使用者或是開發者造成困擾。舉例而言，Android 設計者一開始就先處理預設的所有資源，接著繼續開發中文語系，但是某些資源卻沒有另外在作處理，形成了預設資源比特定資源（目前範例是中文）還要更多的狀況。而使用者的 Android 裝置上設定了日文語系，則將只會看到預設資源的內容，完全不會看到任何中文資源的內容；而設定為中文的使用者，將可以看到所有的中文資源的內容，而對於預設資源中在中文資源所沒有的內容，則只有該部份會以預設資源內容表現出來。這就所謂 Android 的自動選擇較佳的方式處理。

從上述說明中可以看到非常清楚的一點，就是預設資源的重要性是應該擺在第一優先處理。當然，這並不會對 Android 設計者造成任何困擾或是不便，因為一開始的設計方式就是如此。當一切設計就緒之後，再來好好將想要做特定資源處理的項目做語系上的轉換即可（千萬別想太多，通常是手動處理。當然也可以找到一些轉換工具來使用，或是乾脆自己寫）。

因此，就像之前一樣建立一個新的專案，也看到自動建立的資源目錄，其結構如下：



在 LocalTest1 專案之下，存在了一個 res/ 資源目錄。其子目錄有 drawable-xxxx/ , layout/ 及 values/3 個基本資源目錄。現在就直接開始進行專案開發。首先先處理 layout/main.xml ，以配置出使用者的操作介面。

```
res/layout/main.xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <Button
        android:id="@+id/click"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/click"
        />
    <TextView
        android:id="@+id/tv"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        />
</LinearLayout>
```

因為使用了 @string/click ，並不存在預設的字串資源中，所以接著繼續處理 strings.xml

```
res/values/strings.xml
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">LocaleTest1</string>
    <string name="hello">Hello World, LocaleTest1!</string>
    <string name="click">Click Me</string>
</resources>
```

```
src/tw.brad.android.test/LocaleTest1.java
```

```
package tw.brad.android.test;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class LocaleTest1 extends Activity implements OnClickListener {
    private Button click = null;
    private TextView tv = null;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        click = (Button) findViewById(R.id.click);
        tv = (TextView) findViewById(R.id.tv);

        click.setOnClickListener(this);
    }

    @Override
    public void onClick(View arg0) {
        tv.setText(R.string.mesg);
    }
}
```

設定當 Button 按下去之後，TextView 的文字內容將會被修改成為 R.string.mesg 所代表的訊息。此時因為尚未處理 R.string.mesg，所以將會在該列的左方出現一個紅色的叉。再度回到 res/values/strings.xml 補上該項資源內容。

res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">LocaleTest1</string>
    <string name="hello">Hello World, LocaleTest1!</string>
    <string name="click">Click Me</string>
    <string name="mesg">Hello, Brad</string>
</resources>
```

13

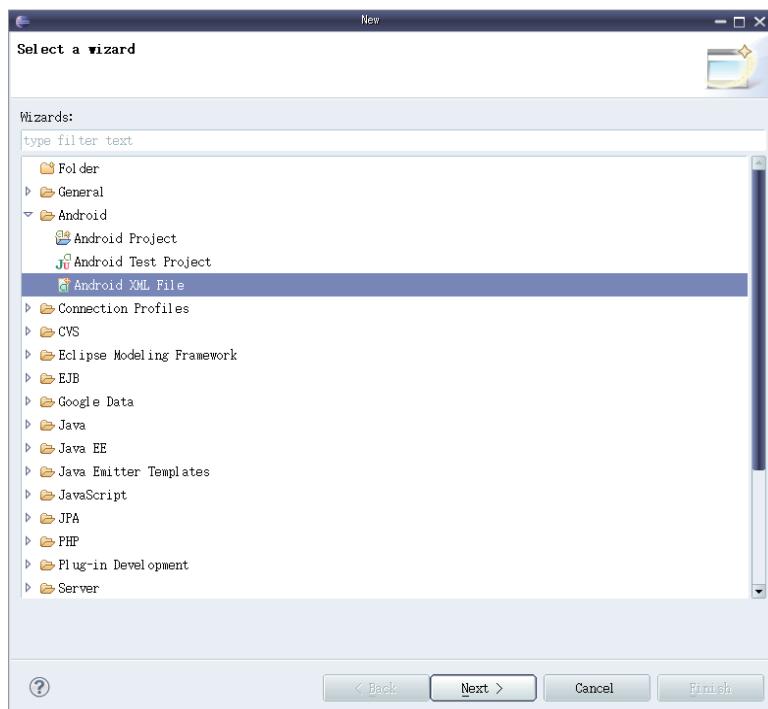
到此為止應該可以正常運作如下：



當按下 Click Me 的 Button 之後：

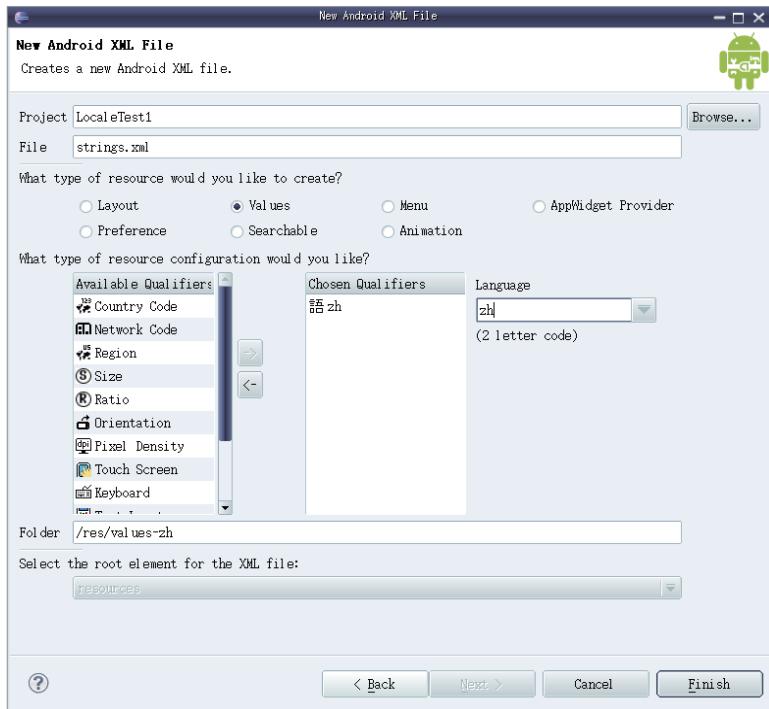
假設這一個龐大的 Android 專案開發已經開發完成。接著下來就是要處理區域化的事情囉 ...

在專案下新增一個 Android XML file：

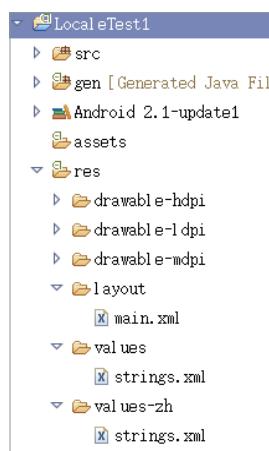


按下 Next 之後，將會看到設定精靈的畫面，只需要填入相關資料即可，
如下：

13



上面的範例是建立出 zh 語系的資源檔，一切正常的狀況下，按下 Finish 之後，會建立出 res/values-zh 的目錄資料夾：



此時的重點就放在 res/values-zh/strings.xml 的檔案內容：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">區域化測試 </string>
    <string name="hello">你好嗎，全世界 </string>
    <string name="click">按下去吧 </string>
</resources>
```

程式都不需要做任何處理及修改，直接執行該專案的應用程式。首先看到圖示的部份：



已經變成區域化測試的圖示文字，接著點按進去。

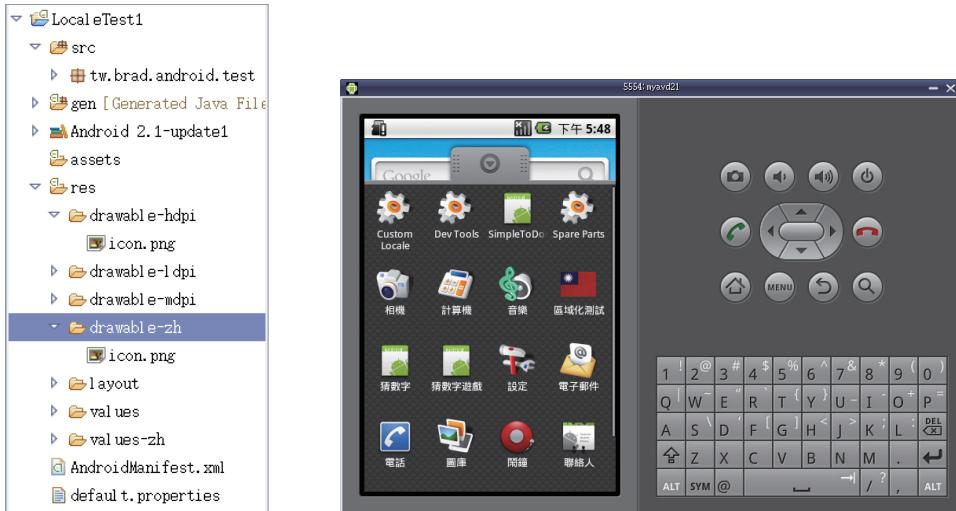


來到這裡都沒有問題，全部內容都如願的變成中文語系，當按下按鈕之後。



這個訊息是設定在 res/values/stringx.xml 中的：`<string name="mesg">Hello, Brad</string>`。但是在 res/values-zh/strings.xml 中卻沒有處理到，因此，Android 自動選擇使用的較佳方式，就是呈現出預設資源中的內容，在這個專案中，還同時看到了兩種不同的資源內容。

再來處理圖示的資源。在選單按下之後，會出現所有應用程式的圖示及標題，這裡的圖示資源預設使用在 res/drawable-xxxx/icon.png 的圖像檔案。而使用區域化程式設計的處理手法類似前段說明的模式。直接手動新增一個 res/ 下的子目錄，名稱為 drawable-zh/，zh 就是中文語系的代碼。直接將所需要呈現的圖示放在該子目錄之下，並且覆蓋掉原來的 icon.png 即可。處理方式相當簡單。



資源型態

資源型態大致上可以區分為以下幾種：

動畫資源（Animation Resources）

- 事先製作完成的動畫資源。
- 二維轉換（twened animation）的動畫資源，放在 res/anim/ 目錄下，使用 R.anim 方式存取。
- 逐一顯示框架（frame by frame animation）的動畫資源，放在 res/drawable/ 目錄架構下，使用 R.drawable 方式存取。

顏色表資源（Color State List Resources）

顏色的對應資源，通常放在 res/color/ 目錄下，以 R.color 方式存取。

繪圖資源（Drawable Resources）

定義影像繪圖的檔案或是 XML，通常放在 res/drawable/ 目錄下，以 R.drawable 方式進行存取。

版面配置資源 (Layout Resources)

應用程式的使用者介面之版面配置資源，通常放在 res/layout/ 目錄下，以 R.layout 方式進行存取。

選單資源 (Menu Resources)

應用程式所需的選單資源，通常放在 res/menu/ 目錄下，以 R.menu 方式進行存取。

13

字串資源 (String Resources)

應用程式中的顯示字串，通常放在 res/values/ 目錄下，以 R.string, R.array 或是 R.plurals 方式進行存取。

樣式資源 (Style Resources)

定義使用者介面元素的呈現樣式，通常放在 res/values/ 目錄下，以 R.style 方式進行存取。

■ 區域化確認檢查 ■

這一節參考 Android 官方網站的確認檢查表，筆者加上自身的經驗提供給讀者在進行區域化程式設計時參考使用。

規劃及設計檢核表

- 確認區域化的策略
- 哪些語系想要支援應用
- 預設語系為何
- 哪些狀況或是時機使用特定的語系或是預設語系
- 確認應用程式中需要區域化的部份
- 針對應用程式中的每個細節做分析，包括文字，影像，聲音，音樂，數字，日期時間及貨幣的表示法。
- 使用者完全看不到的地方就不需要處理區域化

- 注意到應用程式的背景風格一定要搭配特定語系的風格，這樣才有助於銷售出應用程式。
- 程式碼的具體處理
- 使用 R.string 或是 string.xml 方式替代掉直接寫在程式碼中
- 使用 R.drawable 或是 R.layout 方式替代掉直接寫在程式碼中

內容檢核表

- 建立完整的預設資源組
- 確認轉譯的資料內容的正確性
- 使用正確的語系文字內容的格式，尤其是像數字，貨幣及日期時間的表示格式
- 有必要的話，不妨建立一套區域版本來控管

測試及發佈檢核表

- 測試應用程式所支援的所有語系，當然，可以的話，請該語系為母語的人來做測試是最好的方法。
- 測試預設資源組的正確性
- 設定為不支援的語系，是否能正常呈現預設資源的內容
- 測試支援語系中使用到預設資源內容的部份
- 測試預設資源內容在裝置的水平或是垂直的顯示狀況
- 建立一個最終的發佈版本

14

Chapter

系統功能與裝置控制

- 14-1 行動裝置相關辨識
- 14-2 行動電話通話狀態
- 14-3 行動電話用戶相關資料
- 14-4 開發者基本道德





14-1 行動裝置相關辨識

在一般的 App 設計中，很多時候會將行動裝置的特性與一般 PC 產生不同的區隔，就是可以善加運用行動裝置的特性，其中最基本的功能就是電話的使用。

Android.telephony.TelephonyManager 物件實體就是用來獲得使用者行動裝置上與電話相關的資訊。取得該物件實體是透過呼叫 `getSystemService()` 方法，並傳入 `TELEPHONY_SERVICE` 參數即可。

宣告變數：

```
private TelephonyManager tmgr;
```

取得物件實體：

```
tmgr = (TelephonyManager) getSystemService(TELEPHONY_SERVICE);
```

`AndroidManifest.xml` 中需要開啟相關使用權限：`READ_PHONE_STATE`

```
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
```

接著下來就直接呼叫個別資訊相關方法即可。

```
package tw.brad.book.mydeviceinfo;

import android.app.Activity;
import android.os.Bundle;
import android.telephony.TelephonyManager;
import android.widget.TextView;

public class MainActivity extends Activity {
    private TextView info;
    private TelephonyManager tmgr;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

```
info = (TextView)findViewById(R.id.info);
info.setText("");

tmgr = (TelephonyManager) getSystemService(TELEPHONY_SERVICE);
// 手機號碼
String lineNumber = tmgr.getLine1Number();
info.append("手機號碼 :" + lineNumber + "\n");

// IMEI 碼
String IMEI = tmgr.getDeviceId();
info.append("IMEI:" + IMEI + "\n");

// IMSI 碼
String IMSI = tmgr.getSubscriberId();
info.append("tmgr" + IMSI + "\n");

// 漫遊狀態
boolean isRoaming = tmgr.isNetworkRoaming();
info.append("漫遊 :" + (isRoaming?"On":"Off") + "\n");

// 電信網路國別
String Country = tmgr.getNetworkCountryIso();
info.append("電信網路國別 :" + Country + "\n");

// 電信公司
String Operator = tmgr.getNetworkOperator();
info.append("電信公司代號 :" + Operator + "\n");

// 電信公司名稱
String OperatorName = tmgr.getNetworkOperatorName();
info.append("電信公司名稱 :" + OperatorName + "\n");

// 行動網路類型
String[] NetworkTypes =
    {"UNKNOWN", "GPRS", "EDGE", "UMTS",
     "CDMA", "EVDO 0", "EVDO A", "1xRTT",
     "HSDPA", "HSUPA", "HSPA"};
String NetworkType = NetworkTypes[tmgr.getNetworkType()];
info.append("行動網路類型 :" + NetworkType + "\n");

// 行動通訊類型
String[] PhoneTypeArray = {"NONE", "GSM", "CDMA"};
String PhoneType = PhoneTypeArray[tmgr.getPhoneType()];
info.append("行動通訊類型 :" + PhoneType + "\n");

}
```

電話號碼目前以筆者在台灣使用中華電信的狀況下，是空字串資料（不是 null，null 是沒有插 SIM 卡的狀況）

IMEI (International Mobile Equipment Identity number) 為國際行動裝置識別碼，也就是一般所謂的手機序號，算是行動裝置的唯一識別碼，共有 15 碼。前 6 碼為型號核准碼，所以相同 6 碼為相同機型；第 78 碼為裝配碼，也代表產地。通常在行動裝置上面輸入 *#06# 就可以獲得，其資訊內容放在行動裝置的內存記憶體中。

IMSI (International Mobile Subscriber Identity) 為國際行動用戶識別碼，用來識別行動電話用戶的唯一識別碼，通常存放在 SIM 卡中。前 3 碼為行動裝置國碼，台灣 466；第 45 碼表示電信業者，沒插 SIM 卡者沒有資料。

如果 App 有綁定行動裝置的特性，可以使用 IMEI；如果有綁定電信業者或是與電話業務相關，可以使用 IMSI。

實際測試結果：



14-2 行動電話通話狀態

一樣是透過 TelephonyManager 物件實體來處理，繼承 android.telephony.PhoneStateListener 類別，改寫 onCallStateChanged() 方法。

再由 TelephonyManager 物件實體來呼叫 listen() 方法，傳入該物件實體，以及監聽類型。

AndroidManifest.xml 中需要開啟相關使用權限：READ_PHONE_STATE

```
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
```

以下以一個情境來處理，假設想要實做來電電話錄音，通常會透過背景執行的 Service 來處理：

```
package tw.brad.book.mydeviceinfo;

import java.io.IOException;

import android.app.Service;
import android.content.Intent;
import android.media.MediaRecorder;
import android.os.IBinder;
import android.telephony.PhoneStateListener;
import android.telephony.TelephonyManager;
import android.util.Log;

public class MyPhoneService extends Service {
    private TelephonyManager tmgr;
    private MediaRecorder mr;

    @Override
    public IBinder onBind(Intent arg0) {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public void onCreate() {
        super.onCreate();

        tmgr = (TelephonyManager) getSystemService(TELEPHONY_SERVICE);
        tmgr.listen(new MyPhoneStateListener(), PhoneStateListener.
LISTEN_CALL_STATE);
    }

    private class MyPhoneStateListener extends PhoneStateListener {
        @Override
        public void onCallStateChanged(int state, String incomingNumber) {
            switch (state) {
                case TelephonyManager.CALL_STATE_IDLE: // 閑置狀態
                    if (mr != null){
                        // 結束電話錄音
                        mr.stop();
                    }
            }
        }
    }
}
```

14

```
        mr.release();
        mr = null;
    }
    break;
case TelephonyManager.CALL_STATE_OFFHOOK:// 來電拿起話筒
    // 開始電話錄音
    recordPhone();
    break;
case TelephonyManager.CALL_STATE_RINGING:// 響鈴中
    // 顯示對方來電
    Log.i("brad", "RINGING:" + incomingNumber);
}
}

private void recordPhone(){
    mr = new MediaRecorder();
    mr.setAudioSource(MediaRecorder.AudioSource.MIC);
    mr.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
    mr.setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);
    mr.setOutputFile("/mnt/sdcard/brad1.mp4");

    try {
        mr.prepare();
        mr.start();
    } catch (IllegalStateException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

}
```



14-3 行動電話用戶相關資料

在許多內建的 App 中會自動取得使用者相關的設定資料，以方便使用者存取服務，例如 Google Mail/Calendar，或是 Line 會自動取得用戶的聯絡人資料等等。以下簡單介紹處理方式。

■ 使用者帳號 ■

開啟使用權限：

- ACCOUNT_MANAGER
- GET_ACCOUNTS

透過 AccountManager 物件實體來進行：

```
AccountManager mgr = (AccountManager) getSystemService(ACCOUNT_SERVICE);
```

14

接著呼叫 getAccounts() 方法，傳回 Account[] 陣列。每一個 Account[] 陣列元素中的 Account 物件，存在兩個字串屬性 name 和 type。

■ 取得聯絡人姓名 ■

透過 ContentProvider 模式取得。傳遞第二個參數為搜尋關鍵字串，可以是聯絡人姓名或是電話號碼的關鍵字，傳回一個聯絡人姓名字串陣列。

```
// 取得所有聯絡人姓名
static public String[] getContactsName(Context c, String key) {
    // 取得內容解析器
    ContentResolver contentResolver = c.getContentResolver();
    // 設定你要從電話簿取出的欄位
    String name = ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME;
    String number = ContactsContract.CommonDataKinds.Phone.NUMBER;
    String[] projection = new String[]{name, number};

    String selection = name + " like ? or " + number + " like ?";
    String[] selectionArgs = {"%" + key + "%", "%" + key + "%"};

    Cursor cursor;
    if (key != null){
        cursor = contentResolver.query(
            ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
            projection, selection, selectionArgs,
            ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME);
    }else{
        cursor = contentResolver.query(
            ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
            projection, null, null,
```

```

        ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME);
    }

    String[] contactsName = new String[cursor.getCount()];
    String[] contactsPhone = new String[contactsName.length];

    String pretemp = ""; int ri = 0;
    for (int i = 0; i < cursor.getCount(); i++) {
        // 移到指定位置
        cursor.moveToPosition(i);
        if (!pretemp.equals(cursor.getString(1)) &&
            cursor.getString(1).trim().length()>3) {
            // 取得第一個欄位
            String phonenum = cursor.getString(1).replaceAll(" ", "");
            contactsName[ri] = cursor.getString(0) + "\n" + phonenum;
            contactsPhone[ri++] = phonenum;
            pretemp = cursor.getString(1);
        }
    }
    contactsName = Arrays.copyOf(contactsName, ri);
    contactsPhone = Arrays.copyOf(contactsPhone, ri);
    return contactsName;
}

```

|| 使用者的相簿 ||

一般常看到相片管理的 App 呼叫使用。

先取得 ContentResolver 物件實體，

```
ContentResolver resolver = getContentResolver();
```

再來就進行 query() 方法：

```
Cursor c = resolver.query(
    MediaStore.Images.Media.EXTERNAL_CONTENT_URI, null, null, null, null);
```

傳回 Cursor 物件實體之後，就可以開始存取相片欄位。

```
c.getString(c.getColumnIndexOrThrow(MediaStore.Images.Media.DATA));
```

回傳相片檔案在 SD Card 中實際檔案位置。



14-4 開發者基本道德

這章節的應用其實對於使用者而言，應該要善進告知之責。通常從 Google play 市集下載安裝 App，會有一個畫面告知使用者該 App 會開啟的相關權限有哪些。但是大多數的使用者都是直接按下同意，急著看看是否安裝完畢並馬上使用。如果以這兩小節所開啟的權限，再加上一個網際網路，App 開發者可以非常輕鬆的將用戶的電話資訊／來電號碼／秘密錄音等資料傳送到遠端伺服器。

14

水能載舟亦能覆舟，這類有關於使用者隱私相關資料，對於 App 開發者而言，可以用在對於使用者方便好用，卻也能用在不是正當的狀態下。而這類的技術層面一點都不高，無奈目前筆者所接觸到的使用者，99.99% 的比例都不仔細看看所下載的 App 所開啟的權限，反正就是要玩要用。想一想，一個普通的遊戲為何要讀取使用者帳號／電話狀態／通話紀錄...？最可怕得就是網際網路權限，可以將上述資料在背景中傳送出去。

相信大多數 App 都是善良的，但是廣告商的 API 要開啟網際網路權限才能輪播廣告業主的廣告內容，才能統計分析有效的行銷等等。

既然如此，何不大家一起推廣自由軟體／開放原始碼吧。

MEMO

15

Chapter

實際專案開發

- 15-1 彈指磚塊王 (Bricks Fighter)
- 15-2 掏金沙 (Lode Runner)
- 15-3 炸彈超人 (Bomb King)
- 15-4 其他應用程式開發專案



這只是筆者在資策會授課的開發範例專案，為了鼓勵學員參加比賽並分享知相關參賽經驗的實際案例，這是中華電信 2012 電信創新應用大賽中的行動應用／一般組優勝，筆者將此原始碼開放，並進行開發說明，原始碼公開的目的是希望藉此提高讀者學習動力。

<http://innovation.hinet.net/telsoft/history/2012.html>



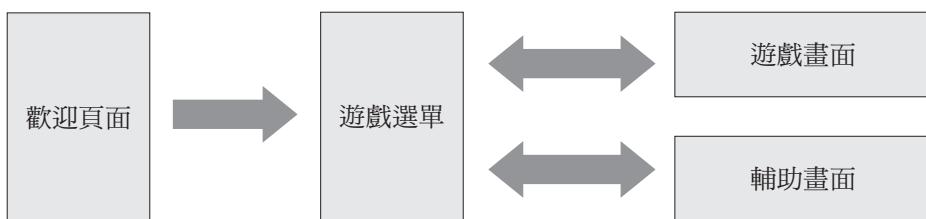
15-1 彈指磚塊王 (Bricks Fighter)

打磚塊遊戲已經在地球上存活超過 30 年以上，這個歷久不衰的經典遊戲移植到觸控螢幕之後，如果只是傳統的反彈棒的操控而已，那就只是一般的打磚塊遊戲。這款遊戲利用觸控螢幕的特性，可以瞬間畫出一條即時反彈棒，玩家可以依照反彈的角度來自由畫出反彈棒以反彈。而遊戲的目的並非將所有的磚塊都打光，是努力取得一開始出現在磚塊最上方的寶物，依照關卡設計而有不同數量的寶物，玩家的策略在於取得所有的寶物。針對遊戲的刺激性，玩家可以選擇計時賽，各個關卡的計時不同。

因應上述的動機，將會應用以下技巧：

- 週期計時的執行緒
- 自訂 View 類別成為遊戲的主畫面
- 觸控螢幕的手勢偵測
- 讀取專案中的關卡資料
- 動畫效果

App 簡易架構



15

歡迎頁面

版面配置：res/layout/welcome.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/welcome"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/bg0" >

    <ImageView
        android:id="@+id/logo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:src="@drawable/welcome" />

</RelativeLayout>
```

而為了增加效果，使用簡單的漸進／增大／轉動畫處理：res/anim/a1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set android:shareInterpolator="false"
      xmlns:android="http://schemas.android.com/apk/res/android"
      >
    <alpha
        android:fromAlpha="0.1"
        android:toAlpha="1.0"
        android:duration="2000"
        android:interpolator="@android:anim/accelerate_interpolator"
      />
    <scale
        android:fromXScale="0.0"
        android:fromYScale="0.0"
        android:toXScale="1.0"
        android:toYScale="1.0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="2000"
      />
    <rotate
        android:fromDegrees="359"
        android:toDegrees="0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="2000"
      />
  </set>
```

開始編寫 Activity：

```
package tw.brad.android.games.BricksFighter;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;
import android.widget.RelativeLayout;

public class Welcome extends Activity {
    private ImageView logo;
```

```
private RelativeLayout welcome;
private Animation a1;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.welcome);

    logo = (ImageView) findViewById(R.id.logo);
    welcome = (RelativeLayout) findViewById(R.id.welcome);

    a1 = AnimationUtils.loadAnimation(this, R.anim.a1);
    logo.startAnimation(a1);

    welcome.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            logo.clearAnimation();
            Intent it = new Intent(Welcome.this, MainMenu.class);
            startActivity(it);
            Welcome.this.finish();
        }
    });
    logo.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            logo.clearAnimation();
            Intent it = new Intent(Welcome.this, MainMenu.class);
            startActivity(it);
            Welcome.this.finish();
        }
    });
}

@Override
public void finish() {
    super.finish();
}
```

15

玩家不會想看這個畫面太久，但是開發者希望現家可停留久一點以留下印象，所以取中權衡後，當出現歡迎畫面後，使用者自行觸摸螢幕離開，進入遊戲關卡選單。

遊戲關卡選單

版面配置：res/layout/mainmenu.xml

採用 RelativeLayout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/mainmenu"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/brickwall"
    >

    <LinearLayout
        android:id="@+id/mtitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        >
        <ImageView
            android:id="@+id/menu_title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/mlogo"
            />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/func"
        android:layout_alignParentBottom="true"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:gravity="center_horizontal|bottom"
        android:background="@drawable/menu_bottom"
        >
        <ImageView
            android:id="@+id/sound"
            android:src="@drawable/sound_on"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            />
        <ImageView
```

```
        android:id="@+id/limit"
        android:src="@drawable/time_nolimit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
    />
<ImageView
    android:id="@+id/help"
    android:src="@drawable/help"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
/>

<ImageView
    android:id="@+id/exit"
    android:src="@drawable/exit"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
/>
</LinearLayout>

<GridView
    android:id="@+id/gv"
    android:layout_below="@+id/mttitle"
    android:layout_above="@+id/func"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:numColumns="4"
    android:horizontalSpacing="32dp"
    android:layout_marginBottom="12dp"
    android:layout_marginTop="2dp"
/>
</RelativeLayout>
```

15

搭配個別影像檔案後，呈現風格如下：



上方為標題列，下方為功能列，中間全部空間為 GridView 的網格配置，其他部份就由 Activity 中處理。

以下就幾個重點解說。

透過 SharedPreferences 取得使用者遊戲相關設定及狀態資訊。

```
sp = getSharedPreferences("gamedata", MODE_PRIVATE);
speditor = sp.edit();

isSound = sp.getBoolean("sound", false); // 是否播放音效
isLimit = sp.getBoolean("limit", false); // 是否限時模式
user_level = sp.getInt("user_level", 0); // 目前最高等級
```

使用者按下下方 Sound 按鈕的切換狀態：(限時模式也是相同處理手法)

```
sound = (ImageView) findViewById(R.id.sound);
sound.setImageResource(resSound[isSound ? 1 : 0]);
sound.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        isSound = isSound ? false : true;
        sound.setImageResource(resSound[isSound ? 1 : 0]);
        speditor.putBoolean("sound", isSound);
        speditor.commit();
    }
});
```

將遊戲選關處理搭配目前玩家的最高等級處理方式：

```

private void reloadMenu() {
    String[] from = { "level_img", "level_txt" };
    int[] to = { R.id.level_img, R.id.level_txt };

    user_level = sp.getInt("user_level", 0);

    ArrayList<HashMap<String, Object>> item = new ArrayList<HashMap
    <String, Object>>();

    for (int i = 0; i <= 27; i++) {
        // 以下正式發佈用
        if (i <= user_level) {
            HashMap<String, Object> map1 = new HashMap<String, Object>();
            map1.put("level_img", R.drawable.level_ed);
            map1.put("level_txt", i);
            item.add(map1);
        } else {
            HashMap<String, Object> map1 = new HashMap<String, Object>();
            map1.put("level_img", R.drawable.level_yet);
            map1.put("level_txt", i);
            item.add(map1);
        }

        // 以下測試用
        // HashMap<String, Object> map1 = new HashMap<String, Object>();
        // map1.put("level_img", R.drawable.level_ed);
        // map1.put("level_txt", i);
        // item.add(map1);
    }
    // 以下設定使用者選按特定遊戲關卡
    gv.setOnItemClickListener(new OnItemClickListener() {
        public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                               long arg3) {
            // 以下正式發佈用
            if (arg2 <= user_level) {
                toGameView(arg2);
            }

            // 以下測試用
            // toGameView(arg2);
        }
    });
}

gv.setSelection(user_level > 3 ? user_level - 4 : 0);
}

```

15

其中還分成正式發佈和測試用，因為筆者可能會開發到很難的關卡，還要和一般玩家一樣一路廝殺過關就太累了，所以測試用的部份可以直接玩任何一關。

上述的 `reloadMenu()` 是在該 `Activity` 一開始的時候被呼叫，以及當玩家從遊戲畫面回到選單的時候，也將會被呼叫，可能的差異在於是否已經玩到後面進階的關卡。而最後呼叫的 `setSelection()` 就是避免每次都是從第一關開始呈現，而玩家已經玩到後面關卡，所以直接定位選單的一開始處在玩家最新關卡列。

|| 遊戲主頁 ||

採用自訂 `View` 的方式來處理，主控的 `Activity` 為 `Main.java`，負責處理：

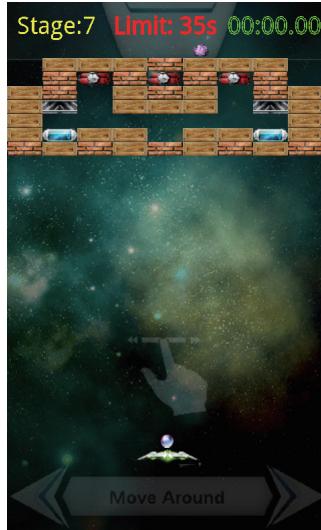
- 載入遊戲的初始畫面
- 遊戲每一回合的結束，並呈現詢問對話框
- 監聽遊戲狀態

而遊戲畫面為 `PlayView.java` 中的自訂 `View` 類別，負責遊戲過程中的所有程序。

因此先來檢視最簡單的版面配置：(`res/layout/main.xml`)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    >
    <tw.brad.android.games.BricksFighter.PlayView
        android:id="@+id/playview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        />
</LinearLayout>
```

這樣就可以開始開發 `PlayView.java` 的遊戲主頁了。



15

主要有以下幾個部份要進行繪製

- 背景（特定數個影像隨機出現，增加不同的感覺）
- 上方遊戲資訊列
- 磚塊關卡
- 中間為手勢觸控區，玩家可在此區域以手勢畫出自訂反彈棒
- 反彈控制器（傳統打磚塊的反彈棒）
- 最下列為左右移動反彈控制器

載入遊戲關卡，使用最簡單的文字資料解析。

事先編寫關卡資料放在 raw/xxx，假設上圖中的第七關為 raw/level07，

```
35
0,6,6,6,5,6,6,6,0
0,5,7,5,7,5,7,5,0
6,5,0,6,5,6,0,5,6
6,9,0,6,6,6,0,9,6
6,0,0,0,0,0,0,0,6
6,8,6,6,0,6,6,8,6
5,6,5,6,5,6,5,6,5
```

第一列的 35 表示如果玩家玩限時模式，該關卡設定為 35 秒。

之後的每一列表示磚塊的一列，自訂如下

- 0：空
- 6：一般磚塊（打擊一次就破掉）
- 5：堅硬磚塊（打擊三次才全破）
- 7：炸彈（可以擊破周圍四周全部磚塊）
- 8：射擊寶物（可以在一定時間內連續發射子彈）
- 9：鋼磚（打不破）

定義全部關卡檔案資源：

```
private static final int[] level = {
    R.raw.level00, R.raw.level01, R.raw.level02,
    R.raw.level03, R.raw.level04, R.raw.level05,
    R.raw.level06, R.raw.level07, R.raw.level08,
    R.raw.level09, R.raw.level10, R.raw.level11,
    R.raw.level12, R.raw.level13, R.raw.level14,
    R.raw.level15, R.raw.level16, R.raw.level17,
    R.raw.level18, R.raw.level19, R.raw.level120,
    R.raw.level21, R.raw.level22, R.raw.level23,
    R.raw.level24, R.raw.level25, R.raw.level26,
    R.raw.level27, R.raw.level28, R.raw.level29,
};
```

載入並同時解析關卡資料

```
private void loadData(){
    // 載入指定 level 的地圖資料
    try {
        noAppleC.clear();
        BufferedReader reader;
        reader =new BufferedReader(
            new InputStreamReader(res.openRawResource(level[play_level])));
        total_bricks = 0; mapLines = 0;
        map = new int[16][9];
        String temp1;
        String[] temp3;
        int i = 0, j = 0, v;

        temp1 = reader.readLine();
        limitSeconds = Integer.parseInt(temp1);
        while ((temp1 = reader.readLine())!=null){
```

```

temp3 = templ.split(",");
for(String temp4 : temp3){
    v = Integer.parseInt(temp4.trim());
    if (v == 9) noAppleC.add(j);
        // 有鋼板的 column 不放寶物
    map[i][j++] = v;
    if (v>0 && v < 10) total_bricks++;
}
mapLines = i;
i++; j=0;
}
reader.close();
} catch (IOException e) {
}

}

```

15

有了資料就可以開始處理影像圖檔，在此筆者並不想要為不同尺寸螢幕而煩惱，所以影像資源會先配合玩家的行動裝置的螢幕而進行比例計算而縮放。

先自訂一個 init() 方法出來，當開啟遊戲畫面只需要執行一次的處理即可。

```

private void init(){
    viewW = getWidth(); viewH = getHeight();

    matrix.reset();
    barW = viewW/5; barH = viewH/32;
    matrix.postScale((float)barW/bar.getWidth(), (float)barH/bar.getHeight());
    bar = Bitmap.createBitmap(bar, 0, 0, bar.getWidth(), bar.
        getHeight(), matrix, false);

    matrix.reset();
    matrix.postScale((float)barW/barfire.getWidth(), (float)barH/barfire.
        getHeight());
    barfire = Bitmap.createBitmap(barfire, 0, 0, barfire.getWidth(),
        barfire.getHeight(), matrix, false);

    bottomY = viewH - viewH/6 + barH;

    matrix.reset();
    matrix.postScale((float)viewW/bgW, (float)viewH/bgH);
    bg = Bitmap.createBitmap(bg, 0, 0, bgW, bgH, matrix, false);

    ...
}

```

```

        matrix.reset();
        matrix.postScale((float)viewW/top.getWidth(), (float)(viewH-
            bottomY)/top.getHeight());
        top = Bitmap.createBitmap(top, 0, 0, top.getWidth(), top.
            getHeight(), matrix, false);

        matrix.reset();
        ballW = viewW/24; ballH = ballW;
        matrix.postScale((float)ballW/ball.getWidth(), (float)ballH/ball.
            getHeight());
        ball = Bitmap.createBitmap(ball, 0, 0, ball.getWidth(), ball.
            getHeight(), matrix, false);

        matrix.reset();
        appleW = barW/3; appleH = barH; apple_dy = appleH/4;
        matrix.postScale((float)appleW/apple.getWidth(), (float)appleH/
            apple.getHeight());
        apple = Bitmap.createBitmap(apple, 0, 0, apple.getWidth(), apple.
            getHeight(), matrix, false);
        ...
    }
}

```

(詳細如光碟所附)

在進行縮放之前，先呼叫 `getWidth()` 與 `getHeight()` 方法取得玩家當前行動裝置的實際寬高。

`Matrix` 物件變數 `matrix` 是重要進行縮放的物件，從頭到尾只用一個物件，所以要進行不同影像縮放之前要先呼叫 `reset()` 方法。而影像尺存完全依照螢幕寬高的比例來處理，所以不需要為上千種不同的螢幕來煩惱。

準備就緒後，就來處理每次需要重新繪製的方法。

```

@Override
public void draw(Canvas canvas) {
    if (!isInit)
        init();

    canvas.drawBitmap(bg, 0, 0, p);
    canvas.drawBitmap(arrow_bg, 0, bottomY, p);
    canvas.drawBitmap(top, 0, 0, p2);

    if (!isStart) {
}

```

```
        canvas.drawBitmap(hand, (viewW - ctW) / 2, barY - ctH - barH
                           * 2, p);
    }
// 限時模式的倒數計時
if (isLimit) {
    switch (limitSeconds - intTimer / 100) {
    case -1:
        stopRound();
        return;
    case 1:
        canvas.drawBitmap(c01, (viewW - ctW) / 2,
                           barY - ctH - barH * 2, p);
        break;
    case 2:
        canvas.drawBitmap(c02, (viewW - ctW) / 2,
                           barY - ctH - barH * 2, p);
        break;
    case 3:
        canvas.drawBitmap(c03, (viewW - ctW) / 2,
                           barY - ctH - barH * 2, p);
        break;
    case 4:
        canvas.drawBitmap(c04, (viewW - ctW) / 2,
                           barY - ctH - barH * 2, p);
        break;
    case 5:
        canvas.drawBitmap(c05, (viewW - ctW) / 2,
                           barY - ctH - barH * 2, p);
        break;
    case 6:
        canvas.drawBitmap(c06, (viewW - ctW) / 2,
                           barY - ctH - barH * 2, p);
        break;
    case 7:
        canvas.drawBitmap(c07, (viewW - ctW) / 2,
                           barY - ctH - barH * 2, p);
        break;
    case 8:
        canvas.drawBitmap(c08, (viewW - ctW) / 2,
                           barY - ctH - barH * 2, p);
        break;
    case 9:
        canvas.drawBitmap(c09, (viewW - ctW) / 2,
                           barY - ctH - barH * 2, p);
        break;
    case 10:
        canvas.drawBitmap(c10, (viewW - ctW) / 2,
                           barY - ctH - barH * 2, p);
```

```
        break;
    }
}

float ballCX = ballX + ballW / 2, ballCY = ballY + ballH / 2;
float ballX2 = ballX + ballW, ballY2 = ballY + ballH; // 繪製目前的磚塊區
for (int r = 0; r < map.length; r++) {
    for (int c = 0; c < map[r].length; c++) {
        if (map[r][c] > 0) {
            float bx = c * (brickW), by = (r + brickStartR)
                * (brickH + 1);
            canvas.drawBitmap(bks[map[r][c]], bx, by, null);
        } else if (map[r][c] < 0) {
            float bx = c * (brickW), by = (r + brickStartR)
                * (brickH + 1);
            switch (map[r][c]) {
case -7:
            canvas.drawBitmap(bomb1, bx, by, null);
            map[r][c]++;
            break;
case -6:
            canvas.drawBitmap(bomb2, bx, by, null);
            map[r][c]++;
            break;
case -5:
            canvas.drawBitmap(bomb3, bx, by, null);
            map[r][c]++;
            break;
case -4:
            canvas.drawBitmap(bomb4, bx, by, null);
            map[r][c]++;
            break;
case -3:
            canvas.drawBitmap(bomb3, bx, by, null);
            map[r][c]++;
            break;
case -2:
            canvas.drawBitmap(bomb2, bx, by, null);
            map[r][c]++;
            break;
case -1:
            canvas.drawBitmap(bomb1, bx, by, null);
            map[r][c]++;
            break;
        }
    }
}
}
```

```
for (int i = 0; i < appleTask.length; i++) {
    if (appleTask[i] != null) {
        canvas.drawBitmap(apple, appleTask[i].getX(),
                          appleTask[i].getY(), null);
    }
}
// 發射子彈
synchronized (bullets) {
    for (Bullet bt : bullets) {
        if (!bt.isHit) {
            canvas.drawBitmap(bullet, bt.bx, bt.by, null);
        }
    }
}

canvas.drawBitmap(ball, ballX, ballY, null);
canvas.drawBitmap(isFireMode ? barfire : bar, barX, barY, null);

// 繪製玩家的手勢反彈棒
if (isShowDrawLine
    && System.currentTimeMillis() - startShowDrawTime <= displayTime) {
    canvas.drawBitmap(drawbar, drawbarX, drawbarY, null);
} else {
    isShowDrawLine = false;
}

// 繪製上方資訊列
canvas.drawText(strTimer, viewW - 100, 32, timerPaint);
canvas.drawText(res.getString(R.string.stage) + play_level, 10, 32,
               stagePaint);
if (isLimit) {
    canvas.drawText(res.getString(R.string.time_limit) + " "
                  + limitSeconds + "s", viewW / 3, 32, limitPaint);
}

// 過關或是失敗的圖案呈現
if (isStopRound) {
    if (isWin) {
        canvas.drawBitmap(success, 0, 0, null);
    } else {
        canvas.drawBitmap(fail, 0, 0, null);
    }
}
```

處理繪製的大原則就是不要有太多或是太複雜的處理程序，盡量單純到只是作繪製的工作而已。相關計算控制變化都應該在各自的物件執行緒中處理，務必使時間週期繪製工作簡單為原則。

最後處理各自物件的執行緒，其實也只有兩種物件：

- 反彈球
- 子彈

而碰撞偵測的原則是：

- 移動物件碰撞靜止物件，是由移動物件進行偵測
- 移動速度快與移動速度慢碰撞，通常是由移動速度快者進行判斷
- 移動速度一樣者，就任選一個即可

最後，呈現成功過關畫面給您：





15-2 掏金沙 (Lode Runner)

開發動機

Lode Runner (中文名稱為掏金沙) 是一款相當經典的遊戲，筆者首次是在 1985 年的 AppleII 上玩的 PC 遊戲，當年的架構畫面非常的簡單。遊戲的主軸是一位主角，在每一個關卡中找尋寶物闖關，但是會有敵人前來阻撓，遊戲主角可以運用智慧避開，或是挖開腳下的磚塊使敵人掉下去，暫時躲過一劫。下圖為當年 AppleII 上的螢幕截圖：http://en.wikipedia.org/wiki/File:Lode_Runner.jpg

15



因為其遊戲主題中包含了智慧及耐心的遊戲特性，加上關卡的豐富變化性，也曾在紅白機上改編為掏金沙。

原遊戲關卡中主要有磚塊、硬磚、樓梯和單槓構成，而寶物出現的位置於每關卡各有不同的設計，敵人的初始位置與數量也會隨關卡而有不同的規劃。筆者對於這款遊戲非常懷念，但是在 2010 年初努力搜尋各個 Market，就是無法找到原汁原味的 Android Game，而其他類似的遊戲，不是太過單調，關卡設計不多，操作不易，不然就是以 3D 呈現而失去了原來的智慧性。因此決定自己下手來設計，盡量呈現出原來的精神感覺，關卡數至少

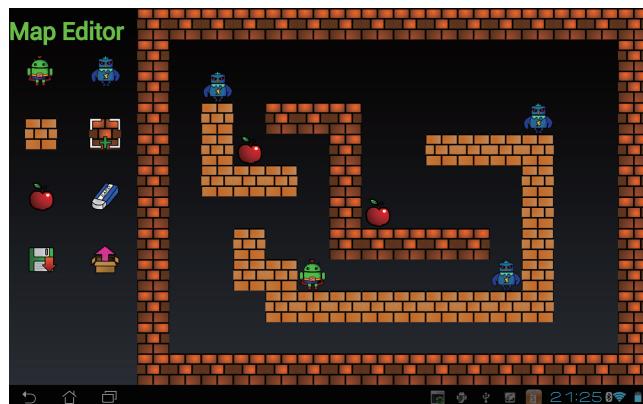
150 關以上，而操作模式務必發揮觸控螢幕的特性。終於花了約兩個月的時間完成這個專案。以下就以該專案開放原始碼方式，為讀者簡單介紹。

■ 著手規劃 ■



這是第 147 關的初始畫面。下方的 Android 是遊戲主角，藍色的壞鳥（Blue Bad Bird, 簡稱：BB Bird）就是敵人，而蘋果（或是可能出現芒果）就是寶物，螢幕的右上方就是當吃完所有蘋果後，要努力抵達的勝利徽章，灰色磚塊的顯示，表示當 Android 停下來的時候，可以挖掘的磚塊。

為了達到持續開發不同的關卡，還特地開發出關卡編輯器，也是利用觸控螢幕的特性，用手指就可以輕易設計出不同特色的關卡。最後還將自編關卡的功能提供給玩家使用，享受自導自演的樂趣。



遊戲架構

大致上與上一個章節的 Bricks Fighter 類似，所以直接以遊戲畫面說明較為易懂。

歡迎畫面

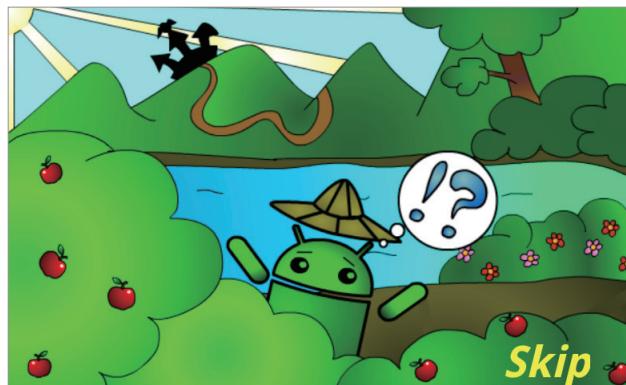
用幾張圖敘述遊戲的故事性。

在 Android 農莊裡，種了許多蘋果樹和芒果樹，Android 農夫正在開心的採收，而 BB Bird 城堡的人卻在對岸覬覦著這些豐收的水果。

15



有一天早上，Android 農夫發現蘋果和芒果少了許多。



於是決定在晚上偷偷觀察，終於知道是隔壁 BB Bird 城堡的壞蛋來偷的。



決定派出 Android Runner 出動前往 BB Bird 城堡（就是各式各樣的關卡）將水果拿回來。



這些圖片的處理，事實上只是一個 Activity 中的一個 ImageView，當使用者觸摸螢幕後切換到下一張影像檔案，如果是觸摸「Skip」文字字樣，則離開 Activity，進入到關卡選單。

```
res/layout/welcome.xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
```

```

    android:background="@drawable/bg_welcome"
    >
    <ImageView
        android:id="@+id/welcome_img"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:src="@drawable/logo"
    />
    <TextView
        android:id="@+id/welcome_skip"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/welcome_skip"
        android:textSize="36dp"
        android:textStyle="bold|italic"
        android:textColor="#ffffffff00"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_marginRight="42dp"
    />
</RelativeLayout>

```

15

而在主程式中的處理就非常簡單。

Welcome.java

```

package tw.brad.apps.LodeRunner;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.WindowManager;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;
import android.widget.TextView;

public class Welcome extends Activity {
    private ImageView welcome_img;
    private TextView welcome_skip;
    private int nowImgIndex = 0;
    private int[] imgs = {
        R.drawable.story1, R.drawable.story2,
        R.drawable.story3, R.drawable.story4
    }
}

```

```
};

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // 防止進入休眠
    getWindow().setFlags(WindowManager.LayoutParams.FLAG_KEEP_
        SCREEN_ON,
    WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

    setContentView(R.layout.welcome);

    welcome_img = (ImageView)findViewById(R.id.welcome_img);
    Animation an = AnimationUtils.loadAnimation(this, R.anim.myanim);
    welcome_img.startAnimation(an);

    welcome_skip = (TextView)findViewById(R.id.welcome_skip);
    welcome_skip.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            // 進入載入資料畫面
            Intent intent =
                new Intent(Welcome.this, MainMenu.class);
            startActivity(intent);
            Welcome.this.finish();
        }
    });
}

welcome_img.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View arg0) {
        if (nowImgIndex == 4){
            // 進入載入選單資料畫面
            Intent intent =
                new Intent(Welcome.this, MainMenu.class);
            startActivity(intent);
            Welcome.this.finish();
        }else {
            welcome_img.setImageResource(imgs[nowImgIndex++]);
        }
    }
});
```

}

關卡選單



15

關卡選單的處理手法與 Bricks Fighter 可以說是完全相同，因此詳細的原始碼請參考書中所附的光碟。

遊戲畫面

完全是以利用自訂 View 的開發方式處理。重點如下：

- 整個螢幕不分尺寸大小，規劃成 16x24 的基本單位來處理
- 關卡資源檔案以 int[] 列存放，俟需要才作載入，因此不需過久的載入等候時間
- 依照玩家自行設定，可以調整遊戲執行快慢，重點是以智慧過關，不是速度快慢過關

一開始處理所有影像元素，自行開發一個自訂方法處理：

```
private Bitmap fitBitmap(int resId){
    Bitmap temp = BitmapFactory.decodeResource(res, resId);
    int bmp_bw = temp.getWidth();
    int bmp_bh = temp.getHeight();
    matrix.reset();
    matrix.postScale((wp)/bmp_bw, (hp)/bmp_bh);
    return Bitmap.createBitmap(temp, 0, 0, bmp_bw, bmp_bh, matrix, true);
}
```

所以每個影像元素都是一個基本單位大小。

當使用過關而進到下一關，差異在於關卡資料讀取，以及各項資料的初始化處理：

```
private void restartGame(int n){
    if (isSound) {
        Music.stop(this);
        if (bgmusic != null){
            bgmusic.setLooping(true);
            bgmusic.setScreenOnWhilePlaying(false);
            bgmusic.setVolume(0.2f,0.2f);
            bgmusic.start();
        }
    }
    System.gc();

    try{
        gview = new GameView(this, null, loadGameData(n));
        // 處理遊戲背景
        gview.setBackgroundResource(R.drawable.bg);
        setContentView(gview);

    }catch(Exception e){
    }
}
```

載入關卡方法與 Bricks Fighter 類似，差異只是遊戲元素的定義而已。

```
private int loadGameData(int n) throws Exception {
    int eCount = 0;

    // 載入指定 level 的地圖資料
    BufferedReader reader;
    if (n == -1) {
        reader = new BufferedReader(new FileReader(play_userfile));
    } else {
        reader = new BufferedReader(new InputStreamReader(
            res.openRawResource(leveldata[n])));
    }

    map = new int[16][24];
    String temp1;
    String[] temp3;
```

```

int i = 0, j = 0, v;
while ((temp1 = reader.readLine()) != null) {
    temp3 = temp1.split(",");
    for (String temp4 : temp3) {
        v = Integer.parseInt(temp4);
        map[i][j++] = v;
        if (v == 7)
            eCount++;
    }
    i++;
    j = 0;
}
reader.close();
return eCount;
}

```

15

傳回值為這關的寶物個數，以利計算是否已經拿完該關的出現的所有寶物，才會出現隱形樓梯抵達勝利徽章。

而關卡檔案類似如下內容：res/raw/level037

```

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-9
0,0,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,-1,1
2,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0
2,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0
2,0,1,2,9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,9,2,1,0
2,0,0,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,1,0
2,0,1,9,0,1,1,1,1,1,9,0,1,0,9,1,1,1,1,1,1,9,1,0
2,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,1,0
2,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,1,0
2,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,1,1,1,1,1,1,2,1,0
2,0,1,1,1,1,1,1,1,1,0,0,0,0,0,2,1,1,1,1,1,2,1,0
2,0,1,1,1,1,1,1,1,1,9,0,0,0,0,0,2,1,1,1,1,1,2,1,0
2,0,0,0,0,0,0,0,0,1,1,1,1,1,2,3,3,3,3,2,0,0
2,0,0,0,0,0,0,0,1,1,1,1,0,2,0,0,0,0,0,0,0,0,0,0,0
2,0,0,0,0,8,0,0,0,0,0,1,1,1,1,0,0,0,0,7,0,0,0,0,0,0
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1

```

表現出如下圖：



|| 關卡地圖 ||

一開始真的是以手工編寫數字資料的關卡，當寫到第十關之後，已經無法再繼續下去了，而當初立下志願是至少 150 關，於是馬上著手撰寫關卡編輯器。

沒有 XML 的版面處理，直接開發在 Activity 中，整個關卡編輯器就是這個 Activity 而已，一點也不複雜。

重點特色說明：

- 左側是目前準備繪製的影像元素，會出現 + 字號
- 利用觸控螢幕特性直接在螢幕點觸或是滑動繪製
- Android 只有一個，所以依照最新點觸位置為主
- 一邊繪製可以一邊思考關卡特性

RunnerMapEditorActivity.java

```
package tw.brad.android.games.RunnerMapEditor;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
```

```
import java.io.IOException;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.SharedPreferences;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Matrix;
import android.graphics.Paint;
import android.graphics.Paint.Style;
import android.os.Bundle;
import android.util.AttributeSet;
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view MotionEvent;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class RunnerMapEditorActivity extends Activity {
    private MapEditor mapeditor;
    private Resources res;
    private int sw, sh;
    private float wp, hp;
    private String savename, openname;
    private SharedPreferences sp;
    private SharedPreferences.Editor sp_editor;
    private int now_level;
    private boolean isNew;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        res = getResources();
        DisplayMetrics dm = res.getDisplayMetrics();
        sw = dm.widthPixels; sh = dm.heightPixels;
        wp = sw / 24.0f; hp = sh / 16.0f;
```

```
sp = getSharedPreferences("level", MODE_PRIVATE);
sp_editor = sp.edit();

mapeditor = new MapEditor(this, null);
setContentView(mapeditor);

now_level = -1; isNew = true;

}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.optionmenu, menu);

    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()){
        case R.id.save:
            mapeditor.saveMap();
            break;
        case R.id.open:
            mapeditor.openMap();
            break;
        case R.id.clear:
            mapeditor.clearMap();
            break;
        case R.id.newlevel:
            now_level = -1; isNew = true;
            mapeditor.clearMap();
            break;
    }
    return super.onOptionsItemSelected(item);
}

private class MapEditor extends View {
    private int[][] map = new int[16][24];
    private Bitmap bmp_b, bmp_l, bmp_ll, bmp_h, bmp_a, bmp_man,
    bmp_enemy, bmp_next;
    private int[] mode = {0,1,2,3,7,8,9,-1};
    private int now = 1;
    private int now_r, now_c;
```

```
private Paint info_txt;
private int count_man = 0;

public MapEditor(Context context, AttributeSet attrs) {
    super(context, attrs);

    bmp_b = BitmapFactory.decodeResource(res, R.drawable.bricks);
    int bmp_bw = bmp_b.getWidth();
    int bmp_bh = bmp_b.getHeight();

    Matrix matrix = new Matrix();
    matrix.postScale((wp)/bmp_bw, (hp)/bmp_bh);
    bmp_b = Bitmap.createBitmap(bmp_b, 0, 0, bmp_bw, bmp_bh, matrix, true);

    bmp_l = BitmapFactory.decodeResource(res, R.drawable.ladder);
    bmp_bw = bmp_l.getWidth();
    bmp_bh = bmp_l.getHeight();
    matrix.reset();
    matrix.postScale((wp)/bmp_bw, (hp)/bmp_bh);
    bmp_l = Bitmap.createBitmap(bmp_l, 0, 0, bmp_bw, bmp_bh, matrix, true);

    bmp_ll = BitmapFactory.decodeResource(res, R.drawable.ladder_1);
    bmp_bw = bmp_ll.getWidth();
    bmp_bh = bmp_ll.getHeight();
    matrix.reset();
    matrix.postScale((wp)/bmp_bw, (hp)/bmp_bh);
    bmp_ll = Bitmap.createBitmap(bmp_ll, 0, 0, bmp_bw, bmp_bh, matrix, true);

    bmp_h = BitmapFactory.decodeResource(res, R.drawable.hbar);
    bmp_bw = bmp_h.getWidth();
    bmp_bh = bmp_h.getHeight();
    matrix.reset();
    matrix.postScale((wp)/bmp_bw, (hp)/bmp_bh);
    bmp_h = Bitmap.createBitmap(bmp_h, 0, 0, bmp_bw, bmp_bh, matrix, true);

    bmp_a = BitmapFactory.decodeResource(res, R.drawable.apple);
    bmp_bw = bmp_a.getWidth();
    bmp_bh = bmp_a.getHeight();
    matrix.reset();
```

```
        matrix.postScale((wp)/bmp_bw, (hp)/bmp_bh);
        bmp_a = Bitmap.createBitmap(bmp_a, 0, 0, bmp_bw, bmp_bh, matrix, true);

        bmp_man = BitmapFactory.decodeResource(res, R.drawable.android);
        bmp_bw = bmp_man.getWidth();
        bmp_bh = bmp_man.getHeight();
        matrix.reset();
        matrix.postScale((wp)/bmp_bw, (hp)/bmp_bh);
        bmp_man = Bitmap.createBitmap(bmp_man, 0, 0, bmp_bw, bmp_bh, matrix, true);

        bmp_enemy = BitmapFactory.decodeResource(res, R.drawable.enemy);
        bmp_bw = bmp_enemy.getWidth();
        bmp_bh = bmp_enemy.getHeight();
        matrix.reset();
        matrix.postScale((wp)/bmp_bw, (hp)/bmp_bh);
        bmp_enemy = Bitmap.createBitmap(bmp_enemy, 0, 0, bmp_bw, bmp_bh, matrix, true);

        bmp_next = BitmapFactory.decodeResource(res, R.drawable.next);
        bmp_bw = bmp_next.getWidth();
        bmp_bh = bmp_next.getHeight();
        matrix.reset();
        matrix.postScale((wp)/bmp_bw, (hp)/bmp_bh);
        bmp_next = Bitmap.createBitmap(bmp_next, 0, 0, bmp_bw, bmp_bh, matrix, true);

        now_r = now_c = 0;

        info_txt = new Paint();
        info_txt.setColor(Color.WHITE);
        info_txt.setTextSize(36);
        info_txt.setStyle(Style.FILL_AND_STROKE);
        info_txt.setStrokeWidth(2);

        clearMap();
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        int click_r = (int)(event.getY()/hp);
        int click_c = (int)(event.getX()/wp);
        if (click_r == now_r && click_c == now_c) {
```

```

        now = (now == mode.length-1)?0:now+1;
    }else {
        now_r = click_r; now_c = click_c;
    }
    if (count_man == 1 && mode[now] == 8)now = (now ==
mode.length-1)?0:now+1;
    if (count_man == 0 || (count_man ==1 && mode[now] != 8)){
        map[click_r][click_c] = mode[now];
        postInvalidate();
    }
    return super.onTouchEvent(event);
}

@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);

    canvas.drawText("Level: " + now_level, 4, 40, info_txt);

    count_man = 0;
    for (int r = 0; r<map.length; r++){
        for (int c = 0; c<map[r].length; c++){
            if (map[r][c]==1){
                canvas.drawBitmap(bmp_b, c*wp, r*hp, null);
            }else if (map[r][c]==2){
                canvas.drawBitmap(bmp_l, c*wp, r*hp, null);
            }else if (map[r][c]==3){
                canvas.drawBitmap(bmp_h, c*wp, r*hp, null);
            }else if (map[r][c]==7){ // 敵人
                canvas.drawBitmap(bmp_enemy, c*wp, r*hp, null);
            }else if (map[r][c]==8){ // Android
                canvas.drawBitmap(bmp_man, c*wp, r*hp, null);
                count_man = 1;
            }else if (map[r][c]==9){ // Apple
                canvas.drawBitmap(bmp_a, c*wp, r*hp, null);
            }else if (map[r][c]==-9){ // 過關
                canvas.drawBitmap(bmp_next, c*wp, r*hp, null);
            }else if (map[r][c]==-1){ // 過關梯
                canvas.drawBitmap(bmp_l1, c*wp, r*hp, null);
            }
        }
    }

    private void clearMap(){
}

```

```
map = new int[16][24];
for (int j=0; j<24; j++){
    map[15][j] = 1;
}
map[0][23] = -9;
postInvalidate();
}

private void saveMap(){
    LayoutInflator dialog = LayoutInflator.
        from(RunnerMapEditorActivity.this);
    final View dview = dialog.inflate(R.layout.dialog_save, null);
    EditText et = (EditText)dview.findViewById(R.id.savename);
    et.setText(sp.getInt("level", 0) + "");

    AlertDialog.Builder builder = new AlertDialog.
    Builder(RunnerMapEditorActivity.this);

    builder.setTitle("Save Level");
    builder.setView(dview);
    builder.setCancelable(true);
    builder.setPositiveButton("OK", new DialogInterface.
    OnClickListener(){
        @Override
        public void onClick(DialogInterface arg0, int arg1) {
            EditText et = (EditText)dview.findViewById(R.
            id.savename);

            savename = et.getText().toString();
            String header = savename;
            if (savename.length()==1){
                savename = "00" + savename;
            }else if (savename.length()==2){
                savename = "0" + savename;
            }

            try {
                BufferedWriter bw = new BufferedWriter(
                    new FileWriter("/mnt/sdcard/level" + savename));
                StringBuffer line;
                for (int r=0; r<map.length; r++){
                    line = new StringBuffer();
                    line.append(header + ":");
                    for (int c=0; c<map[r].length; c++){
                        line.append(map[r][c]);
                    }
                    bw.write(line.toString());
                }
                bw.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    });
    builder.show();
}
```

```
        if (c==map[r].length-1) {
            line.append("\n");
        }else{
            line.append(",");
        }
    bw.write(line.toString());
}
bw.flush();
bw.close();
if (isNew){
    sp_editor.putInt("level", Integer.parseInt(savename)+1);
    sp_editor.commit();
}
Toast.makeText(RunnerMapEditorActivity.this,
    "Save OK", Toast.LENGTH_LONG).show();
} catch (FileNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

}

});
AlertDialog alert = builder.create();
alert.show();

}
private void openMap(){
    LayoutInflater dialog = LayoutInflater.from
        (RunnerMapEditorActivity.this);
    final View dview = dialog.inflate(R.layout.dialog_open, null);
    EditText et = (EditText)dview.findViewById(R.id.openname);
    AlertDialog.Builder builder = new AlertDialog.Builder
        (RunnerMapEditorActivity.this);

    builder.setTitle("Open Level");
    builder.setView(dview);
    builder.setCancelable(true);
    builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
        @Override
```

```
public void onClick(DialogInterface arg0, int arg1) {
    EditText et = (EditText)dview.findViewById(R.
        id.openname);

    openname = et.getText().toString();
    now_level = Integer.parseInt(openname); isNew = false;
    if (openname.length()==1){
        openname = "00" + openname;
    }else if (openname.length()==2){
        openname = "0" + openname;
    }

    try {
        BufferedReader reader = new BufferedReader(
            new FileReader("/mnt/sdcard/level" +
                openname));

        String temp1;
        String[] temp3;
        int v, i = 0, j = 0;
        while ((temp1 = reader.readLine())!=null){
            temp3 = temp1.split(",");
            for(String temp4 : temp3){
                v = Integer.parseInt(temp4);
                map[i][j++] = v;
            }
            i++; j=0;
        }
        reader.close();
        postInvalidate();
        Toast.makeText(RunnerMapEditorActivity.
            this, "Load OK", Toast.LENGTH_LONG).show();
    } catch (FileNotFoundException e) {
        now_level = -1; isNew = true;
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
});
```

AlertDialog alert = builder.create();
alert.show();

```
}
```

敏感爭議

比起經典的 Lode Runner 還多帶了一個明確的故事性，但是也利用故事隱含了諷刺性。蘋果和芒果剛好是另外兩個陣營的代表性水果，遊戲主軸不是掏金沙，而改成吃掉蘋果或是芒果，就算是勝利過關。而在 2011 年參加 App 高手爭霸時的評審看過展示後，都不約而同的覬覦一笑，那場比賽是三大陣營的 App 共同比賽，光是觸及敏感議題就不易出線。

而在 2012 年再度將該款遊戲中的蘋果改成電子發票圖示，主題改為發票蒐集王「Invoice Runner」，參加中區國稅局的 App 創意設計大賽獲得佳作。

15



15-3 炸彈超人 Bomb King

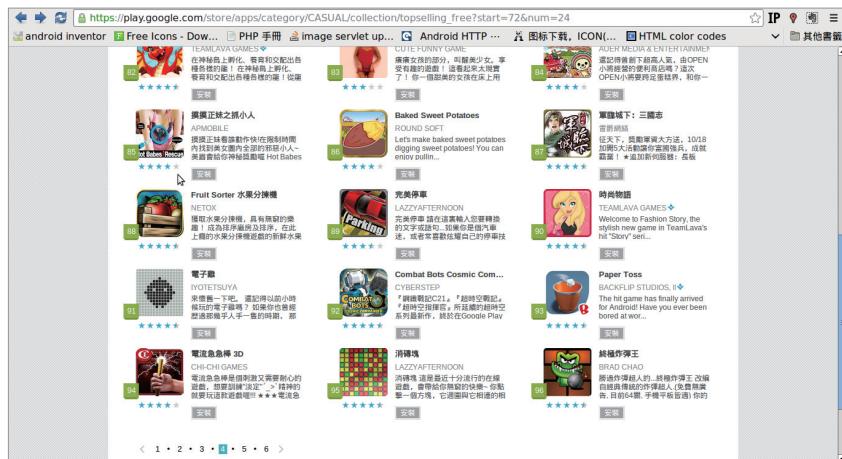
筆者開發的磚塊系列，繼 Lode Runner，Bricks Fighter 之後的第三發。

原名為 Bomber Man 的炸彈超人，應該不用太多詳細介紹的經典遊戲。其實這款遊戲與 Lode Runner 有著蠻大的相似度，有著一樣的操控模式，雖然一個是縱切面的視角 (Lode Runner)，而一個是橫切面 (Bomb King)，反而縱切面視角要處理從上而下的落下動作，橫切面就簡單許多。

Bomb King 的遊戲樂趣在於玩家要控制手上有限的炸彈數量，將炸彈放在正確的放置，才能炸出藏在磚塊牆壁中的寶物，並且救出小雞。詳細完整原始碼請參考本書所附的光碟。

整體遊戲架構也與 Lode Runner 類似，放在此說明只是想要始讀者明白一點，事實上許多遊戲在骨子裡都是非常類似的處理。試回想，馬利歐不也是在玩磚塊嗎？會跳，那是用當時行進的速度，加上其他因素來決定跳躍的力量而已，並且將關卡改為捲軸式處理；再進化就又可以改編魂斗羅遊戲了 …

Bomb King 曾擠進 Google play 市集休閒遊戲百大熱門。



歡迎畫面



關卡選擇畫面處理，



主要遊戲畫面：

15



勝利過關畫面，



遊戲解說畫面，



15-4

其他應用程式開發專案

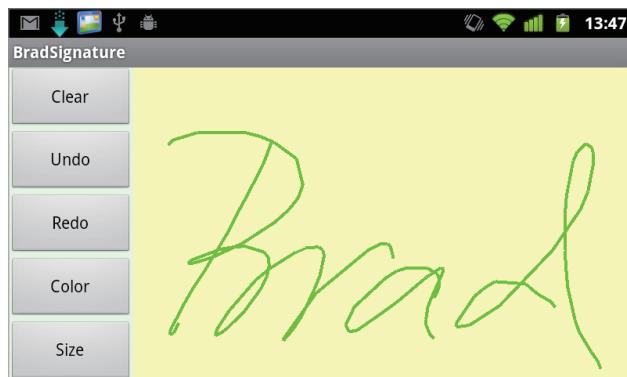
上述三個小節都是以遊戲為主軸的開發，而一般的文件或是相關書籍介紹，都會提及製作 2D 或是 3D 的遊戲專案大多數是以 SurfaceView 來處理，Why？作者 Brad 也不知道為何？！但是 Brad 用實際的專案開發來提出反證，就是製作 2D 遊戲可以使用自訂 View 即可達到功效，至少不是說說而已，是以實際專案開發證明，並且可以同時滿足手機和大到 10 吋的平板電腦（至少 Brad 的 TF-101 變形金剛第一代預購版）都可以跑得非常順暢。一個結論，在技術層面上，別人說的是我的參考，事實是自己證明出來的。專家，很多甚至是自己專門騙人家都還不自知的，因為說說建議而已可以不用負責任，這類的專家常見於各大媒體論壇場合，包括 Brad 本人，重點是學習者不應該只照單全收，而是需要馬上作實驗。

再來完成前面章節中的觸控簽名 App 吧。

個性簽名產生器

換名稱了？總要有個響亮名稱，才能襯托出有自信的 App，所以從此改稱為個性簽名產生器。還記得開發狀況吧，目前可以簽名，有 Clear／Undo／Redo 的功能，繼續開發下去的功能是：

- 改變顏色
- 改變簽字筆的粗細
- 最後可以存檔



15

改變顏色對於目前程式而言，絕對是相當簡單的一件事，只需要將 Paint 物件實體設定為使用者想要的顏色即可，但是如何提供給使用者一個友善的介面呢？

這邊就先不考慮自行開發，先上網看看有沒有適合的 API 可以使用。Brad 的開發原則：「站在巨人的肩膀上面，可以跑得比較快，前提是這位巨人願意」。

上網找到 Android Open Source Project 的開放原始碼如下：

ColorPickerDialog.java

```
/*
 * Copyright (C) 2007 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
```

```
* See the License for the specific language governing permissions and
* limitations under the License.
*/
package com.example.android.apis.graphics;

import android.R;
import android.os.Bundle;
import android.app.Dialog;
import android.content.Context;
import android.graphics.*;
import android.view.MotionEvent;
import android.view.View;

public class ColorPickerDialog extends Dialog {

    public interface OnColorChangedListener {
        void colorChanged(int color);
    }

    private OnColorChangedListener mListener;
    private int mInitialColor;

    private static class ColorPickerView extends View {
        private Paint mPaint;
        private Paint mCenterPaint;
        private final int[] mColors;
        private OnColorChangedListener mListener;

        ColorPickerView(Context c, OnColorChangedListener l, int color) {
            super(c);
            mListener = l;
            mColors = new int[] {
                0xFFFF0000, 0xFFFF00FF, 0xFF0000FF, 0xFF00FFFF, 0xFF00FF00,
                0xFFFFFF00, 0xFFFF0000
            };
            Shader s = new SweepGradient(0, 0, mColors, null);

            mPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
            mPaint.setShader(s);
            mPaint.setStyle(Paint.Style.STROKE);
            mPaint.setStrokeWidth(32);

            mCenterPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
            mCenterPaint.setColor(color);
            mCenterPaint.setStrokeWidth(5);
        }
    }
}
```

```
}

private boolean mTrackingCenter;
private boolean mHighlightCenter;

@Override
protected void onDraw(Canvas canvas) {
    float r = CENTER_X - mPaint.getStrokeWidth()*0.5f;

    canvas.translate(CENTER_X, CENTER_X);

    canvas.drawOval(new RectF(-r, -r, r, r), mPaint);
    canvas.drawCircle(0, 0, CENTER_RADIUS, mCenterPaint);

    if (mTrackingCenter) {
        int c = mCenterPaint.getColor();
        mCenterPaint.setStyle(Paint.Style.STROKE);

        if (mHighlightCenter) {
            mCenterPaint.setAlpha(0xFF);
        } else {
            mCenterPaint.setAlpha(0x80);
        }
        canvas.drawCircle(0, 0,
                CENTER_RADIUS + mCenterPaint.getStrokeWidth(),
                mCenterPaint);

        mCenterPaint.setStyle(Paint.Style.FILL);
        mCenterPaint.setColor(c);
    }
}

@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    setMeasuredDimension(CENTER_X*2, CENTER_Y*2);
}

private static final int CENTER_X = 100;
private static final int CENTER_Y = 100;
private static final int CENTER_RADIUS = 32;

private int floatToByte(float x) {
    int n = java.lang.Math.round(x);
    return n;
}
private int pinToByte(int n) {
```

```
if (n < 0) {
    n = 0;
} else if (n > 255) {
    n = 255;
}
return n;
}

private int ave(int s, int d, float p) {
    return s + java.lang.Math.round(p * (d - s));
}

private int interpColor(int colors[], float unit) {
    if (unit <= 0) {
        return colors[0];
    }
    if (unit >= 1) {
        return colors[colors.length - 1];
    }

    float p = unit * (colors.length - 1);
    int i = (int)p;
    p -= i;

    // now p is just the fractional part [0...1) and i is the index
    int c0 = colors[i];
    int c1 = colors[i+1];
    int a = ave(Color.alpha(c0), Color.alpha(c1), p);
    int r = ave(Color.red(c0), Color.red(c1), p);
    int g = ave(Color.green(c0), Color.green(c1), p);
    int b = ave(Color.blue(c0), Color.blue(c1), p);

    return Color.argb(a, r, g, b);
}

private int rotateColor(int color, float rad) {
    float deg = rad * 180 / 3.1415927f;
    int r = Color.red(color);
    int g = Color.green(color);
    int b = Color.blue(color);

    ColorMatrix cm = new ColorMatrix();
    ColorMatrix tmp = new ColorMatrix();

    cm.setRGB2YUV();
    tmp.setRotate(0, deg);
```

```
cm.postConcat(tmp);
tmp.setYUV2RGB();
cm.postConcat(tmp);

final float[] a = cm.getArray();

int ir = floatToByte(a[0] * r + a[1] * g + a[2] * b);
int ig = floatToByte(a[5] * r + a[6] * g + a[7] * b);
int ib = floatToByte(a[10] * r + a[11] * g + a[12] * b);

return Color.argb(Color.alpha(color), pinToByte(ir),
                  pinToByte(ig), pinToByte(ib));
}

private static final float PI = 3.1415926f;

@Override
public boolean onTouchEvent(MotionEvent event) {
    float x = event.getX() - CENTER_X;
    float y = event.getY() - CENTER_Y;
    boolean inCenter = java.lang.Math.sqrt(x*x + y*y) <= CENTER_RADIUS;

    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            mTrackingCenter = inCenter;
            if (inCenter) {
                mHighlightCenter = true;
                invalidate();
                break;
            }
        case MotionEvent.ACTION_MOVE:
            if (mTrackingCenter) {
                if (mHighlightCenter != inCenter) {
                    mHighlightCenter = inCenter;
                    invalidate();
                }
            } else {
                float angle = (float)java.lang.Math.atan2(y, x);
                // need to turn angle [-PI ... PI] into unit [0....1]
                float unit = angle/(2*PI);
                if (unit < 0) {
                    unit += 1;
                }
                mCenterPaint.setColor(interpColor(mColors, unit));
                invalidate();
            }
    }
}
```

```

        break;
    case MotionEvent.ACTION_UP:
        if (mTrackingCenter) {
            if (inCenter) {
                mListener.colorChanged(mCenterPaint.getColor());
            }
            mTrackingCenter = false; // so we draw w/o halo
            invalidate();
        }
        break;
    }
    return true;
}
}

public ColorPickerDialog(Context context,
                         OnColorChangedListener listener,
                         int initialColor) {
    super(context);

    mListener = listener;
    mInitialColor = initialColor;
}

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    OnColorChangedListener l = new OnColorChangedListener() {
        public void colorChanged(int color) {
            mListener.colorChanged(color);
            dismiss();
        }
    };
    setContentView(new ColorPickerView(getContext(), l, mInitialColor));
    setTitle("Pick a Color");
}
}

```

心中先感謝原創的作者，再好好檢視一下程式碼，發現就是一個自訂 Dialog，Dialog 的重點是必須建構出物件實體之後呼叫 show() 方法，並實做相關的監聽事件方法即可，於是就直接拿來使用，改為 MyColorPicker.java，放在相同的 Package 之下。

先在 PaintView.java 中建立兩個方法：

- int getColor()：取得目前的顏色值
- void setColor(int newcolor)：設定新的顏色值

```
int getColor() {
    return paintLine.getColor();
}

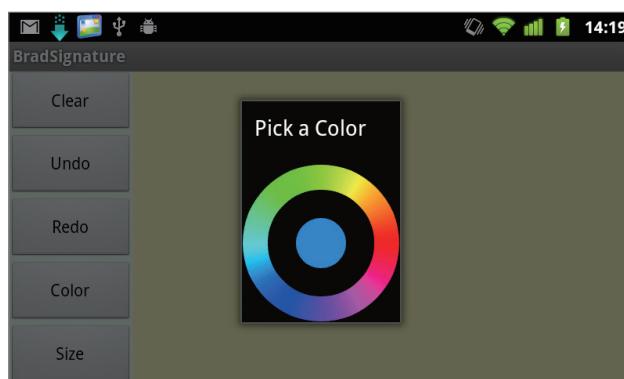
void setColor(int newcolor) {
    paintLine.setColor(newcolor);
}
```

再回到 MainActivity.java 中，增加處理 chcolor 的按鈕物件功能，就變成非常簡單的一件事了。

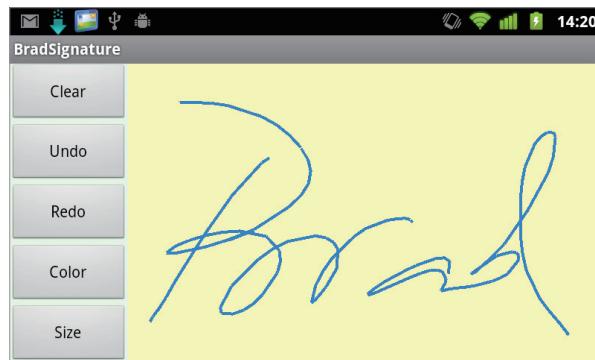
15

```
chcolor = findViewById(R.id.chcolor);
chcolor.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        new MyColorPicker(MainActivity.this, new OnColorChangedListener() {
            @Override
            public void colorChanged(int color) {
                pview.setColor(color);
            }
        }, pview.getColor()).show();
    }
});
```

按下 Color 按鈕後呈現如下圖：



試著簽名看看：



再來處理改變簽字筆粗細及存檔功能，先將版面配置作些許異動：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="4"
        android:background="#1100ff00"
        android:orientation="vertical" >

        <Button
            android:id="@+id/clear"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Clear" />

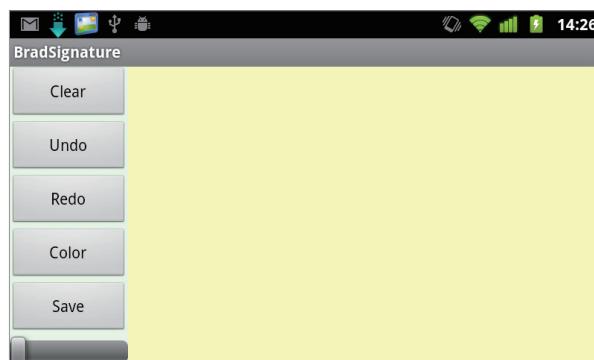
        <Button
            android:id="@+id/undo"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Undo" />

        <Button
            android:id="@+id/redo"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Redo" />
    
```

15

```
<Button  
    android:id="@+id/chcolor"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_weight="1"  
    android:text="Color" />  
  
<Button  
    android:id="@+id/save"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_weight="1"  
    android:text="Save" />  
  
<SeekBar  
    android:id="@+id/chsize"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_weight="1"  
    android:text="Size" />  
  
</LinearLayout>  
  
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:layout_weight="1"  
    android:orientation="vertical" >  
  
<tw.brad.apps.BradSignature.PaintView  
    android:id="@+id/pview"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:background="#44ffff00" />  
</LinearLayout>  
  
</LinearLayout>
```

最後呈現如下：



先從調整簽字筆粗細開始，因為是使用SeekBar，在之前單元的音樂播放上使用過，應該比較不陌生。先從PaintView下手處理兩個方法：

- int getSize()：傳回目前粗細大小
- void setSize()：設定目前粗細大小

```
int getSize() {
    return (int)paintLine.getStrokeWidth();
}

void setSize(int newsize) {
    paintLine.setStrokeWidth(newsize);
}
```

回到 MainActivity.java 中：

```
chsizer = (SeekBar) findViewById(R.id.chsize);
chsizer.setMax(24);
chsizer.setProgress(pview.getSize());
chsizer.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onProgressChanged(SeekBar seekBar, int progress,
        boolean fromUser) {
        if (fromUser) {
            pview.setSize(progress);
        }
    }
});
```

這樣就可以變更簽字筆粗細。

再來就將簽名檔案存檔發送出去，建議將背景改為透明方式來處理簽名影像檔案。而要將 View 的顯示內容進行存檔，應該事先呼叫 setDrawingCacheEnabled(true)，否則預設為 false。

```
pview = (PaintView) findViewById(R.id.pview);
pview.setBackgroundColor(Color.TRANSPARENT);
pview.setDrawingCacheEnabled(true);
```

處理按下 Save：

```
save = findViewById(R.id.save);
save.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        FileOutputStream fout;
        try {
            fout = new FileOutputStream("/mnt/sdcard;bradsign.png");
            boolean isSaveOK = pview.getDrawingCache().compress(
                CompressFormat.PNG, 100, fout);
            if (isSaveOK) {
                Intent i = new Intent(Intent.ACTION_SEND);
                i.setType("message/rfc822");
                i.putExtra(Intent.EXTRA_EMAIL,
                    new String[] { "brad@brad.tw" });
                i.putExtra(Intent.EXTRA_SUBJECT, "Brad的簽名");
                i.putExtra(Intent.EXTRA_TEXT, "如附件");
                i.putExtra(Intent.EXTRA_STREAM, Uri.fromFile(new File(
                    "/mnt/sdcard;bradsign.png")));
                try {
                    startActivityForResult(Intent.createChooser(i,
                        "Send mail..."));
                } catch (android.content.ActivityNotFoundException ex) {
                    Toast.makeText(MainActivity.this, "ERR",
                        Toast.LENGTH_SHORT).show();
                }
            }
        } catch (FileNotFoundException e) {
        }
    });
});
```

15

要記得開啟相關的使用權限：

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.INTERNET"/>
```

這樣完成一個小品專案。

|| 開發觀念原則 ||

目前許多在 Android App 的專案開發中，其實架構流程並不大，比起以往 Web-based 的專案而言，真的非常小，真要說團隊合作，大概頂多就是兩三個人就可以，一個開發程式、一個視覺藝術，加上一個音效製作。但是 Brad 很常看到許多創投育成中心，不斷強調團隊開發合作模式的重要性，還必須要要特定地點大家非得面對面討論，激發創意靈感等等。Brad 却深深不以為然，至少對於程式開發的部份真的沒有那麼複雜，就是作自己想做的東西，做出來的東西都無法滿足自己的要求，更何況是全世界使用者的嚴酷考驗。面對面互相討論的事情，難道視訊無法克服嗎？而在台中、花蓮、台東所 Google 搜尋出來的結果會與在台北的結果不一樣嗎？如果真的不一樣，那麼創投們何不去直接去國外設立呢？

因此，一個重要的觀念就是，Android Apps 給了許多獨立開發者一個很大的開發舞台。學習者老是認為開發工作有多艱鉅，事實上就是去作而已 (Just do it)，以本書中所開發的個性簽名，重點就是如何規劃以資料結構來處理簽名劃線，如果處理得宜，後面的功能性開發就非常容易。通常這只是授課實務範例而已。

16

Chapter

APP 發佈

16-1 包裝發佈到 Google Play

16-2 App 創意開發與比賽經驗心得分享





16-1 包裝發佈到 Google Play

Google Play 算是在 Android Apps 上非常大的交易市集。開發者可以透過這個平台，將所開發的 Android App 上傳上去後，以付費或是免費的機制散佈到世界上大多數地區的使用者，而使用者得以利用這個平台機制，找尋喜愛的 App，進而免費或是付費下載使用。

包裝成為 apk

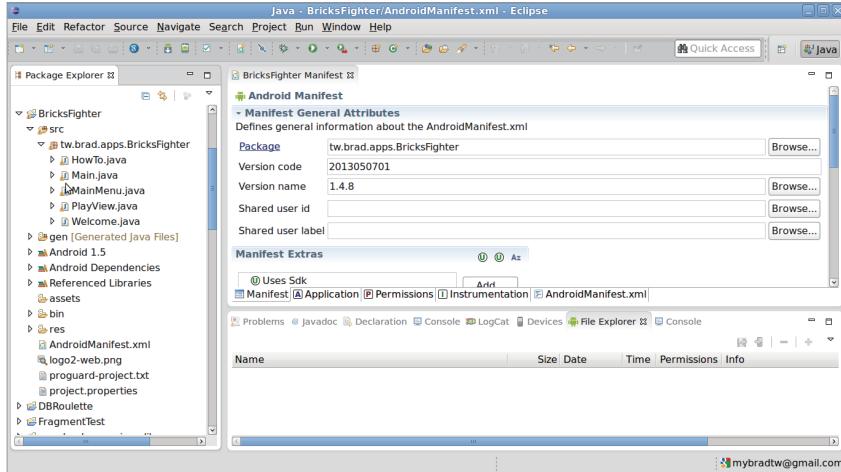
開發完成的專案，終於要與世人見面了！以下就以 Bricks Fighter 專案實際處理方式介紹認識。

開啟專案目錄下的 AndroidManifest.xml 中的 Manifest 的頁籤，看到有兩項重要資料：

- Version code：版本碼
- Version name：版本名稱

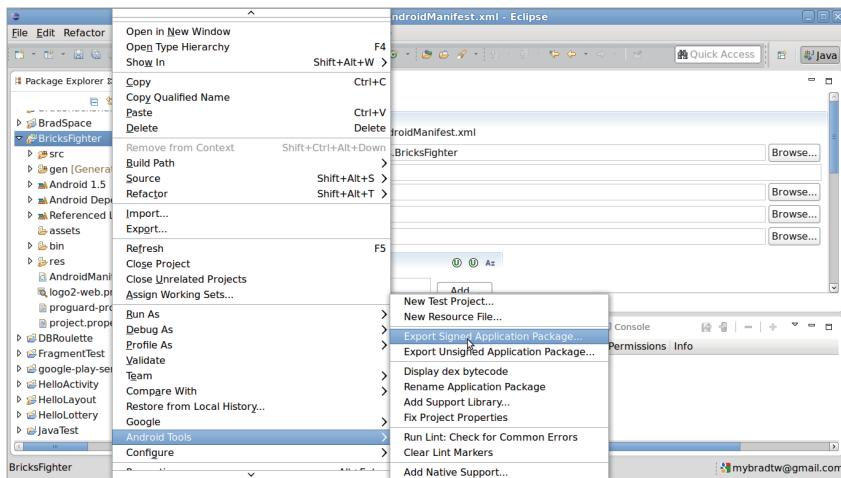
Version code 是開發者識別專用的碼，每次上傳到 Google Play 的時候，會檢查該碼是否比上次上傳的數字要大，作者建議的編碼方式是 YYYYMMDD01，亦即西元年 + 月份 + 日期 + 兩位流水碼，因為今天的日期永遠比昨天大，流水碼配置兩位數，一天要能更新發布 99 次應該是不常見的事，而往後還可以透過該碼知道上次發布的時間。

Version name 則為使用者會看到的版本名稱。

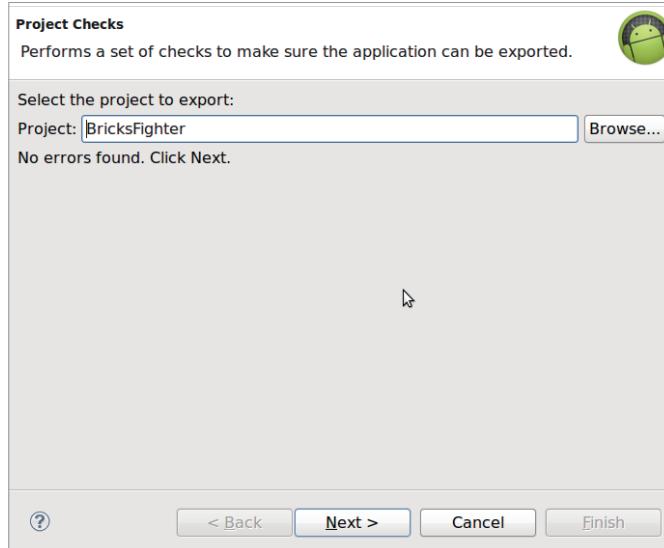


接下來點按專案名稱右鍵，「Android Tools → Export Signed Application Package」，如下展開：

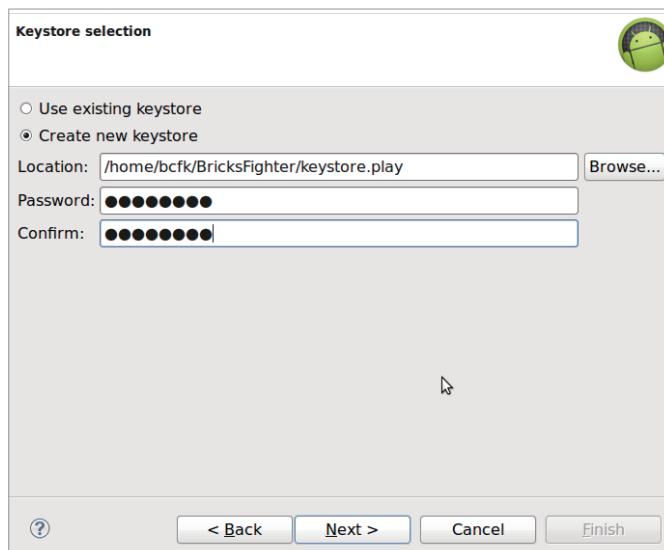
16



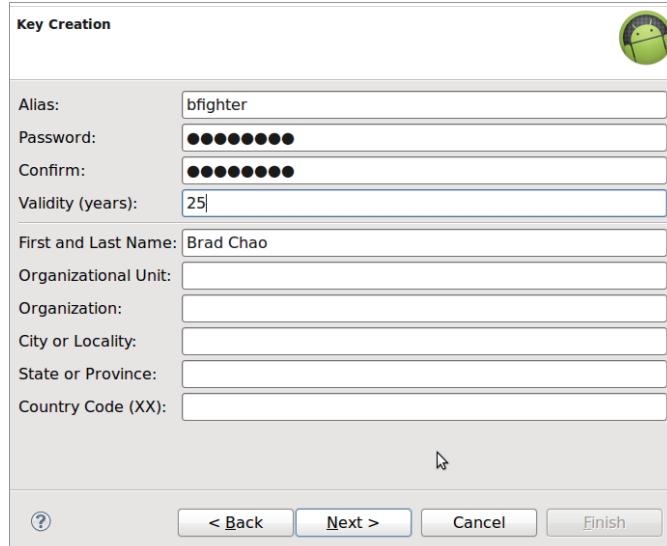
開始進行一連串詢問精靈視窗，選擇專案進行匯出。



第一次發佈沒有任何 keystore 檔案，必須建立一組。而該 App 往後的更新發佈都是以該組 keystore 進行，因此務必善加保存該檔案。

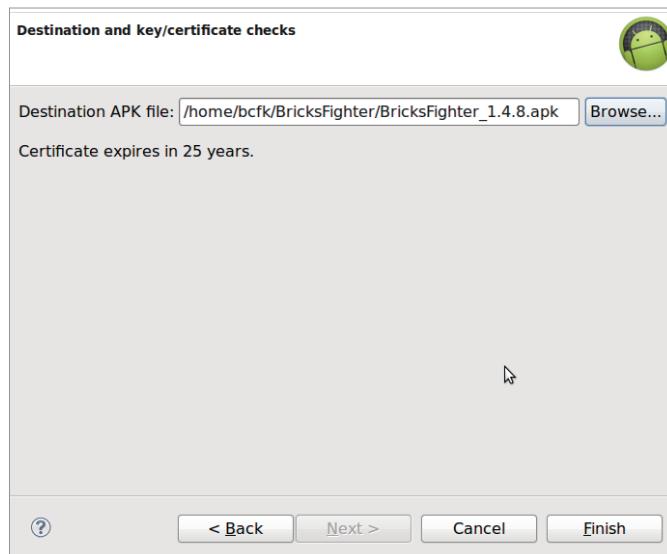


填寫該組 keystore 相關資料，以下圖舉例為最基本要填寫的項目。(有效期，官方建議是 25 年)



16

設定 Apk 檔案產生的目錄及檔名，作者建議加上版本名稱以作識別。



按下「Finish」後開始進行囉。

■首次註冊開發者■

開發者想要透過 Google Play 交易平台發佈，有以下幾個重點要進行：

1. 註冊 Google 帳號，最簡單的方式就是 Gmail 的帳號即可。
2. 註冊 Google Play 開發人員控制台帳號。
3. 25 元美金。
4. 註冊程序依照官方網站說明為 48 小時內，作者實際經驗應該是 24 小時之內（甚至更快）。
5. 如果要販售付費 Apps，之前需要另外註冊 Google Checkout 商家，從 2013 年 3 月起，已經由 Google 電子錢包商家中心取代。

前一陣子台北市政府對於 Google Play 僅提供消費者 15 分鐘之猶豫期，而使該問題爭議進入法律程序，而在 2012 年 12 月 27 日對外發佈已遭法院判決撤銷，表示 Google Play 上面的所有開發者可以對台灣進行發佈付費 Apps。而也在 2013 年農曆過年前後開始，陸續有使用者發現可以下載付費的 Apps 了。

作者對於此事件無意論斷孰是孰非，但是對於台灣許多開發者的建議，國際世界市場應該比台灣的消費市場大很多，針對創意性的創作，不要只是著眼開發應用層面較為區域性的 Apps。當然，如果是一家中小企業或是機關組織為了其商業或是宣傳便民之目的而開發的作品，那就另當別論。有公司要開發社區管理統，僅提供該公司管轄的社區住戶使用的 Apps，功能包含掛號信通知，公告通知等等。有特殊目的需求，當然就開發囉，那叫做接案子。如果現在閒下來，想要進行創意性的開發，我不會想到要作 " 某地美食報你知 " 這類的 Apps，因為從下載廣告或是付費用戶數量觀點來看，付出的開發代價應該不太值得的。

註冊程序中的 Google Play 開發人員註冊非常簡單：

1. 使用 Google 帳戶登入
2. 接受開發人員協議

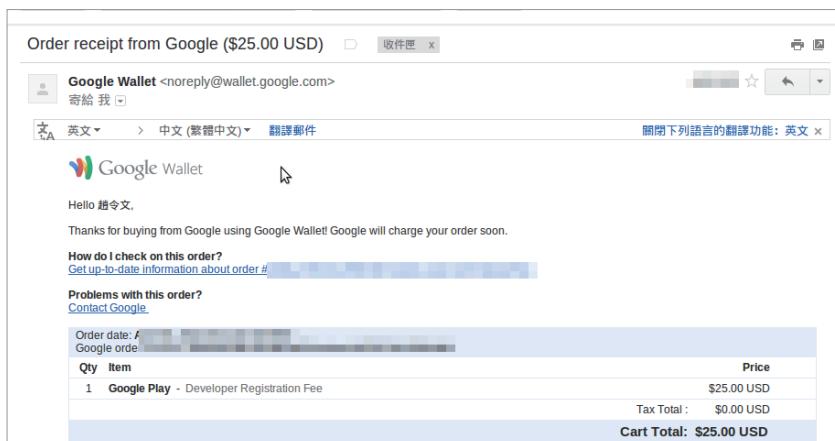
3. 支付註冊費

4. 填妥帳戶詳細資料

搜尋 play developer console 即可找到開發人員控制台。當以 Google 帳戶登入後，會看到以下網頁內容：



跟著步驟進行就應該可以順利完成註冊（重點是要付 25 元美金）。應該會收到如下圖的通知信件：



從此之後，如果都是以該帳號發布 Apps，都不再需要支付其他費用，這一點與其他兩個陣營的機制是不一樣之處。

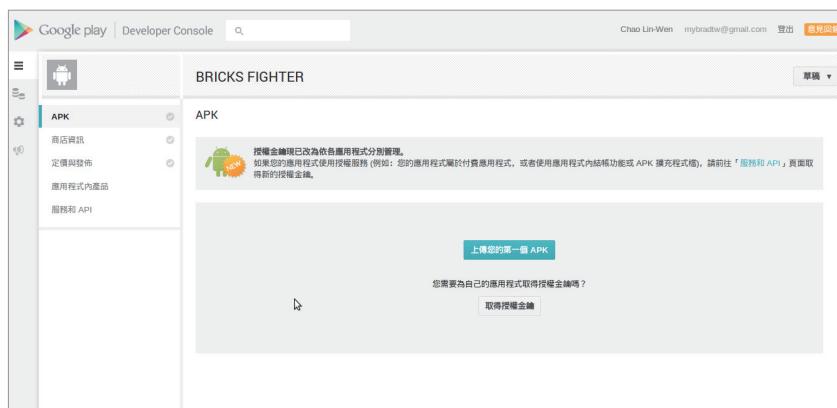
■ 發佈 Apk 到 Google Play ■

App 已經開發完畢，也包裝成為 .apk；而 Google Play 也已經註冊為開發者，就開始進行發布的動作了。

首先先登錄到開發者控制台，直接點按「新增應用程式」後，出現如下畫面：



開始將已經包裝好的 apk 檔案進行上傳。



正常上傳中，



完成上傳後出現的狀況報告，

版本代碼	版本名稱	支援的裝置	排除的裝置
2013050701	1.4.8	2645 顯示詳細資訊	0 管理排除的裝置

16

點選左側的商品資訊後填寫相關資料，

名稱 *	Bricks Fighter
英文 (美國) – en-US	已輸入 14 個字元 (最多 30 個字元)
說明 *	Brick, Brick, more Brick. Break bricks by your Finger.
宣傳文字	已輸入 54 個字元 (最多 4000 個字元) Brick, Brick, more Brick.

螢幕截圖的提供有助於使用者了解 App 的外觀。



填完以上資料後，點按「定價與發布」即可開始設定付費與免費相關機制。

一切順利後就算是完成上架程序了。



16-2 App 創意開發與比賽經驗心得分享

還是一句老話，作者 Brad 並非專家（也不是專門騙人家，從事教育工作時戲謔為誤人子弟），僅就這近四年來玩 Android Apps 開發及教學經驗作分享。而本書中所分享的開放原始碼，也提供最完整的專案程式碼，希望讀者可以檢視到最完整的應用，但是 Brad 並非最專業開發工作者，所以提供的原始碼也並非專業的開發考量，期待讀者能以學習的角度參考使用。

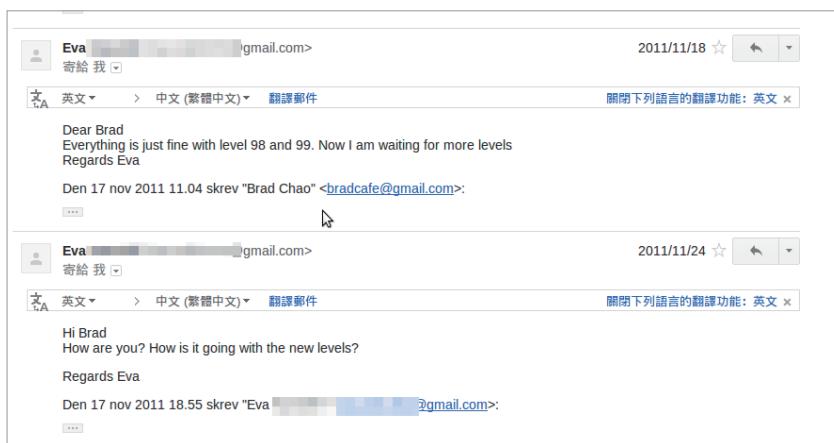
創意的發想是開發者面臨的第一個問題，Brad 也不是創意達人，只想作自己想要的東西而已。2010 年間在資策會輔導學員製作專題時，曾經提供學員一個想法：「既然行動裝置都有網際網路的應用，何不寫個類似 MSN 的 App，這樣使用者就可以省下簡訊的費用」，後來學員還是做了其他主題，而我當年如果完成這個想法，那現在可能你不是用 Line，而是 BradMSN...（後

悔 ing)。無論如何，我還是作我想做的東西，前年開始想寫些紅白機系列的遊戲，於是就從 Lode Runner 下手，陸續有炸彈超人及打磚塊等等。這樣的過程至少滿足自己所需，而且也和兒子玩得非常開心。當然，如果也能帶給全世界不相識的玩家快樂，那又是不同的喜悅 ...

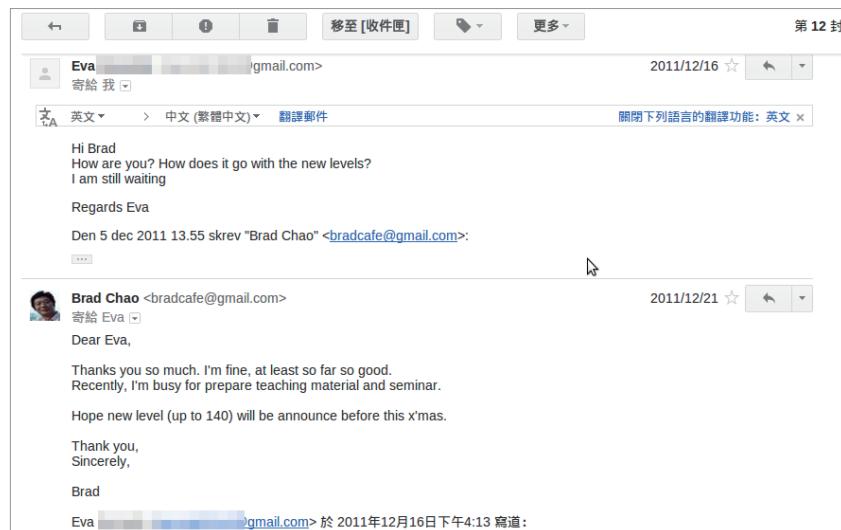
這是一位來自瑞典的玩家，不斷要求增加關卡 ...



我增加設計更多關卡，但是似乎仍無法滿足 ...



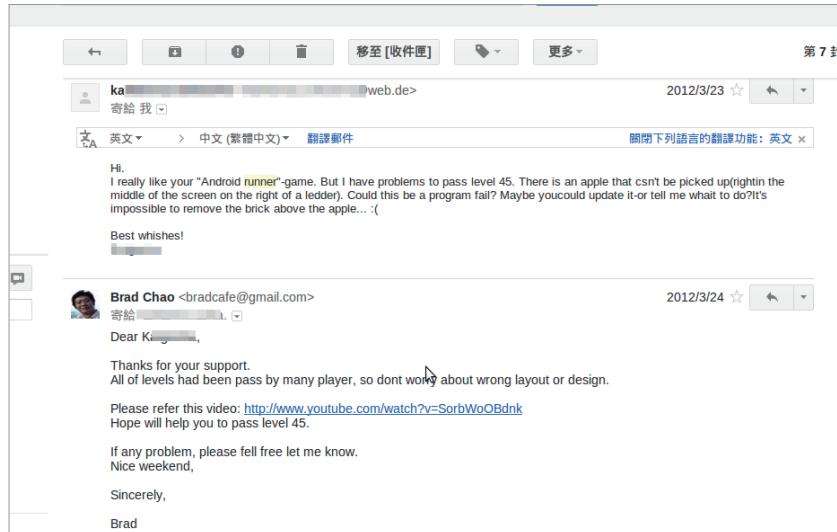
最後只好承諾會在聖誕節前推出新關卡，目前的 160 關都是被這位玩家給逼出來的（感覺好像被出版社催稿）。



這又是另外一位來自德國的玩家，幫我設計關卡，非常甘心。



這位玩家真的玩不出關卡，要來求救密技，我只好拍成 youtube 紹他參考囉。



16

太多來自世界各地玩家享受這款遊戲，即使是免費，至少人生中有這樣的際遇，就讓 Brad 覺得相當值得，不是用金錢可以換來的價值。

而每年台灣都有許多相關的比賽可以參加。Brad 本來不想參與其中，但是曾經有位學員半開玩笑的說：「老師，你自己也沒有參加過比賽呀！」OK，就在出版一本 Android 書期間，最直接的示範動作，就是將授課範例，參加當年 2010 年間的「創意成金」活動，共有四個範例參加，被選中了三個範例：「猜數字遊戲」、「我的待辦事項」及「旅遊全紀錄」，也得到一筆獎金。當年原本是將猜數字遊戲範例放進上一本書中介紹，基於活動規則，還特定將公開發表的書中範例變更為猜數字大小遊戲，而不是 1A2B 的猜數字遊戲。如此一來，學生就應該比較能夠信服，所以後來陸續參加的比賽，都是為了增加授課及輔導的實戰經驗。

參加比賽可以督促開發者將一個作品進行完整化的包裝，而不是一天拖過一天的開發。

最重要的是，參加比賽千萬不要得失心太重，參賽的過程與經驗的累積，才是重要的實質意義。

MEMO
