美化App

講師 劉治廷

自製外觀

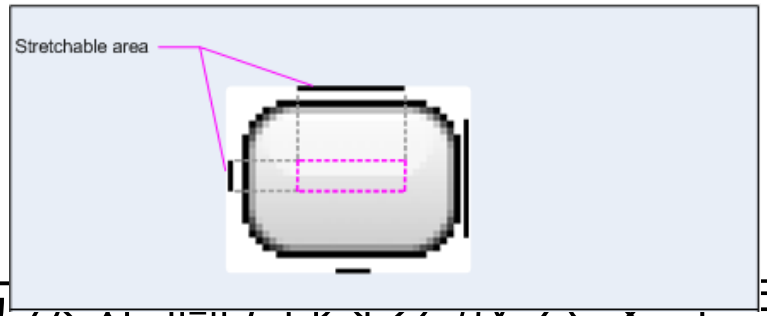
- 搭配專案：9PatchTrying

使用9-patch

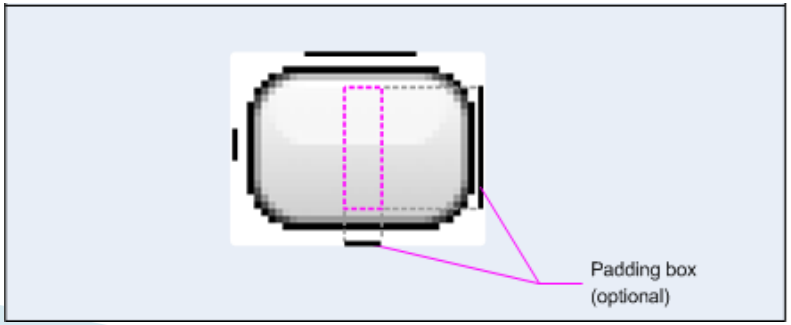
- android特殊圖片格式
- 可以讓圖片在不失真且不失去原有比例的狀況下，放大圖片
- 適用於各種View的背景或是App的背景
- 9-patch是藉由圖片四周的1個像素來設定縮放和擺放內容物的旗標

9-patch說明

- 左方和上方決定圖片縮放時可以縮放的範圍



- 右方和下方決定圖片內容要放入文字或其它圖片時(例如圖片按鈕)，可以擺放的區域



9-patch說明

- 所以當放入大小不同、長度不同的文字時
- 就會維持圖片要呈現的效果

9-patch實際範例

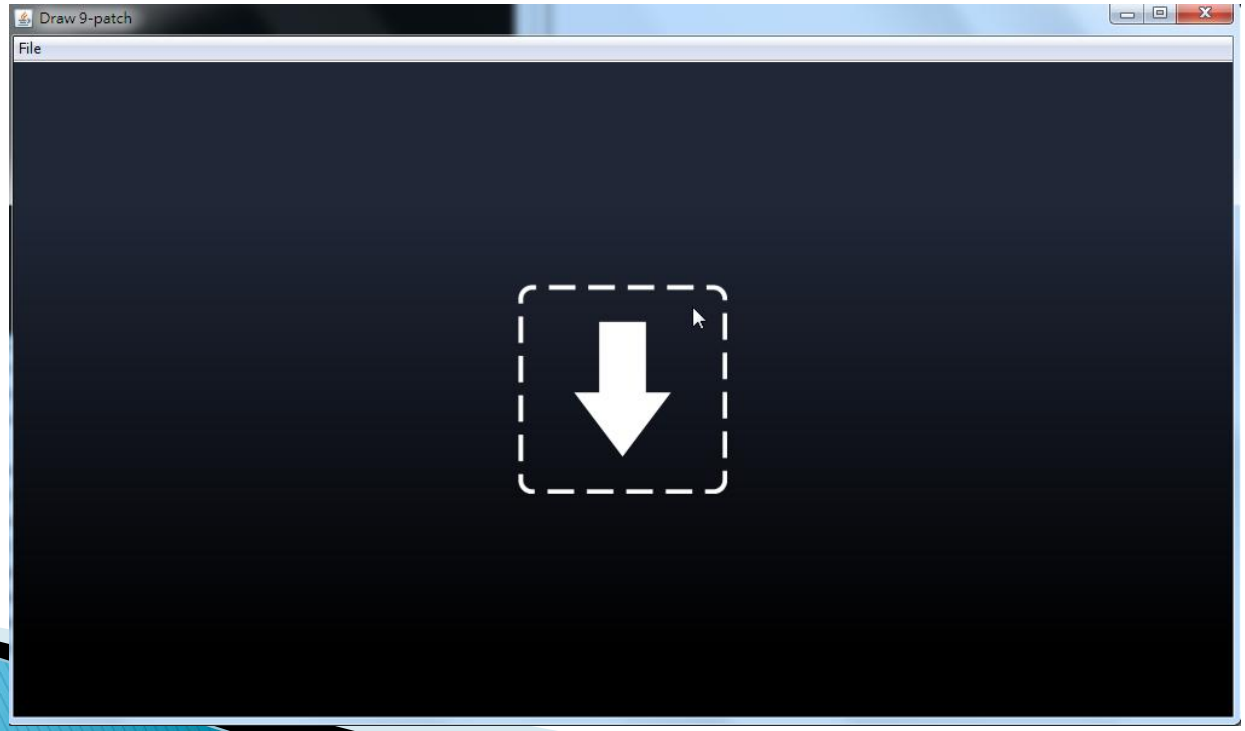
- 如何做出以下兩個按鈕？

9-patch實際範例

- 步驟一：製作背景圖片
  - 首先可以用繪圖軟體(Photoshop, GIMP)來製作出小的背景圖片，並存檔
  - 記得，可以把圖做小，因為9patch可以自動縮放圖片

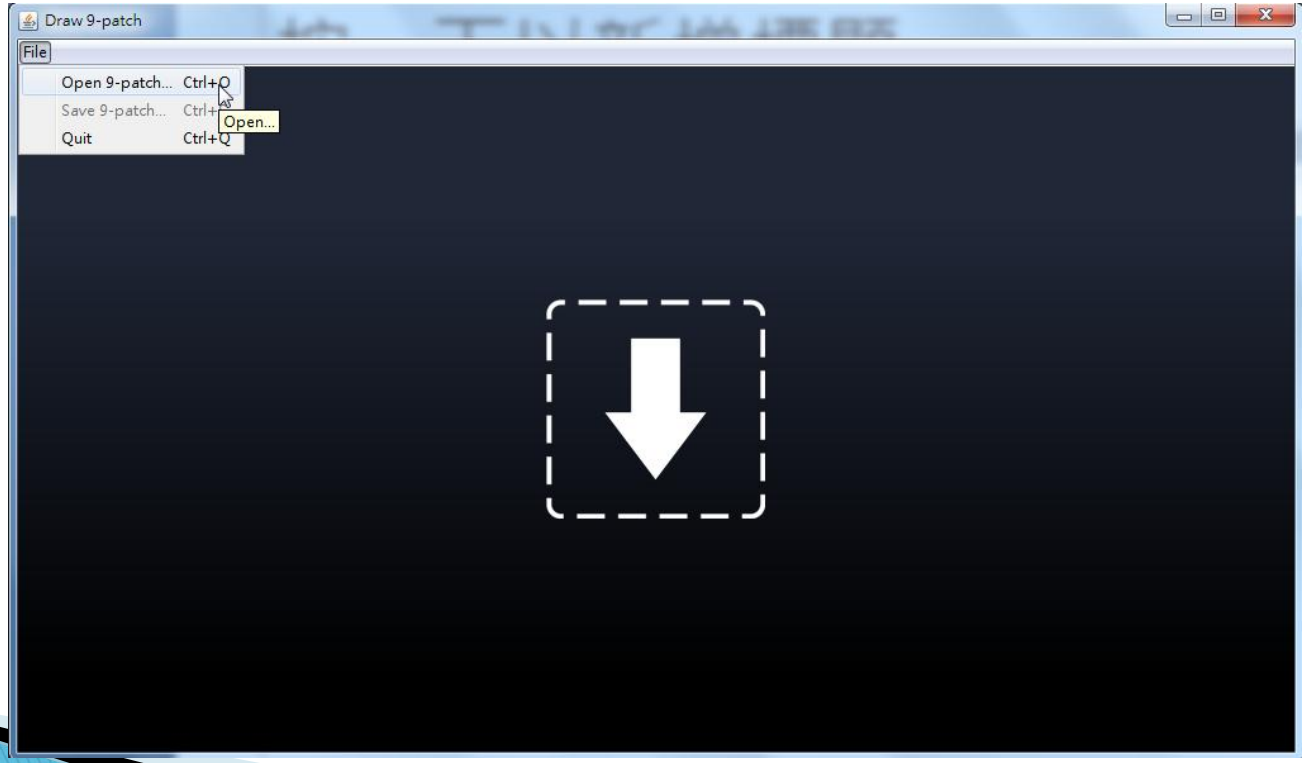
9-patch實際範例

- 步驟二：打開9patch製作工具
  - 打開工具<android_sdk>/tools/draw9patch.bat



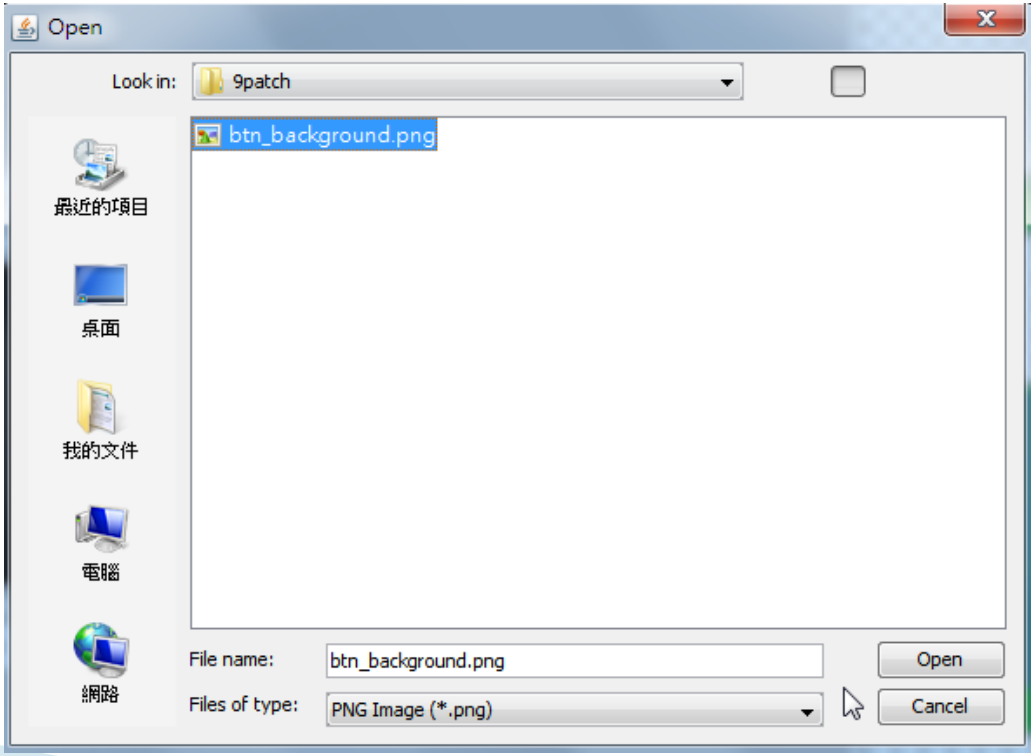
9-patch實際範例

- 步驟三：打開檔案
  - 選擇右上角File → Open 9-patch



9-patch實際範例

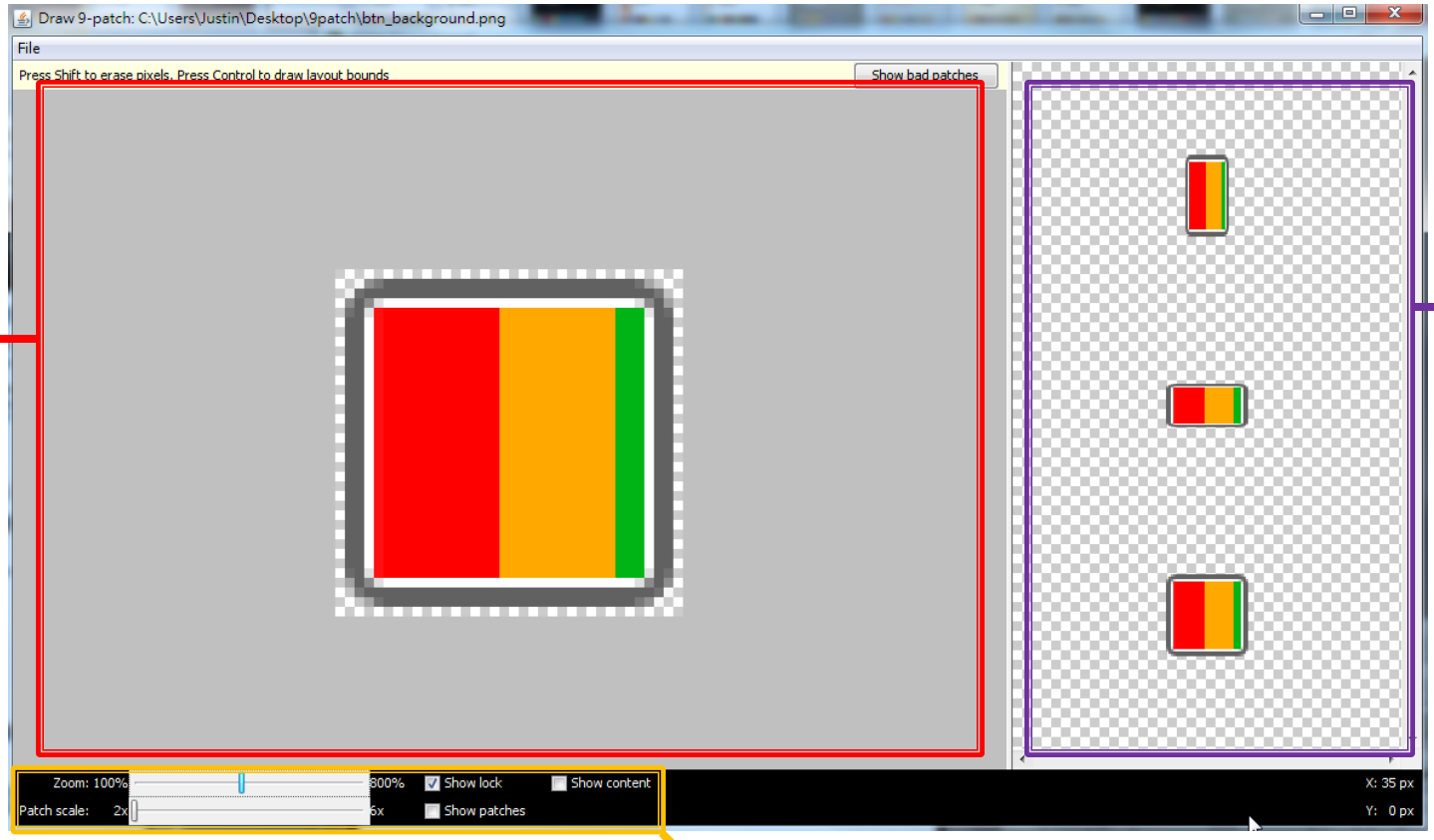
- 步驟三：打開檔案
  - 選擇在步驟一製作的檔案



9-patch實際範例

- 步驟四：工具使用說明
  - 打開後會見到如下圖的介面

9-patch實際範例



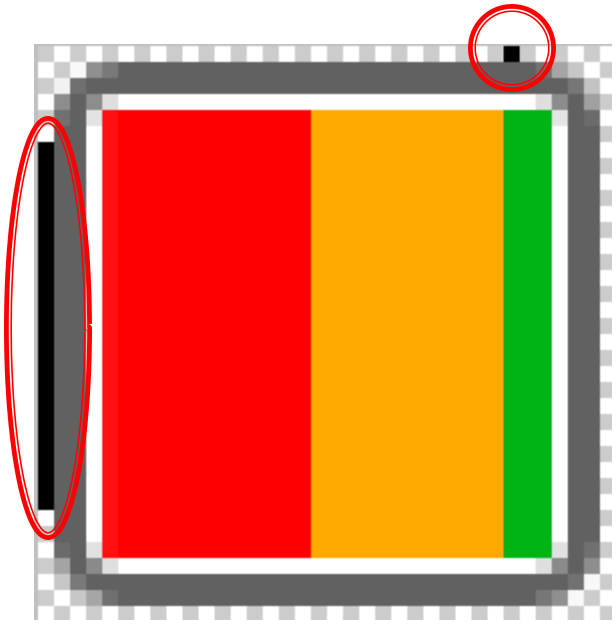
編輯區

預覽區

縮放控制項

9-patch實際範例

- 步驟五：繪製縮放的旗標
  - 在「編輯區」圖片上方和左方空白的一個像素處，點出如下圖的黑線

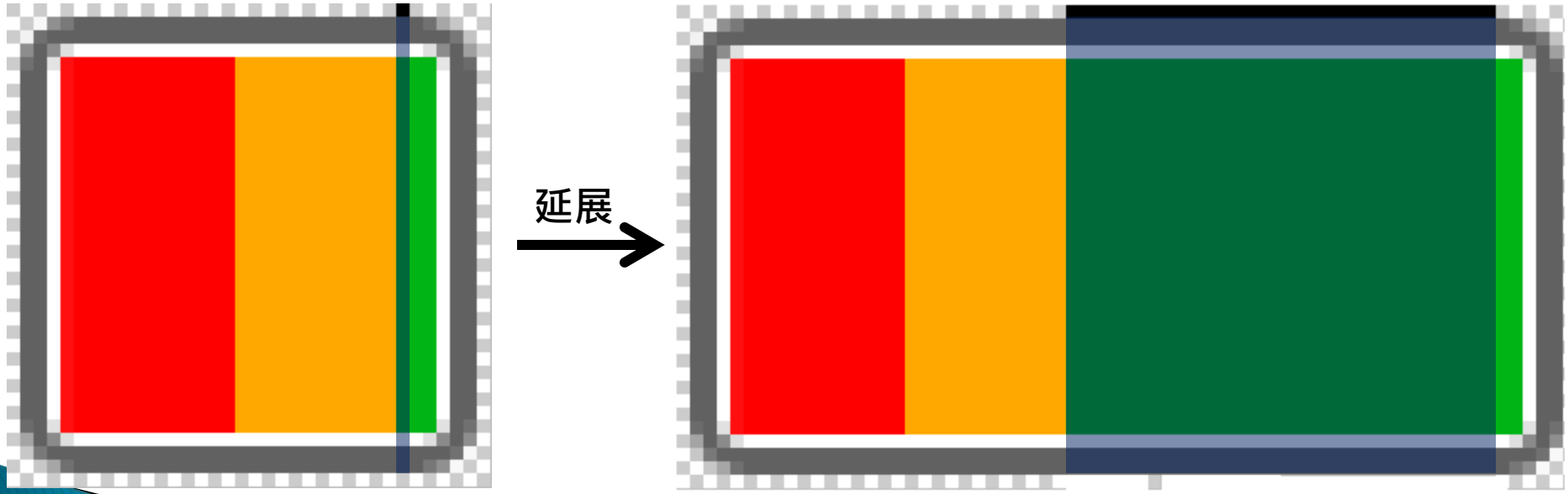


9-patch實際範例

- 步驟五：繪製縮放的旗標
  - 點選時可以同時發現右方的預覽區也在改變

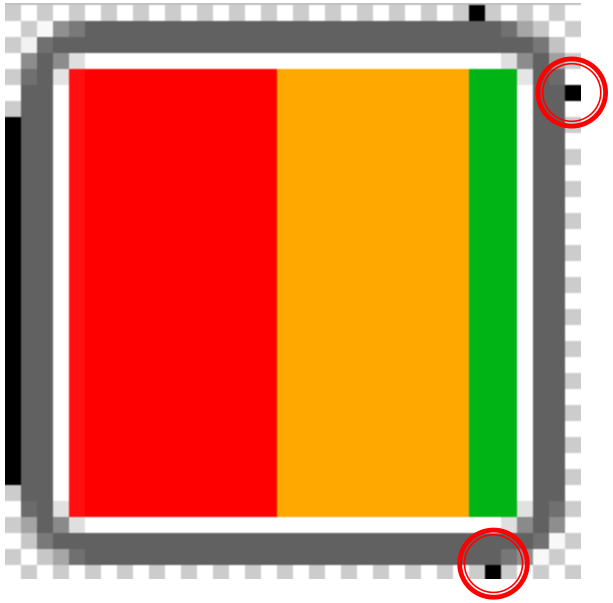
9-patch實際範例

- 步驟五：繪製縮放的旗標
  - 上方的點表示當圖片要橫向延展時，會以上方的黑線所圈選的範圍為延展基礎



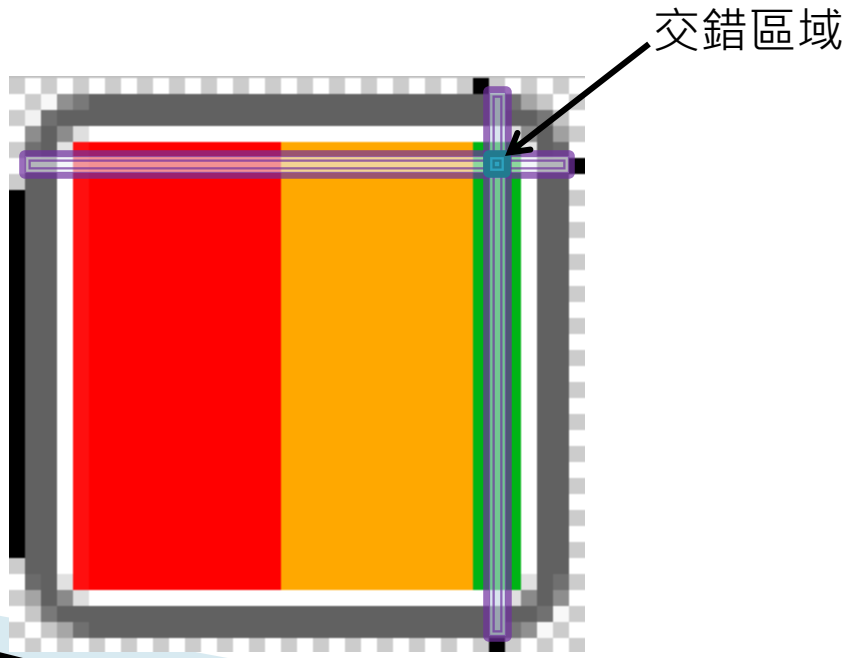
9-patch實際範例

- 步驟六：繪製擺放文字或圖片的位置
  - 在「編輯區」圖片下方和右方空白的一個像素處，點出如下圖的黑線



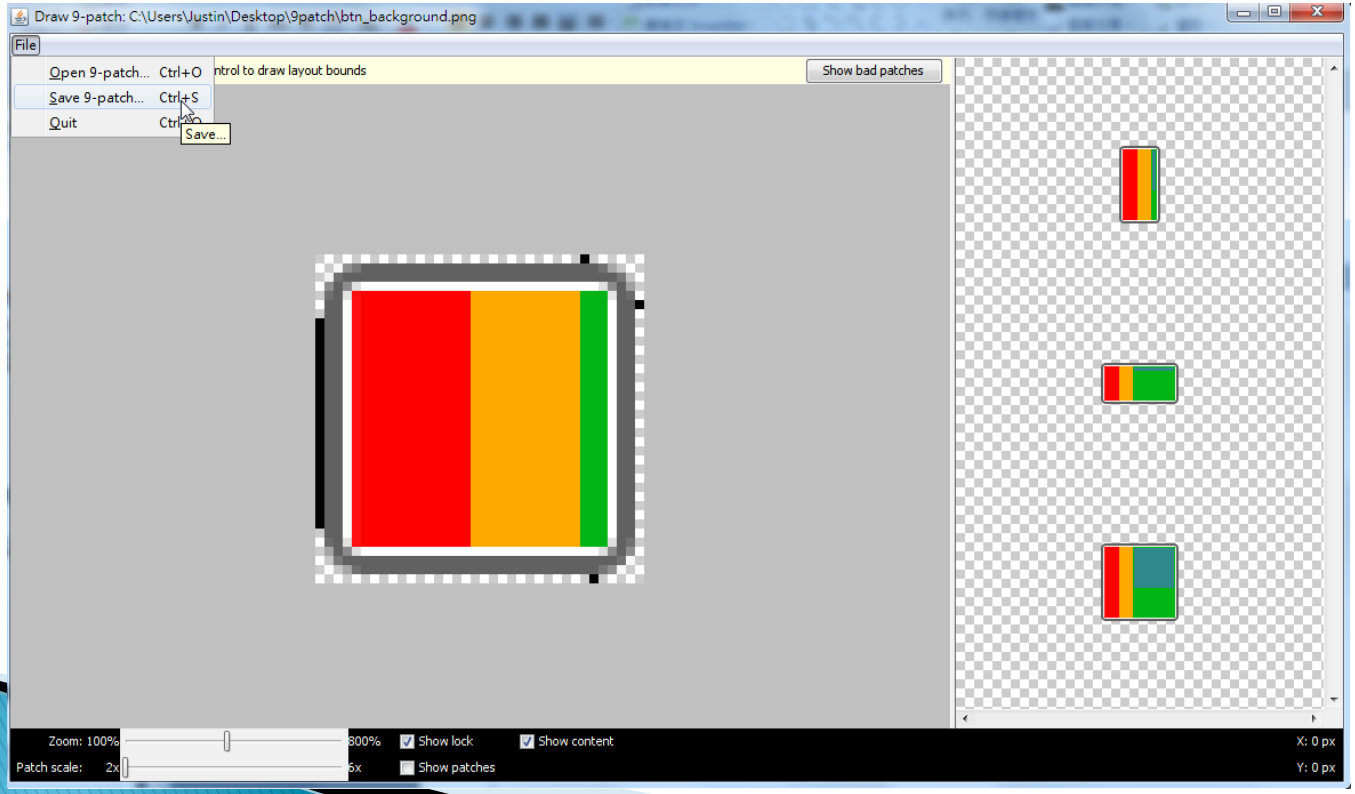
9-patch實際範例

- 步驟六：繪製擺放文字或圖片的位置
  - 右方和下方兩黑線交錯的區域是文字或圖片擺放的區域
  - 當擺放的文字或圖片大於這個區域時，將會放大該區域



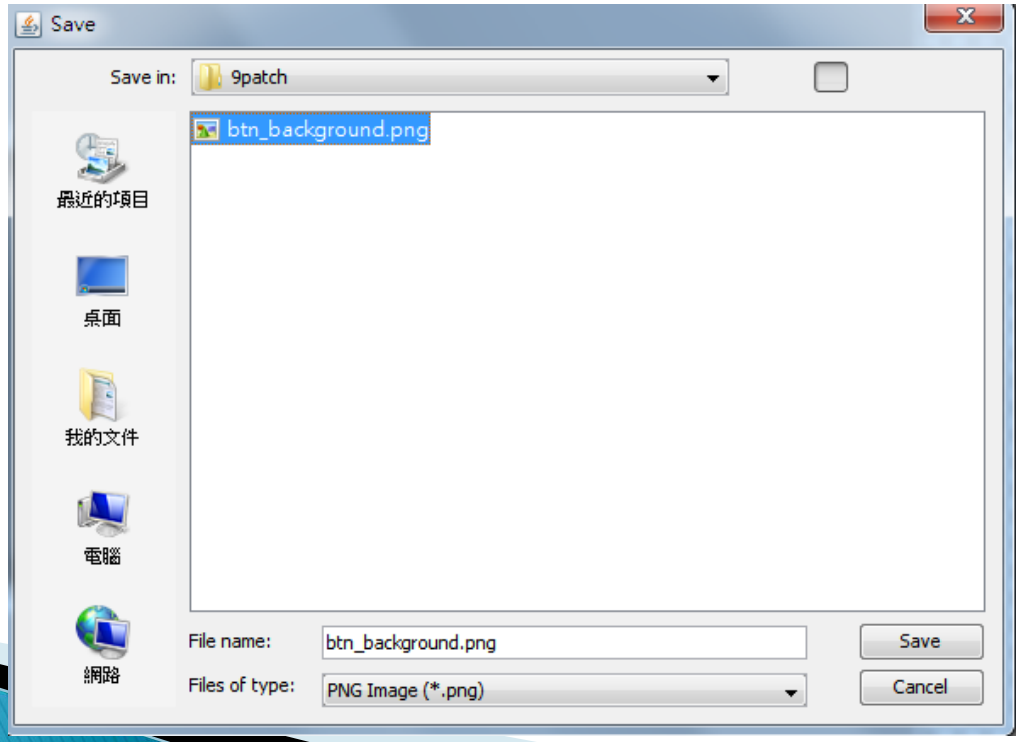
9-patch實際範例

- 步驟七：存檔
  - 點選左上角File→ Save 9-patch…



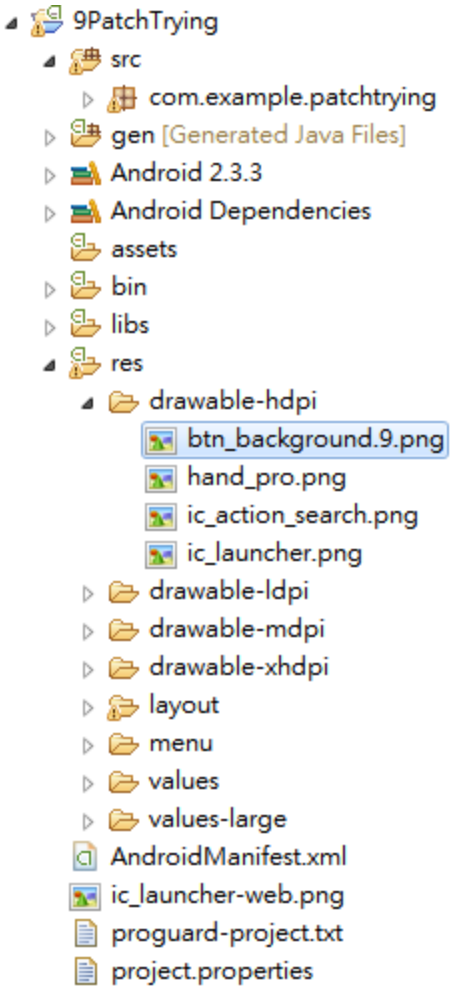
9-patch實際範例

- 步驟七：存檔
  - 在File name輸入儲存的檔案名稱
  - 存檔的檔案副檔名會自動變為「.9.png」



9-patch實際範例

- 步驟八：並置放於Eclipse專案中的資料夾drawable-hdpi (其他解析度也可以)
  - 複製.9.png那個檔案到專案中



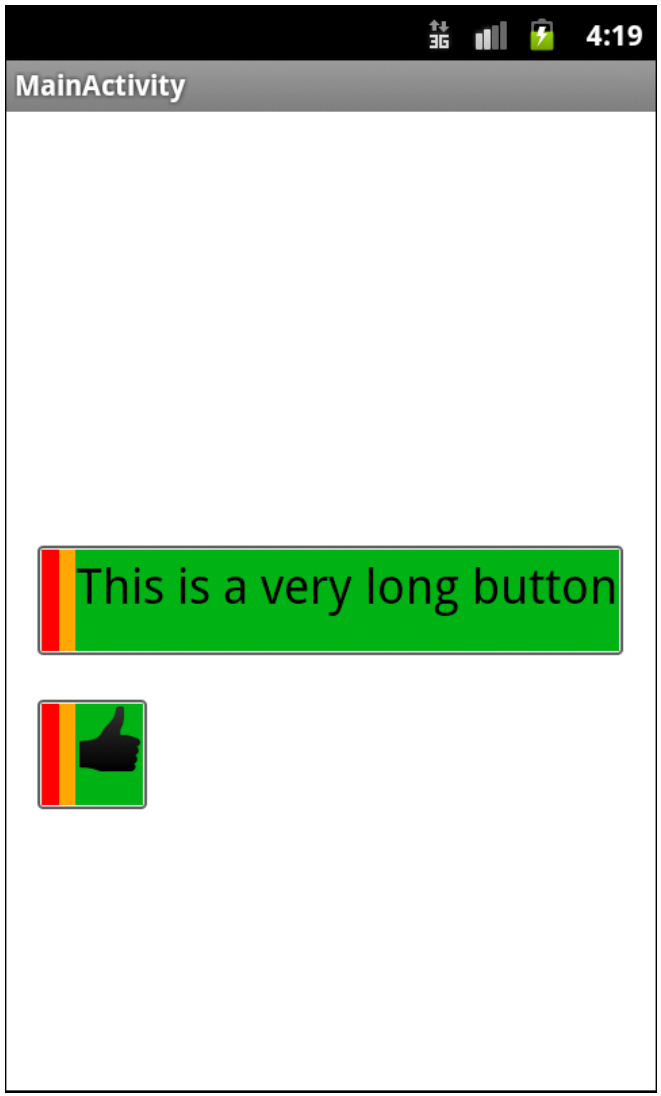
9-patch實際範例

- 步驟九：設定9patch圖片為按鈕背景

```
<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" >

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:textSize="24sp"
        android:background="@drawable/btn_background"
        android:text="This is a very long button" />

    <ImageButton
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/button1"
        android:layout_below="@+id/button1"
        android:layout_marginTop="22dp"
        android:src="@drawable/hand_pro"
        android:background="@drawable/btn_background" />

</RelativeLayout>
```

9-patch實際範例

- 步驟10：觀看結果



總結

- 應用程式啟動前會將drawable先讀取進記憶體，使用9patch因為檔案小，可以加快應用程式啟動速度
- 使用9patch可以大大降低App本身的大小，可以減少下載時所需花的費用，且可增加電池使用時間

問題

- 當這次的應用程式裝到手機或模擬器上後，點選按鈕卻沒有任何狀態的改變？
- 解決方案就在下一章

物件狀態與呈現效果

- 搭配專案：CustomButton

物件狀態

- Android中可以藉由XML組合成具有「狀態」的圖片
- View可自動依照狀況改變呈現圖片
- 常用state
  - state_focused
  - state_pressed
  - state_enable

按鈕例子

說明	平時狀態	以trackball選擇按鈕	以手指觸控按鈕	按鈕取消點選功能	按鈕取消點選功能時以trackball選擇按鈕
state_focused	false	true	false	false	true
state_pressed	false	false	true	false	false
state_enable	true	true	true	false	false

物件狀態檔案

- 位於<android_sdk>\platforms\android-<版本>\data\res\drawable\<檔案名稱>.xml

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_window_focused="false" android:state_enabled="true"
        android:drawable="@drawable/btn_default_normal" />
    <item android:state_window_focused="false" android:state_enabled="false"
        android:drawable="@drawable/btn_default_normal_disable" />
    <item android:state_pressed="true"
        android:drawable="@drawable/btn_default_pressed" />
    <item android:state_focused="true" android:state_enabled="true"
        android:drawable="@drawable/btn_default_selected" />
    <item android:state_enabled="true"
        android:drawable="@drawable/btn_default_normal" />
    <item android:state_focused="true"
        android:drawable="@drawable/btn_default_normal_disable_focused" />
    <item
        android:drawable="@drawable/btn_default_normal_disable" />
</selector>
```

對應方式

drawable/btn_default.xml

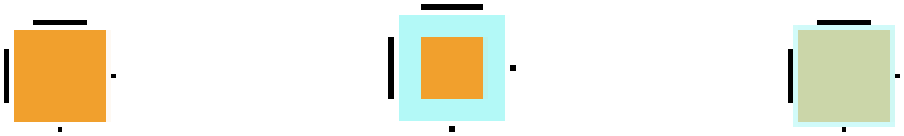
```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_window_focused="false" android:state_enabled="true"
        android:drawable="@drawable/btn_default_normal" />
    <item android:state_window_focused="false" android:state_enabled="false"
        android:drawable="@drawable/btn_default_normal_disable" />
    <item android:state_pressed="true"
        android:drawable="@drawable/btn_default_pressed" />
```

drawable-<?dpi>/
btn_default_normal.9.png

drawable-<?dpi>/
btn_default_pressed.9.png

drawable-<?dpi>/
btn_default_normal_disable.9.png

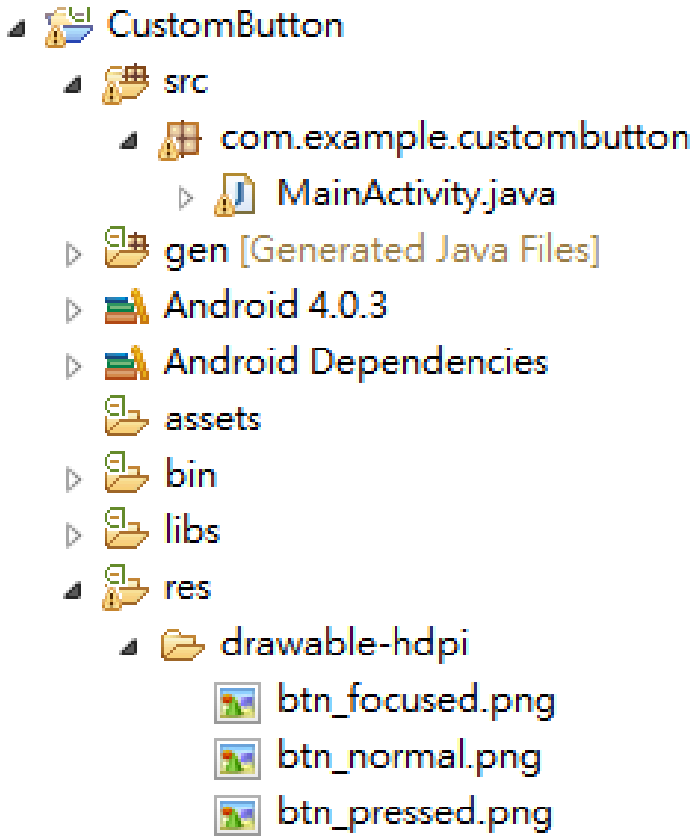
自訂狀態範例



說明	平時狀態	以track ball選擇按鈕	以手指觸控按鈕
state_focused	false	true	false
state_pressed	false	false	true
state_enable	true	true	true

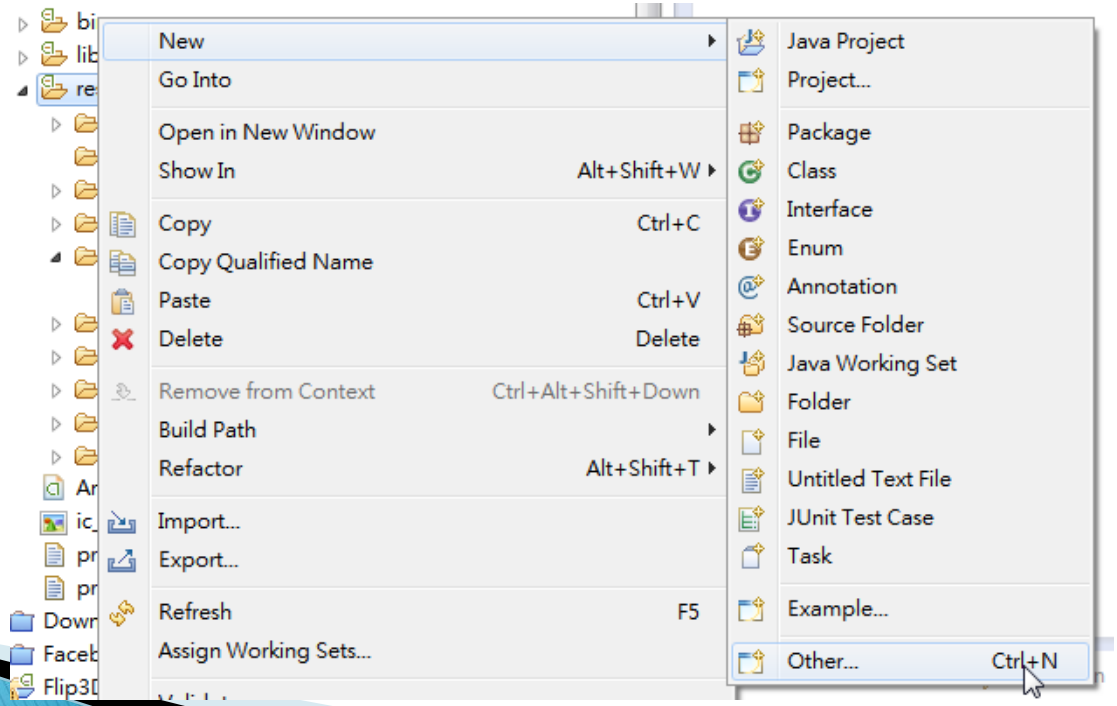
自訂狀態範例

- 步驟一：將圖片複製到Eclipse中專案的資料夾



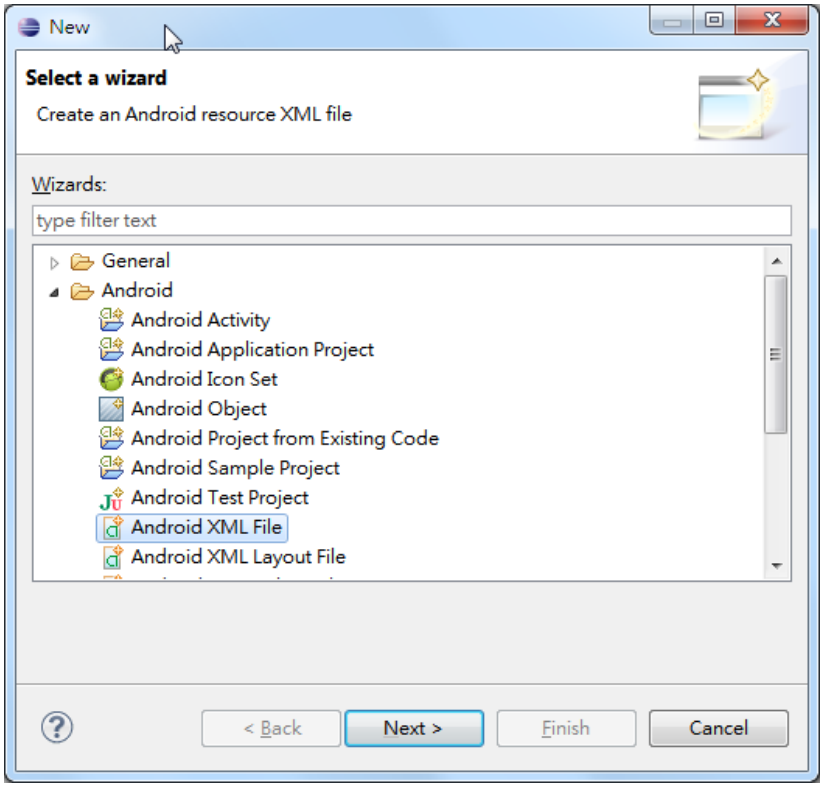
自訂狀態範例

- 步驟二：新增自訂的drawable
  - 對著專案的res資料夾按下滑鼠右鍵
  - 選擇「New」→「Other…」



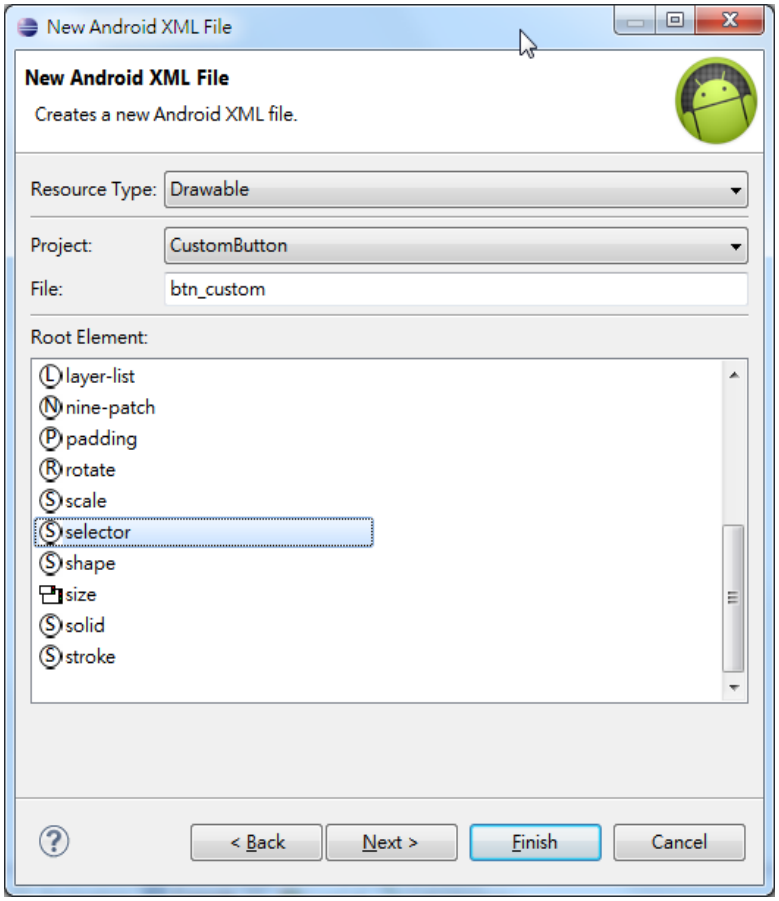
自訂狀態範例

- 步驟二：新增自訂的drawable
  - 選擇「Android」中的「Android XML File」
  - 按下「Next」



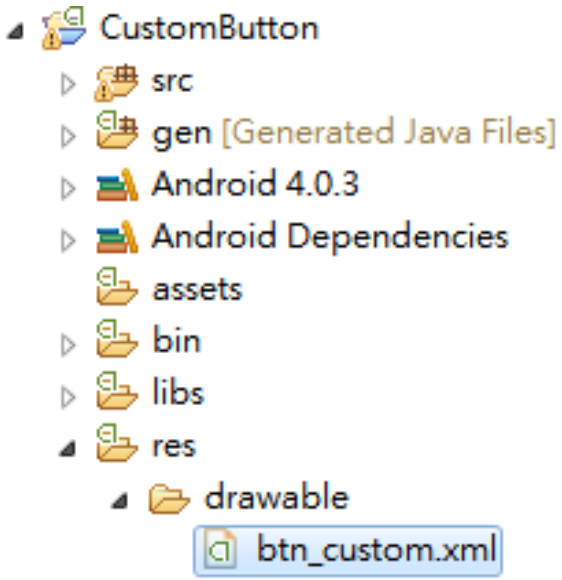
自訂狀態範例

- 步驟二：新增自訂的drawable
  - Resource Type選擇「Drawable」
  - File輸入「btn_custom」
  - Root Element選擇「selector」
  - 按下「Finish」



自訂狀態範例

- 步驟二：新增自訂的drawable
  - 在專案資料夾res中就會出現一個新的資料夾drawable



自訂狀態範例

- 步驟三：打開btn_custom.xml檔案
  - 打開後會出現如下方的內容

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


</selector>
```

自訂狀態範例

- 步驟四：設定狀態及對應drawable
  - 每一個狀態都是一個<item>
  - 每個state都是android:state_xxx=
  - 對應的drawable使用android:drawable=
  - 最後的item沒有狀態只有drawable表示一般狀態

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:state_focused="true"
        android:state_enabled="true"
        android:drawable="@drawable/btn_focused"/>
    <item
        android:state_pressed="true"
        android:state_enabled="true"
        android:drawable="@drawable/btn_pressed"/>
    <item
        android:drawable="@drawable/btn_normal"/>
</selector>
```

自訂狀態範例

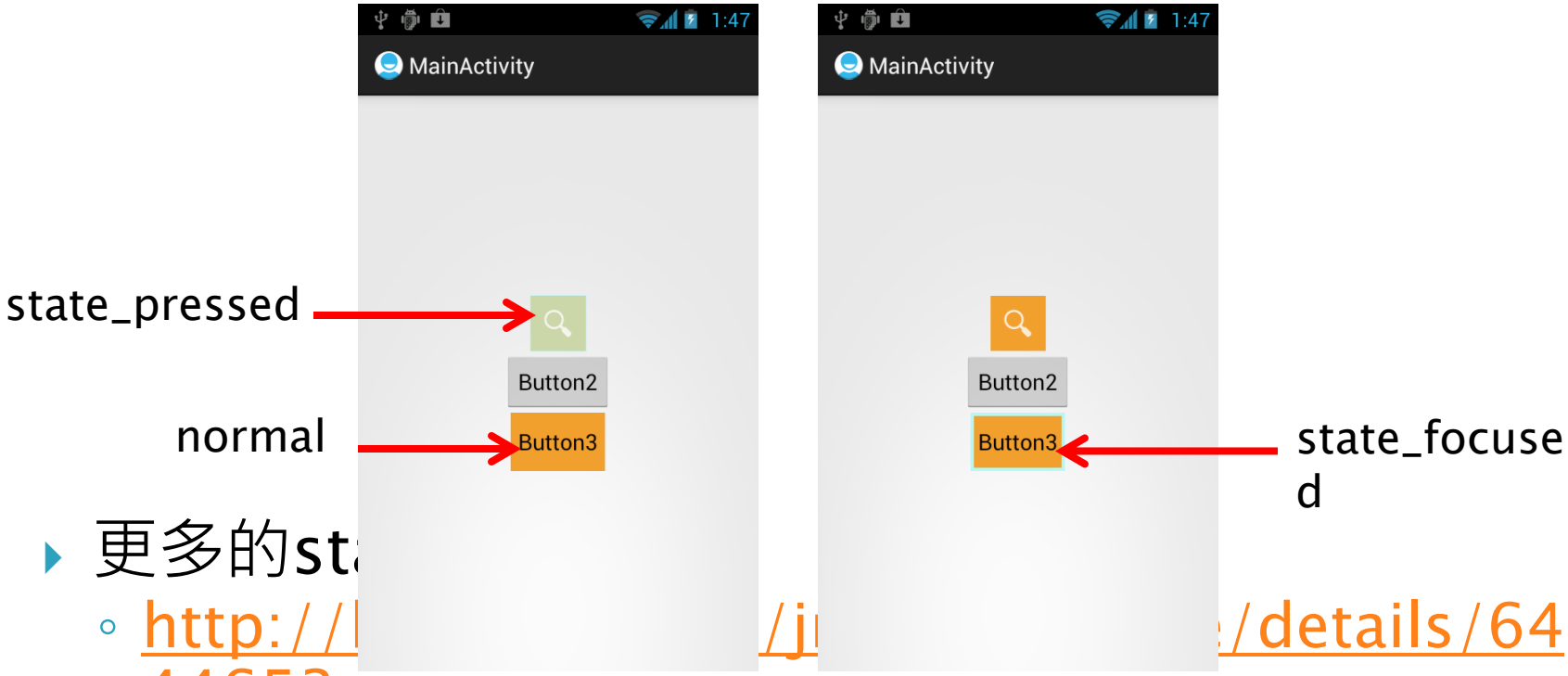
- 步驟五：設定Button
  - 可將Button的背景為btn_custom
  - 也可以將ImageButton的前景圖設為btn_custom，背景圖設定為@null

```
<Button
    android:id="@+id/button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/button2"
    android:layout_centerHorizontal="true"
    android:textStyle="bold"
    android:textSize="24sp"
    android:background="@drawable/btn_custom" />
```

```
<ImageButton
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:src="@drawable/btn_custom"
    android:background="@null"/>
```

自訂狀態範例

- 步驟六：執行範例



- 更多的st
  - http://l /j /details/6444653

問題

- 如果要保持App介面美觀，必須保持「一致性」所以一個按鈕改變時，其他按鈕也要一起改變
- 假設我的App是很多頁面合在一起，其中的按鈕無數，那要一個一個更改background嗎？
- 如果要動態改變介面的話，如何做到？
- 解決方案：下一章

Style的定義

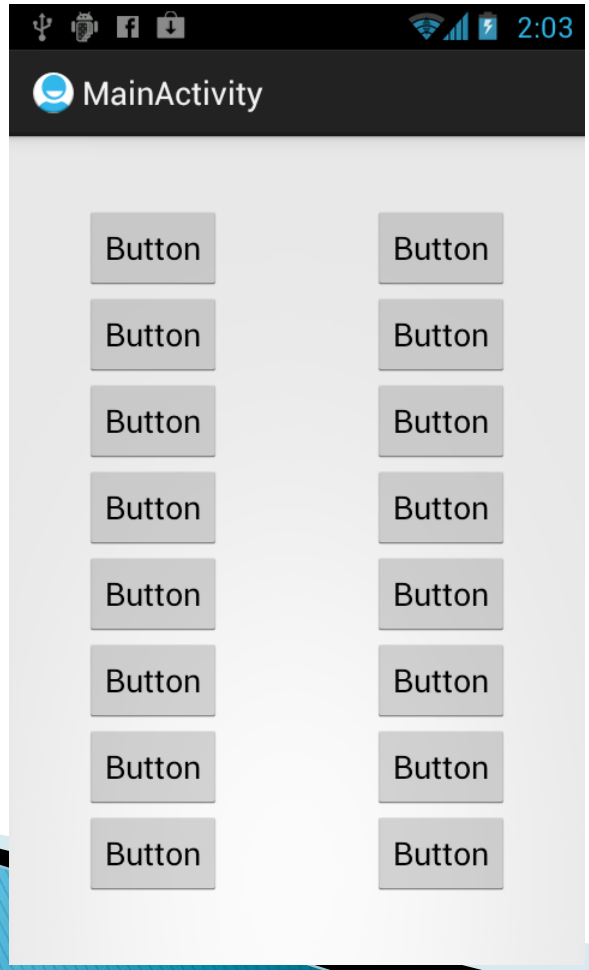
- 搭配專案：CustomStyle

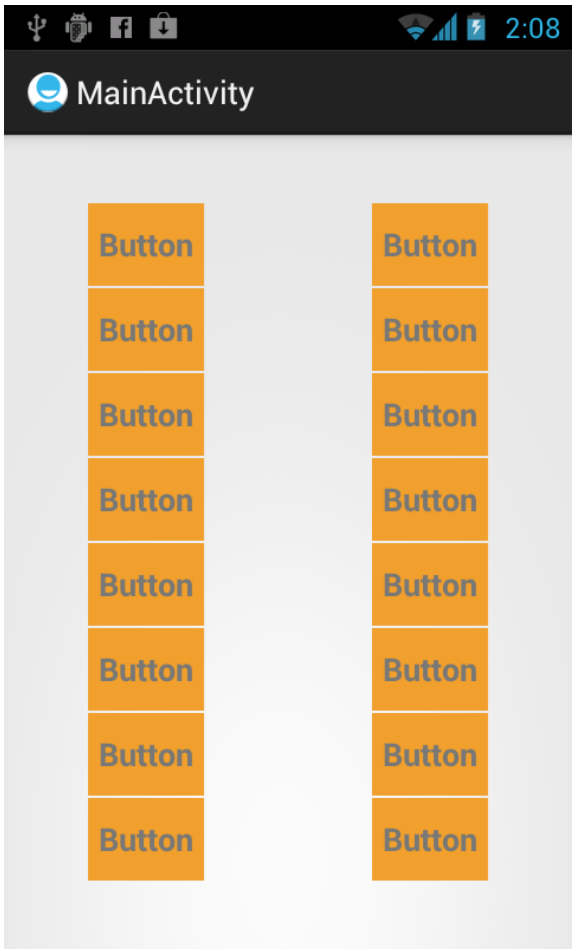
Style

- Style是包含一種或多種的物件格式的集合
- 可以取出相類似的屬性以便部屬在應用程式中
- 屬於資源檔
- 擺放於專案的res/values資料夾下
- Android內的預設Style擺放位置
  - **<android_sdk>\platforms\android-<版本號>\data\res\values\styles.xml**

客製化Style

- 若要將畫面由左邊變為右邊該如何處理？





客製化Style

- 當然可以自己慢慢修改每一個<Button>的屬性

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/btn_custom"
    android:textSize="18sp"
    android:textColor="#787878"
    android:textStyle="bold"
    android:text="Button" />
```

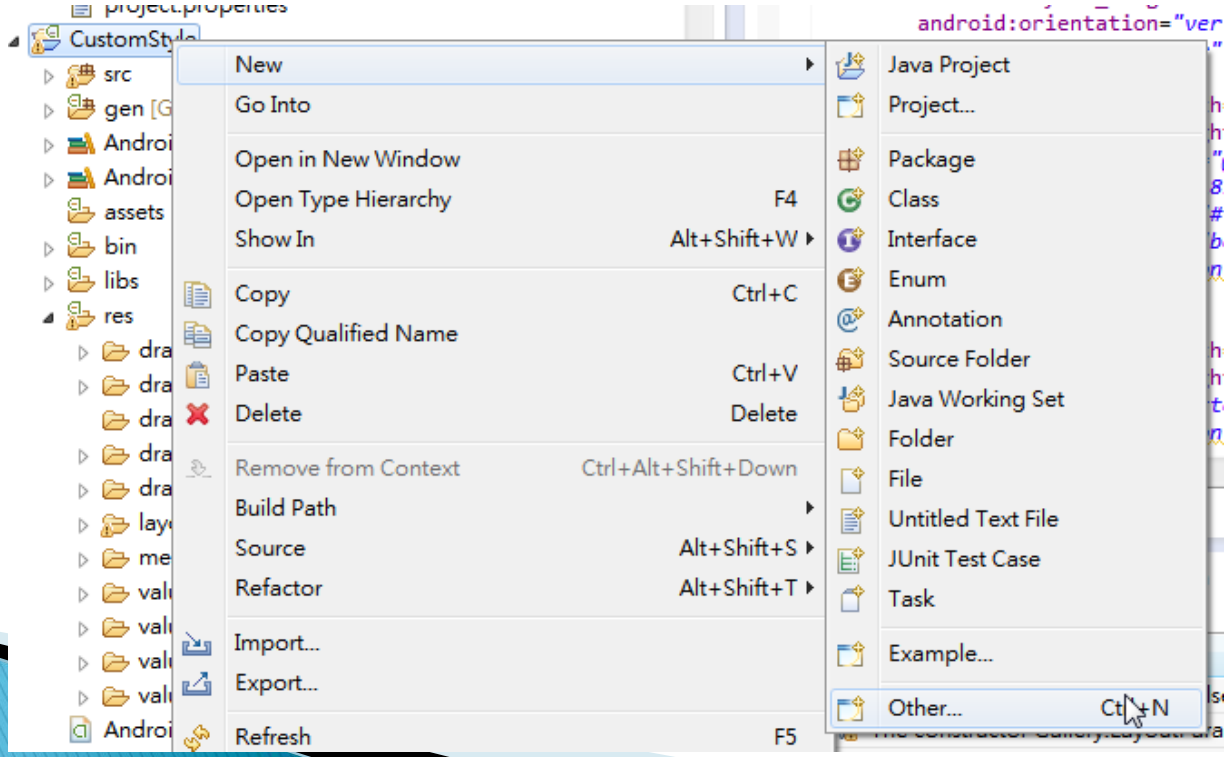
- 但每次若要改變介面的外觀，將要不停的修改很多地方
- 所以要善用Style的功能

客製化Style

- 步驟
  1. 新增style
     - 在res/values之中
  2. 設定style的內容
  3. 附加在要套用的layout中

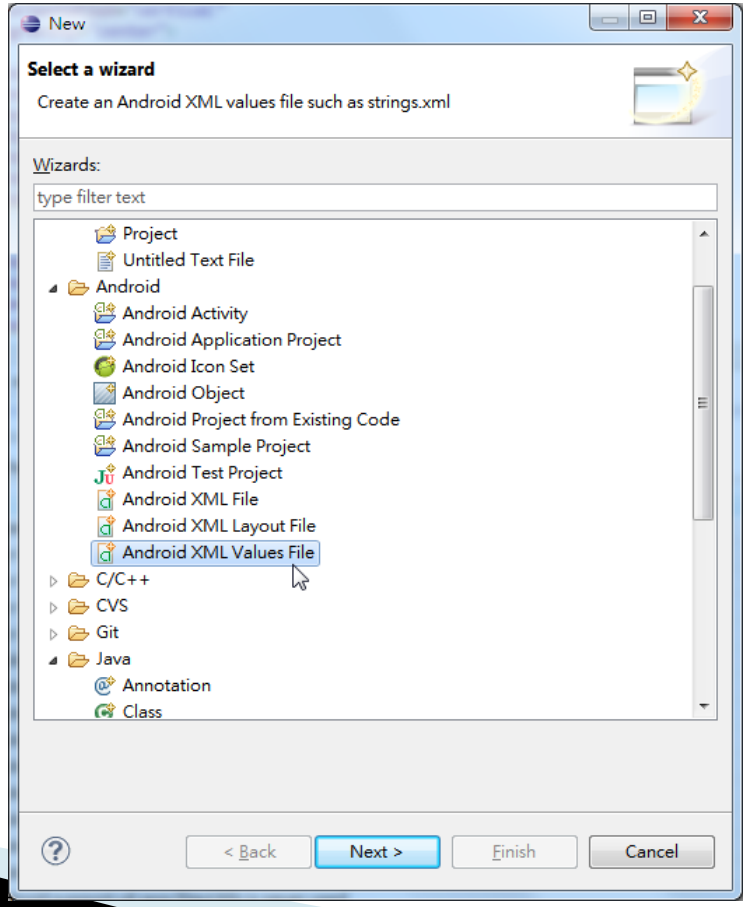
步驟一：新增Style

- 若專案res/values資料夾下已經有style.xml可以省略此步驟
- 對專案按滑鼠右鍵，選擇「New」→「Others」



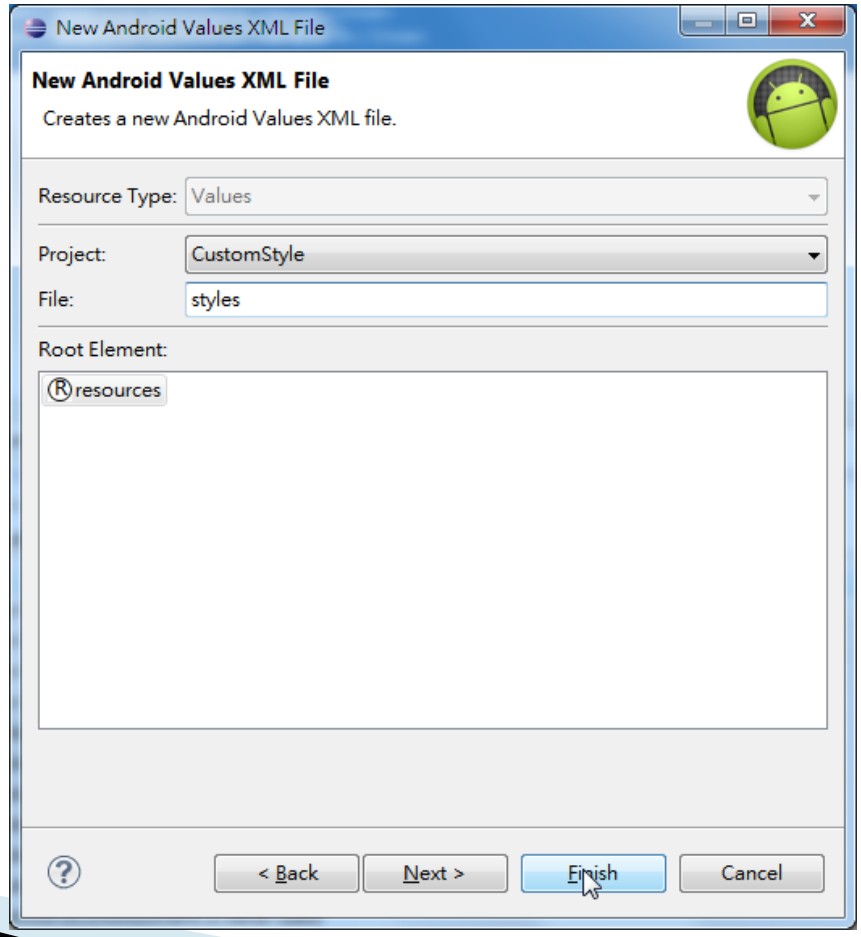
步驟一：新增Style

- 選擇「Android XML Values File」



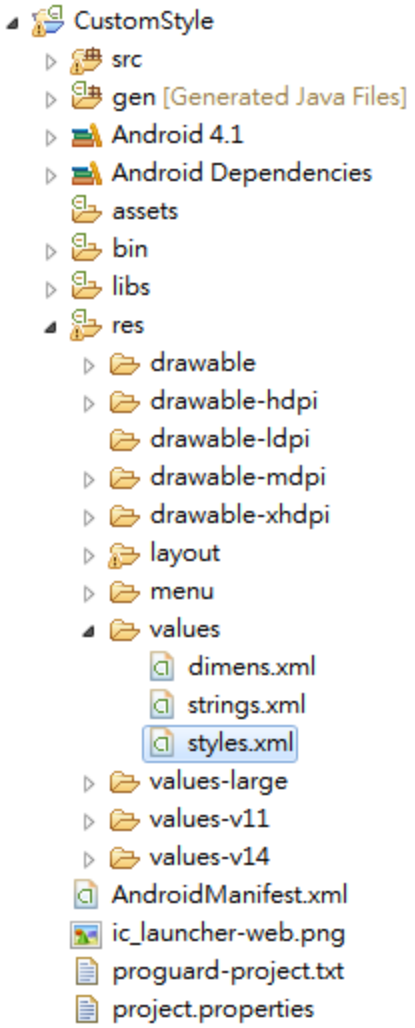
步驟一：新增Style

- File輸入style，當然也可以取別的名稱
- 按下Finish



步驟一：新增Style

- 確認專案res/values資料夾下出現styles.xml



步驟二：設定style的內容

- 打開styles.xml檔案
- 也許有人會看到以下內容 (依照ADT版本會有所不同)
- 這是專案建立時自動產生的佈景主題，沒有也沒關係

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

    <style name="AppTheme" parent="android:Theme.Light" />

</resources>
```

步驟二：設定style的內容

- 新增以下內容在其中

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

    <style name="AppTheme" parent="android:Theme.Light" />

    <style name="MyButtonStyle">
        <item name="android:background">@drawable/btn_custom</item>
        <item name="android:textSize">18sp</item>
        <item name="android:textStyle">bold</item>
        <item name="android:textColor">#787878</item>
    </style>

</resources>
```

步驟三：附加在要套用的layout中

- 打開Activity使用的layout XML檔案
- 將其中所有<Button>的屬性多加一個
  - style=“@style/MyButtonStyle”
  - 不要寫成~~android:style=“@style/MyButton…”~~

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/btn_custom"
    style="@style/MyButtonStyle"
    android:text="Button" />
```

結論

- 所以如果之後要客制化Button成為不同的背景、不同的字型大小或樣式，只要對Style修改即可，不用再打開layout檔案一個一個修改了
- 善用Style可以讓App變的畫面一致化，更容易達到良好的「使用者經驗」(User Experience)

Theme

Theme定義

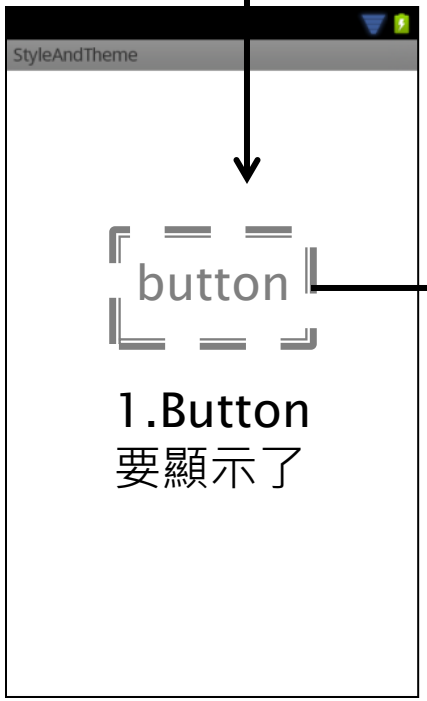
- Theme是佈景主題，包含了一種或多種的Style，且是針對某一特定的Activity進行配置
- Android中我們一般使用的Widget，如TextView、Button、EditText都有其Style
- 藉由Android內部的Theme將Widget的Style整合起來

Theme定義

- Theme也是資源檔，位在專案res/values資料夾下
- Android內部的Theme定義可在以下資料夾中找到
  - **<android_sdk>\platforms\android-<版本號>\data\res\values\styles.xml**
  - 關於Android內部Theme的說明檔
    - **<android_sdk>\platforms\android-<版本號>\data\res\attrs.xml**

Theme定義

在AndroidManifest.xml中定義此Activity使用的佈景主題為
@android:style/Theme



1.Button
要顯示了

2. Button
要長什麼樣？

themes.xml

```
<style name="Theme">
    <item name="buttonStyle">
```

3.在theme.xml中
依據
AndroidManifest.
xml對
Activity定義的
Theme找到對應的
XML區塊，於區塊
再找出Widget style

styles.xml

```
<style name="Widget.Button">
    <item name="android:background">
```

4. 依據3中找到的
style於style.xml
中定義了物件細節

App的Theme設定

- 在AndroidManifest.xml中可以設定佈景主題
  - 使用專案自定的佈景主題
    - android:theme="@style/…"

```
<application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
```

  - 使用Android預設的佈景主題
    - android:theme="@android:style/Theme.XXX"

```
<application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.Light" >
```

  - AndroidManifest.xml沒有定義，系統自動使用預設值

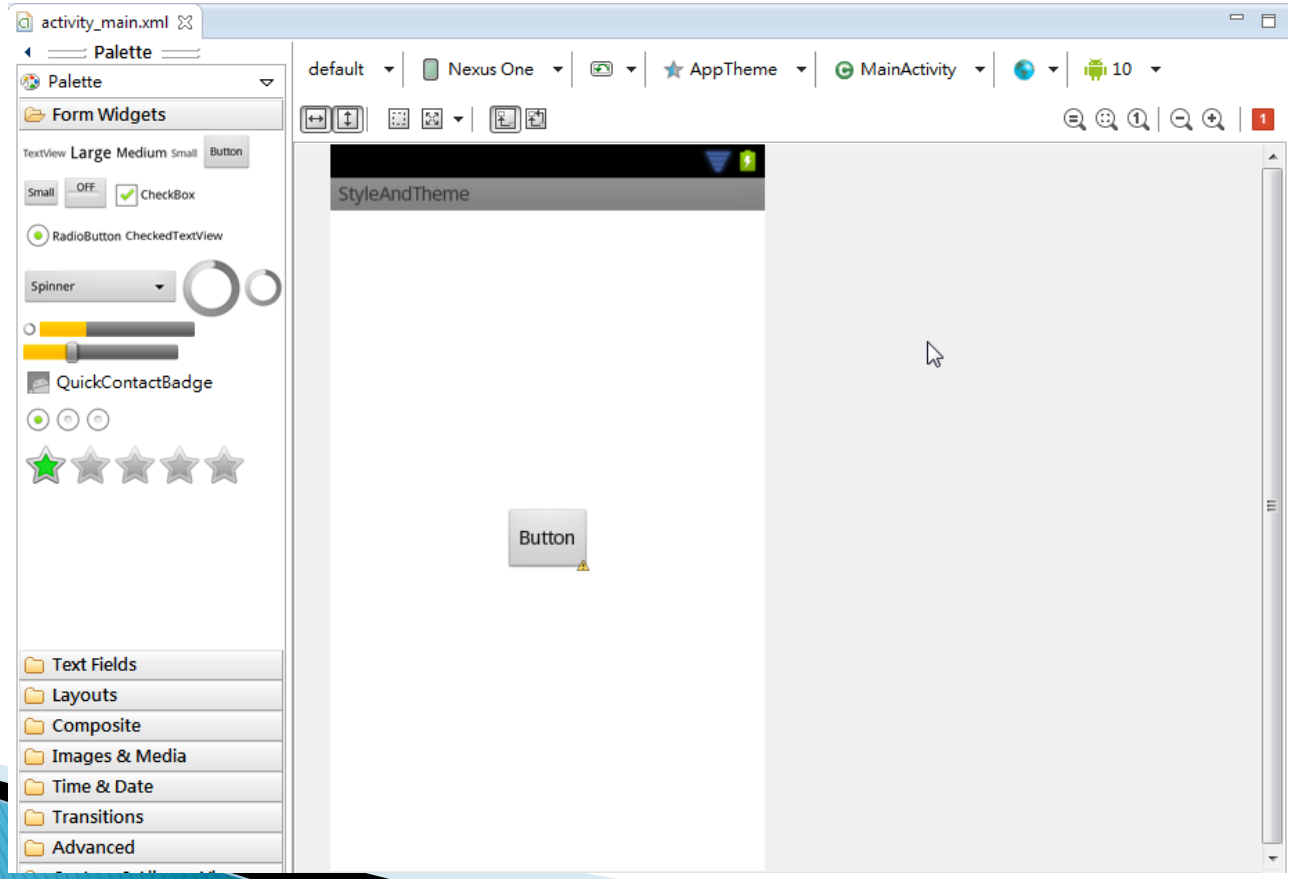
預覽App的Theme

如何預覽Theme

- 這裡分成兩個部份，舊專案 (ADT-19以前建立的專案)和新專案(ADT-20以上)
- 首先針對新專案來說明
- 步驟一：更新ADT至版本20以上 (建議最新版)
- 步驟二：新增一個Android專案

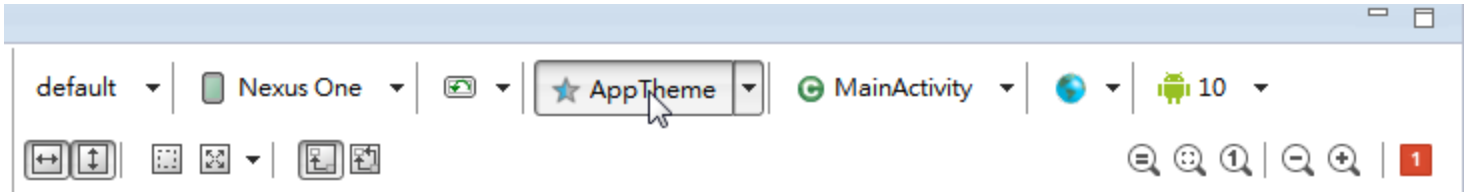
如何預覽Theme

- 步驟三：打開任何一個Layout檔案 (位在專案 res/layout/之下)



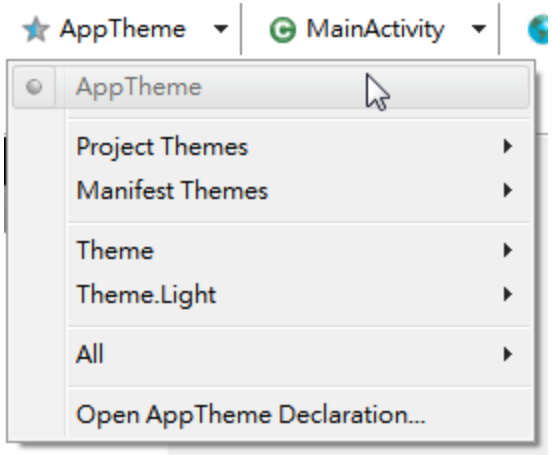
如何預覽Theme

- 步驟四：AppTheme表示目前使用的佈景主題點選一下便可以切換Theme



如何預覽Theme

- 步驟五：選擇要觀看的Theme
  - 此處分為Project Themes和Manifest Themes是定義在自己專案中的佈景主題
  - Theme和Theme.Light是Android預設的佈景主題
  - All表示全部可供使用的佈景主題列表
  - Open AppTheme Declaration 可以開啟專案的themes.xml或style.xml



如何預覽Theme

- 針對舊專案
- 步驟一：一樣更新ADT至最新版
- 步驟二：打開Layout檔案
- 步驟三：上方工具中顯示的佈景主題是預設的Theme
  - 不會是AppTheme，因為舊專案不會自動建立佈景主題

客製化Theme

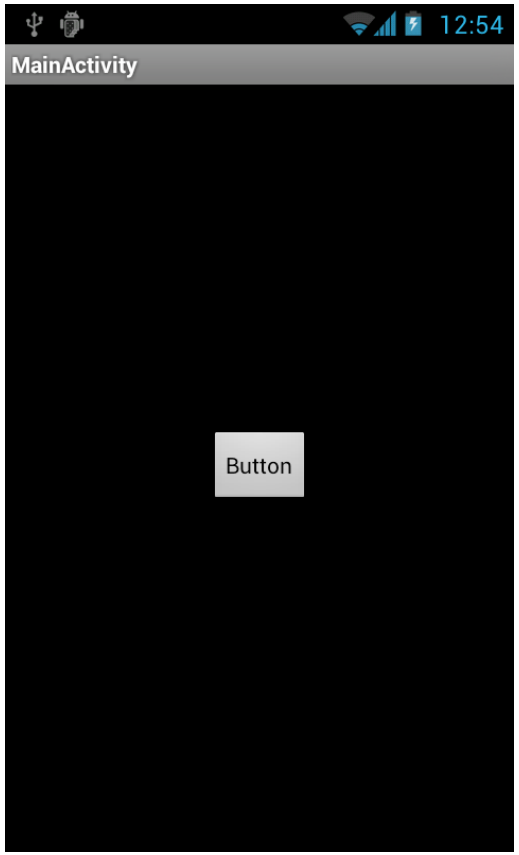
- 搭配專案：CustomTheme

客製化Theme

- 我們製作Android應用程式時幾乎都是使用Android提供的Widget，如Button, TextView
- 所以要「真正」客製化這些物件的外觀，最好的方式就是「複寫」Android針對不同Widget訂出的Style
- 以下將以更改android-10 (Gingerbread)Button的呈現外觀為例子

客製化Theme

- 建立一個Activity，Layout如下



客製化Theme

- 客製化Theme共有六個步驟
  1. 找到Android針對Widget定義的屬性
  2. 打開Android的themes.xml找尋屬性
  3. 打開Android的styles.xml找到屬性對應的style
  4. 專案的styles.xml中複寫Android中的style
  5. 在自訂的Theme中將屬性指定至複寫的Style
  6. 在AndroidManifest.xml中設定Application或Activity使用的Theme

客製化Theme

- 步驟一、找到Android針對Widget定義的屬性
  - 首先到**<android_sdk>\platforms\android-10\data\res\values\attrs.xml**
    - 這檔案定義了所有Android對每個Widget設定的屬性
    - 請參閱每個XML標籤上方的註解說明
      - 是英文的，沒錯！
      - 很難閱讀、更難找到架構，沒錯！

客製化Theme

- 步驟一、找到Android針對Widget定義的屬性
  - 下方可以看到註解「Normal Button styles」表示是一般按鈕的屬性定義
  - name是buttonStyle，請記得這個索引值
  - format表示可以賦予本屬性的值為何？是一個reference
    - reference表示@layout, @string, @drawble, @color…都是reference

```
<!-- ============= -->
<!-- Button styles -->
<!-- ============= -->
<eat-comment />

<!-- Normal Button style. -->
<attr name="buttonStyle" format="reference" />
```

客製化Theme

- 步驟二、打開android的themes.xml並找尋剛剛的屬性索引值
  - 打開**<android_sdk>\platforms\android-10\data\res\values\themes.xml**

客製化Theme

- 步驟二、打開android的themes.xml並找尋剛剛的屬性索引值
  - 開啟themes.xml可以發現第一個物件就是Theme

```
<style name="Theme">
```

  - 這是所有Android的Theme的基底，其餘的佈景主題都是針對這個基底去修改的

客製化Theme

- 步驟二、打開android的themes.xml並找尋剛剛的屬性索引值
  - 「buttonStyle」是我們找尋的目標

```
<!-- Button styles -->
<item name="buttonStyle">@android:style/Widget.Button</item>
```

  - android:style/Widget.Button表示關於按鈕的設定寫在styles.xml中的Widget.Button項目

客製化Theme

- 步驟三、找到styles.xml中找到屬性所對應的style
  - 打開檔案**<android_sdk>\platforms\android-10\data\res\values\styles.xml**
  - 我們找尋的目標是「Widget.Button」

客製化Theme

- 步驟三、找到styles.xml中找到屬性所對應的style
  - 以下就是找到的項目
    - 找到了嗎？找不到就擅用搜尋吧

```
<style name="Widget.Button">
    <item name="android:background">@android:drawable/btn_default</item>
    <item name="android:focusable">true</item>
    <item name="android:clickable">true</item>
    <item name="android:textAppearance">?android:attr/textAppearanceSmallInverse</item>
    <item name="android:textColor">@android:color/primary_text_light</item>
    <item name="android:gravity">center_vertical|center_horizontal</item>
</style>
```

客製化Theme

- 步驟三、找到styles.xml中找到屬性所對應的style
  - 這表示Android要顯示Button前會使用以下的屬性來定義Button的外觀
    - background 按鈕背景圖
    - textAppearance 按鈕上呈現文字的長相
    - textColor 按鈕文字的顏色
    - gravity 按鈕文字在按鈕中的對齊方式

```
<style name="Widget.Button">
    <item name="android:background">@android:drawable/btn_default</item>
    <item name="android:focusable">true</item>
    <item name="android:clickable">true</item>
    <item name="android:textAppearance">?android:attr/textAppearanceSmallInverse</item>
    <item name="android:textColor">@android:color/primary_text_light</item>
    <item name="android:gravity">center_vertical|center_horizontal</item>
</style>
```

客製化Theme

- 步驟四、在專案的styles.xml中複寫Android中的style
  - 複寫時多半不會自製一份，而是針對需要的部分進行複寫
  - 首先自己取名一個Style稱為MyButtonStyle

```
<style name="MyButtonStyle" parent="android:style/Widget.Button">
    <item name="android:background">@drawable/btn_custom</item>
    <item name="android:textColor">#FFFF00</item>
</style>
```

客製化Theme

- 步驟四、在專案的styles.xml中複寫Android中的style
  - parent=“android:style/Widget.Button”表示這個Style是繼承android內建的Style「Widget.Button」

```
<style name="MyButtonStyle" parent="android:style/Widget.Button">
    <item name="android:background">@drawable/btn_custom</item>
    <item name="android:textColor">#FFFF00</item>
</style>
```

客製化Theme

- 步驟四、在專案的styles.xml中複寫Android中的style
  - 以下表示複寫Android內建Style中的background以及textColor
  - 其餘沒寫的，如textAppearance或gravity則維持原本Widget.Button的設定

```
<style name="MyButtonStyle" parent="android:style/Widget.Button">
    <item name="android:background">@drawable/btn_custom</item>
    <item name="android:textColor">#FFFF00</item>
</style>
```

客製化Theme

- 步驟五：在自訂的Theme中將屬性指定至複寫Style
  - 在專案的res資料夾下自己建立themes.xml
  - 當然也可以直接將theme的定義寫在styles.xml中
  - 我建議「分開」會比較好控管

客製化Theme

- 步驟五：在自訂的Theme中將屬性指定至複寫Style
  - 在自行建立的themes.xml中建立以下內容

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppTheme" parent="android:Theme" >
        <item name="android:buttonStyle">@style/MyButtonStyle</item>
    </style>
</resources>
```

客製化Theme

- 步驟五：在自訂的Theme中將屬性指定至複寫Style
  - 這個佈景主題名稱為AppTheme
  - 繼承於Android內建佈景主題Theme
    - parent=“android:Theme”

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppTheme" parent="android:Theme" >
        <item name="android:buttonStyle">@style/MyButtonStyle</item>
    </style>
</resources>
```

客製化Theme

- 步驟五：在自訂的Theme中將屬性指定至複寫Style
  - 在此將buttonStyle指定給自己客製化的MyButtonStyle
    - 記得要寫成android:buttonStyle！

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="AppTheme" parent="android:Theme" >
        <item name="android:buttonStyle">@style/MyButtonStyle</item>
    </style>
</resources>
```

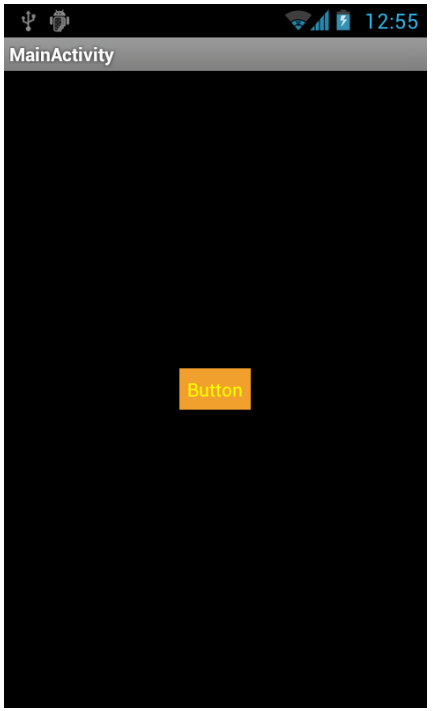
客製化Theme

- 步驟六：在AndroidManifest.xml中設定Application或Activity使用的Theme
  - 打開AndroidManifest.xml
  - 對著application增加屬性android:theme=“@style/AppTheme”表示本應用程式使用這個佈景主題
  - android:theme也可以針對<activity>使用

```
<application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
```

客製化Theme結論

- 試著編譯並安裝應用程式，可以發現按鈕變樣了，而且不論建立幾個都會自動變成新指定的按鈕樣式，這就是佈景主題
- 佈景主題的設定需要多練習，因為官方並沒有良好的文件以及相關教學



動態設定佈景主題

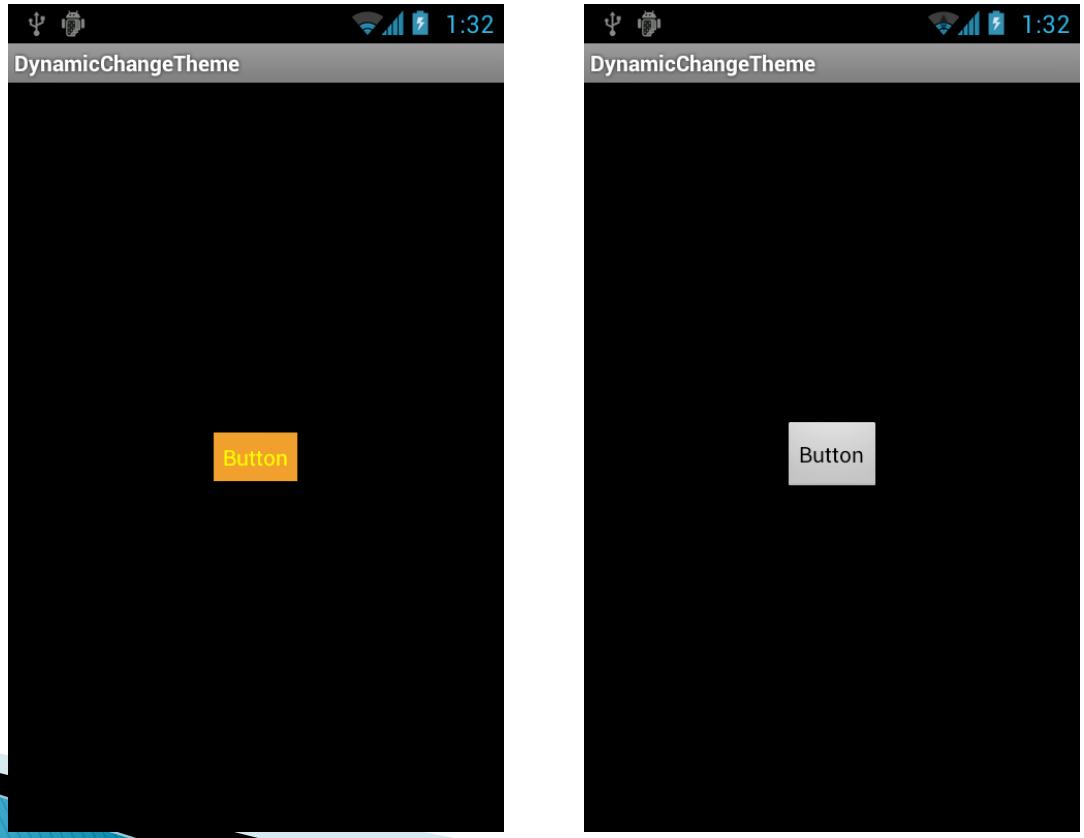
- 搭配專案：
  DynamicChangeTheme

如何動態改變佈景主題

- 藉由Activity的方法**setTheme()**設定佈景主題
- 佈景主題的設定**必須得在Activity呼叫setContentView()之前完成**
- 依照Android正規機制，設定心的佈景主題給應用程式，應用程式必須得**重新啟動**
- 來看看專案DynamicChangeTheme

動態改變佈景主題

- 專案出現時會如下圖：按畫面中的按鈕就會更換佈景主題



動態改變佈景主題

- 打開MainActivity中的onCreate()
  - 這是試著取得SharedPreferences中佈景主題的值
    若是0就用專案的AppTheme，否則就用Android內建的Theme

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

    final SharedPreferences pref = PreferenceManager.getDefaultSharedPreferences(this);
    final int theme = pref.getInt("theme", 0);

    setTheme((theme == 0) ? R.style.AppTheme : android.R.style.Theme);
    setContentView(R.layout.activity_main);
```

動態改變佈景主題

- 特別注意，setTheme()寫在setContentView()之前
  - setTheme()寫在setContentView()之後會出錯

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

    final SharedPreferences pref = PreferenceManager.getDefaultSharedPreferences(this);
    final int theme = pref.getInt("theme", 0);

    setTheme((theme == 0) ? R.style.AppTheme : android.R.style.Theme);
    setContentView(R.layout.activity_main);
```

動態改變佈景主題

- 在畫面中按鈕的OnClickListener
  - 程式碼中紅色框的部分就是結束掉目前Activity (呼叫finish())，並且重新啟動 (startActivity())的方法

```
Button button = (Button) findViewById(R.id.button1);
if (button != null) {
    button.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
            Editor edit = pref.edit();
            edit.putInt("theme", (theme == 0) ? 1 : 0);
            edit.commit();
            finish();
            startActivity(new Intent(MainActivity.this, MainActivity.class));
        }
    });
}
```

END