

# Android App

## 程式設計教本 之無痛起步

引領入門

最簡單、最易懂的初學教材



## 第 2 章

# Android 程式設計基礎講座

本投影片（下稱教用資源）僅授權給採用教用資源相關之旗標書籍為教科書之授課老師（下稱老師）專用，老師為教學使用之目的，得摘錄、編輯、重製教用資源（但使用量不得超過各該教用資源內容之80%）以製作為輔助教學之教學投影片，並於授課時搭配旗標書籍公開播放，但不得為網際網路公開傳輸之遠距教學、網路教學等之使用；除此之外，老師不得再授權予任何第三人使用，並不得將依此授權所製作之教學投影片之相關著作物移作他用。

# 前言

- 2-1 Android App 的主角：Activity
- 2-2 Android 程式的設計流程
- 2-3 認識 Activity 的基本程式邏輯
- 2-4 元件的佈局與屬性設定
- 2-5 開始動手寫程式
- 2-6 輸入欄位 EditText 元件
- 2-7 使用 USB 線將程式部署到手機上執行

## 2-1 Android App 的主角：Activity

1. Activity (活動)
2. Service (背景服務)
3. Content Provider (內容提供者)
4. Broadcast Receiver (廣播接收端)

# Android App 的主角：Activity

- Activity (活動)
- 實際的狀況是這樣

# Activity (活動)

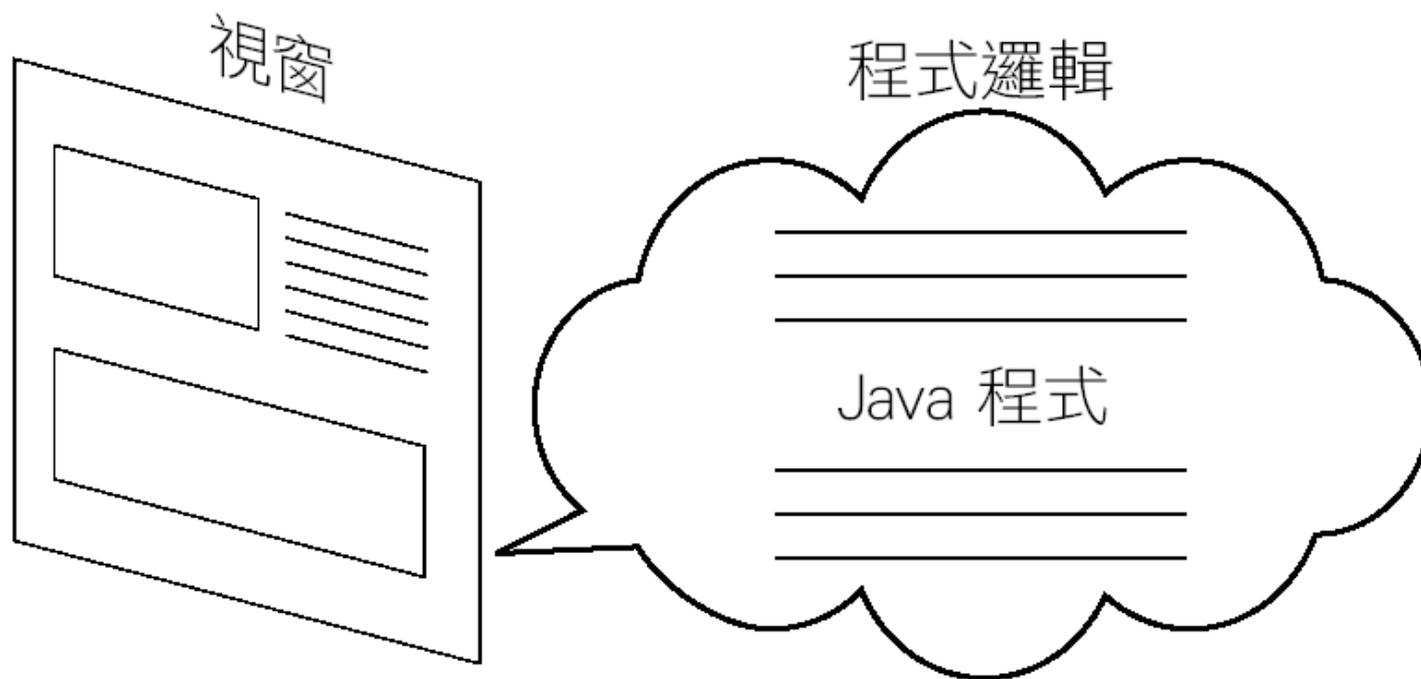
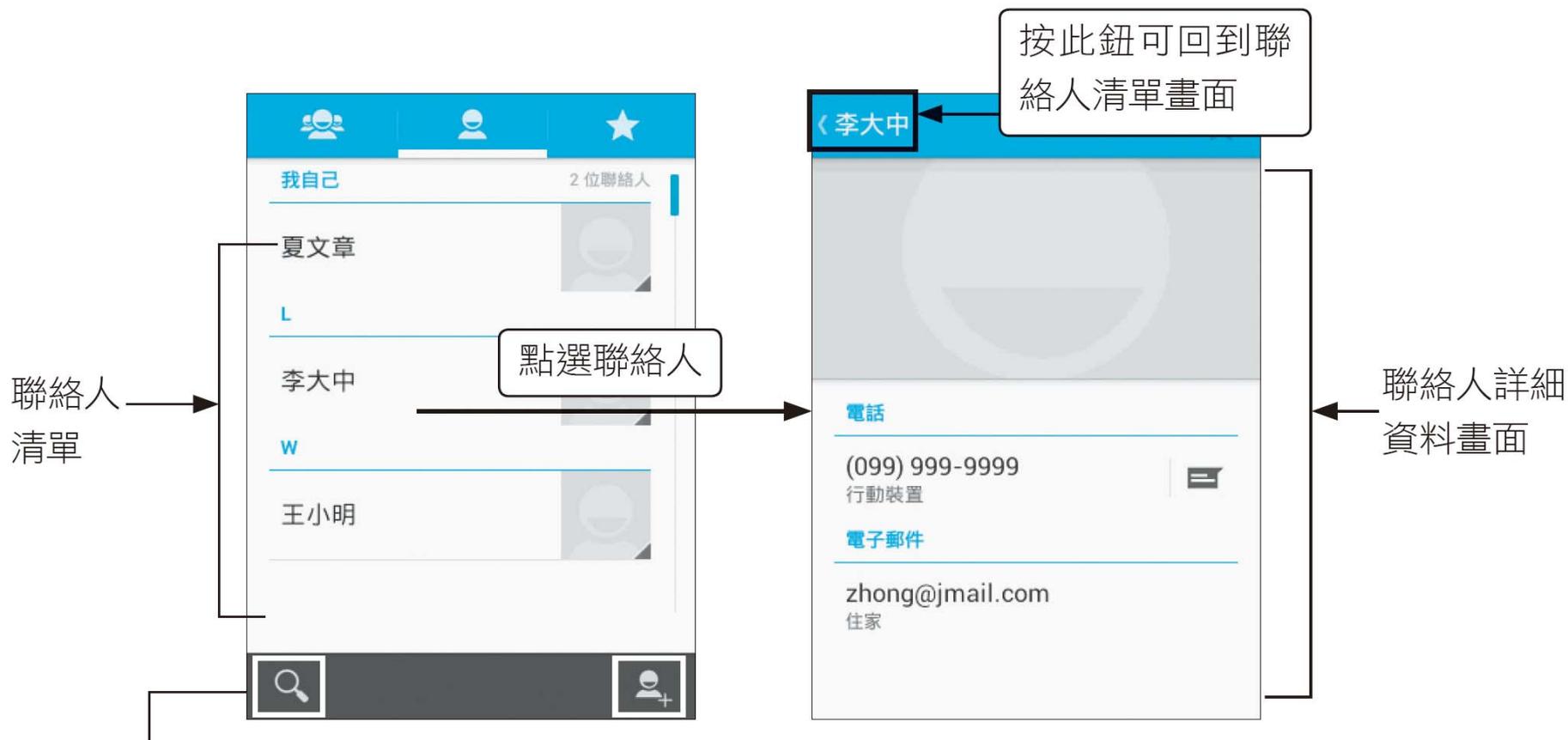
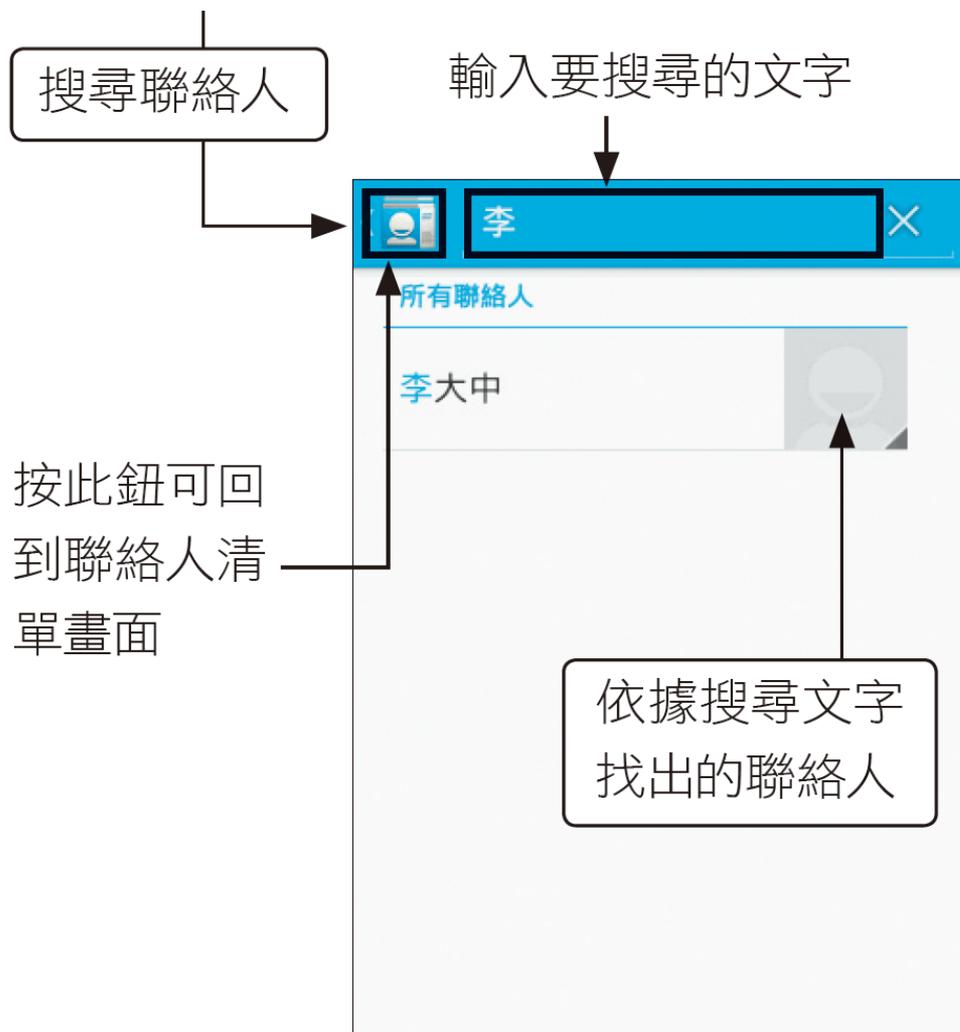


圖 2-1 一個典型的 Activity

# 實際的狀況是這樣



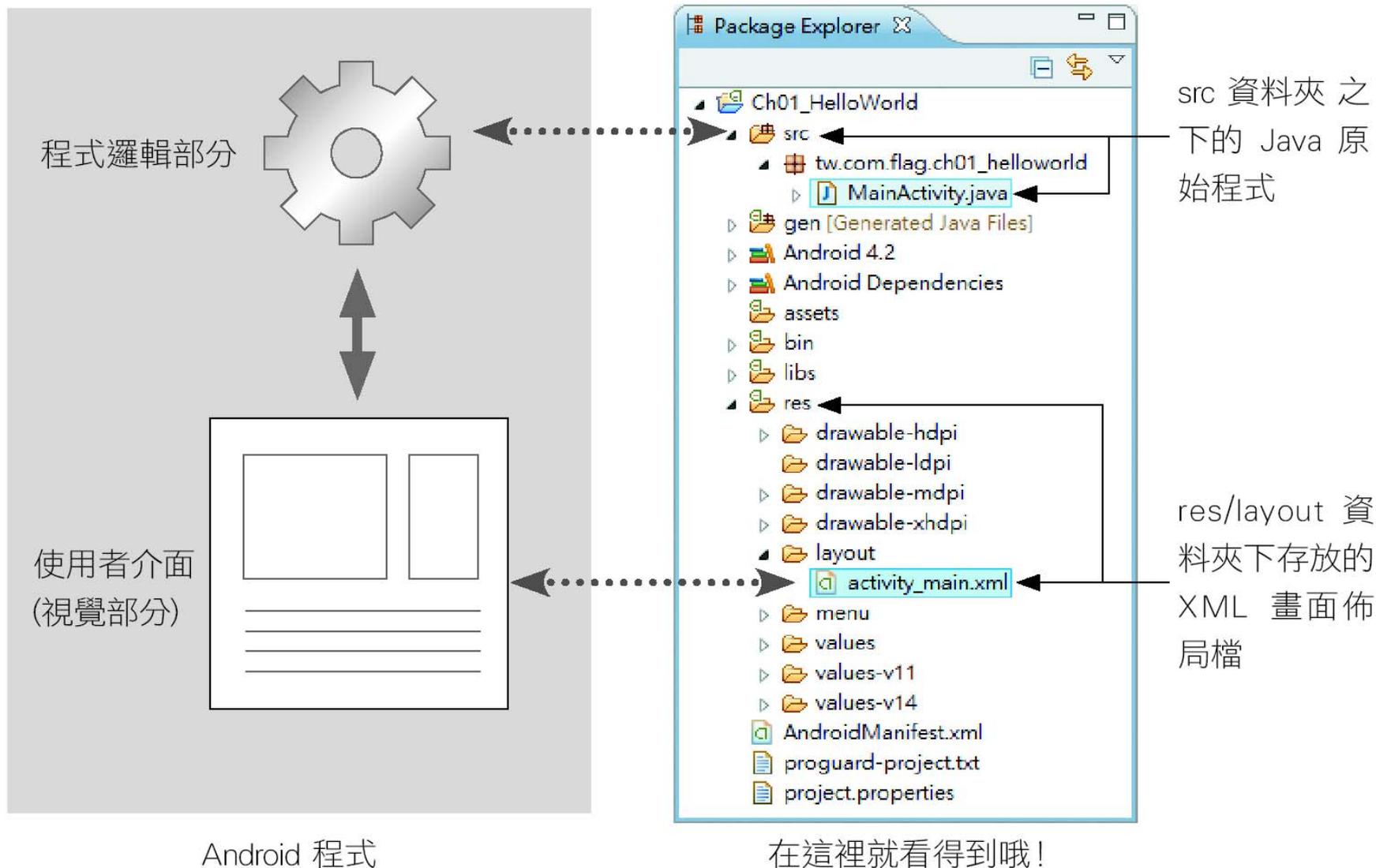
# 實際的狀況是這樣



## 2-2 Android 程式的設計流程

- 視覺設計 + 程式邏輯
- 用圖形化介面來做視覺設計
- 用 Java 來寫程式邏輯
- 最後把視覺設計與程式碼組建 (Build) 起來

# 視覺設計 + 程式邏輯



# 視覺設計 + 程式邏輯

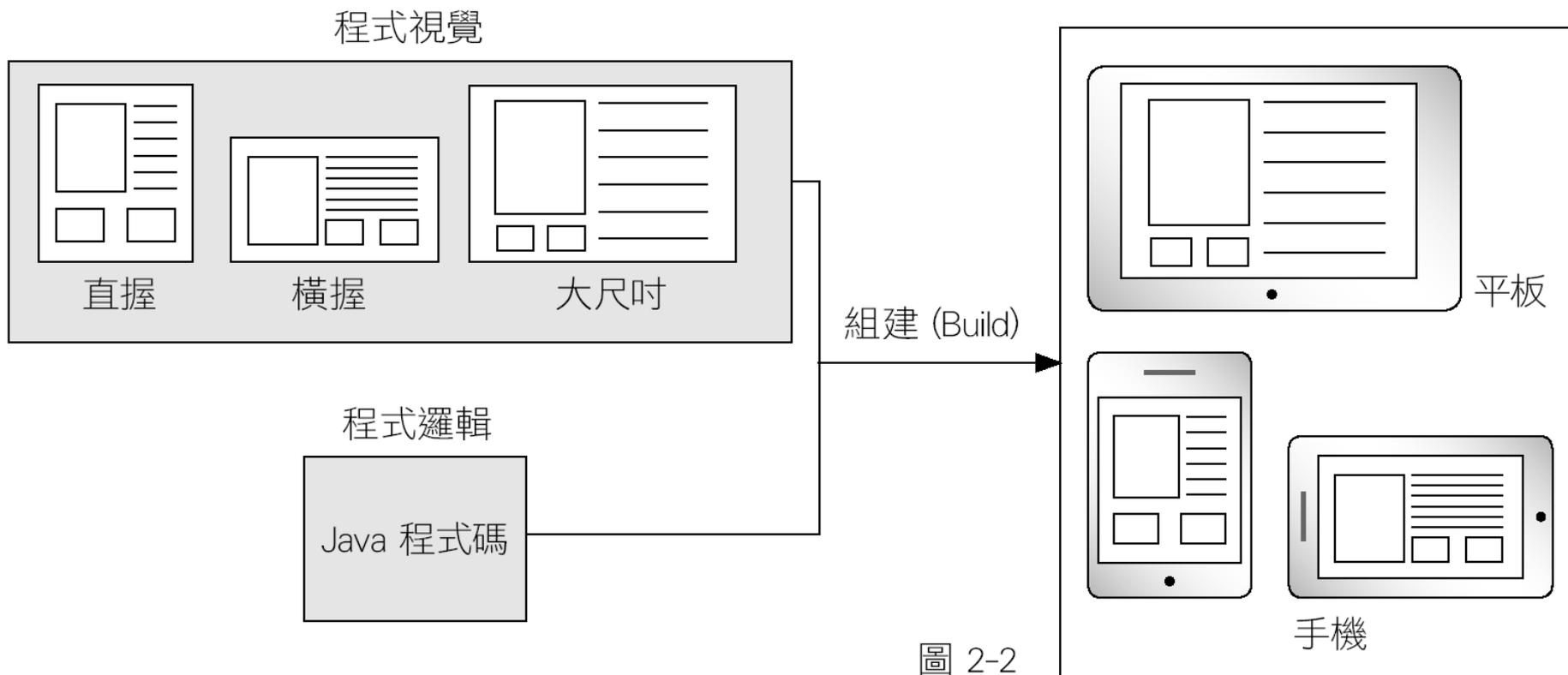
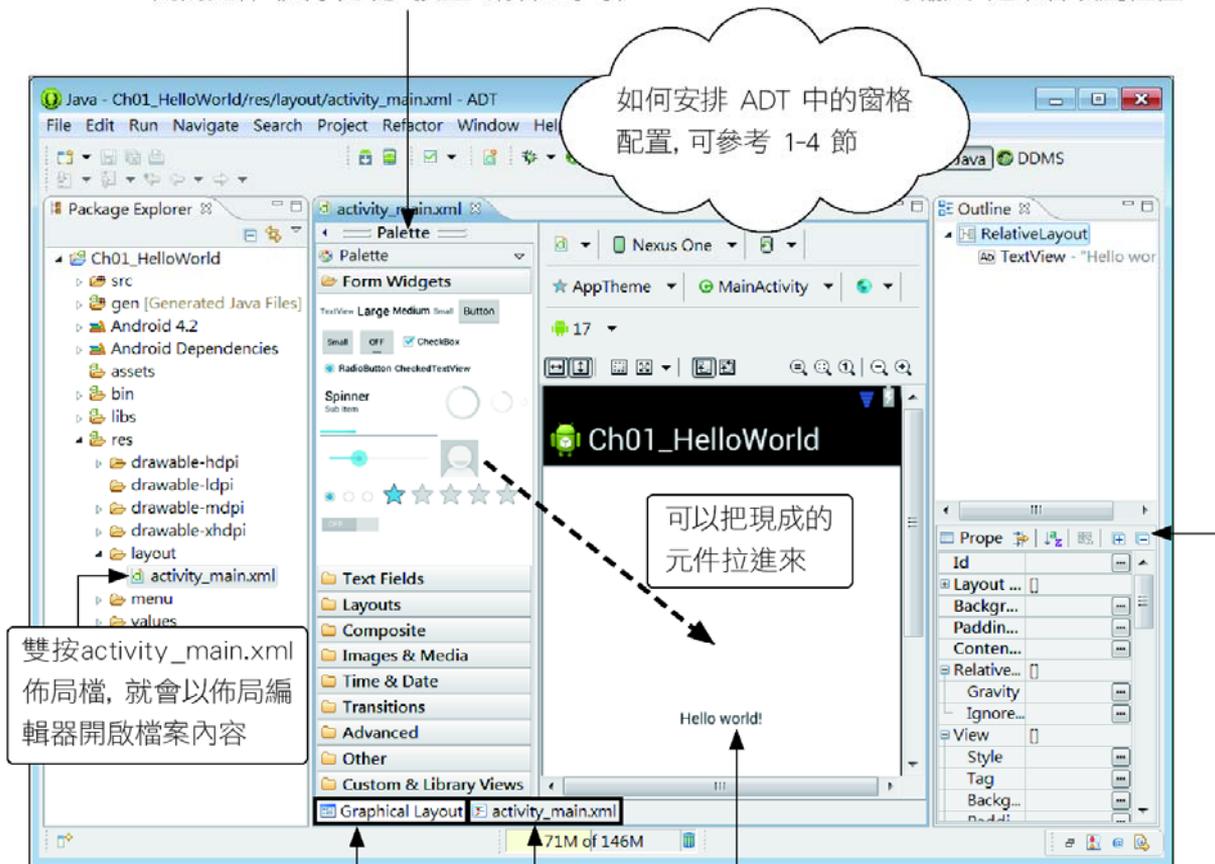


圖 2-2

# 用圖形化介面來做視覺設計

設計介面左側 **Palette** 窗格會列出可使用的元件 (文字方塊、按鈕、滑條...等等)

在右下角屬性 (Properties) 窗格可輸入、選取各項屬性值

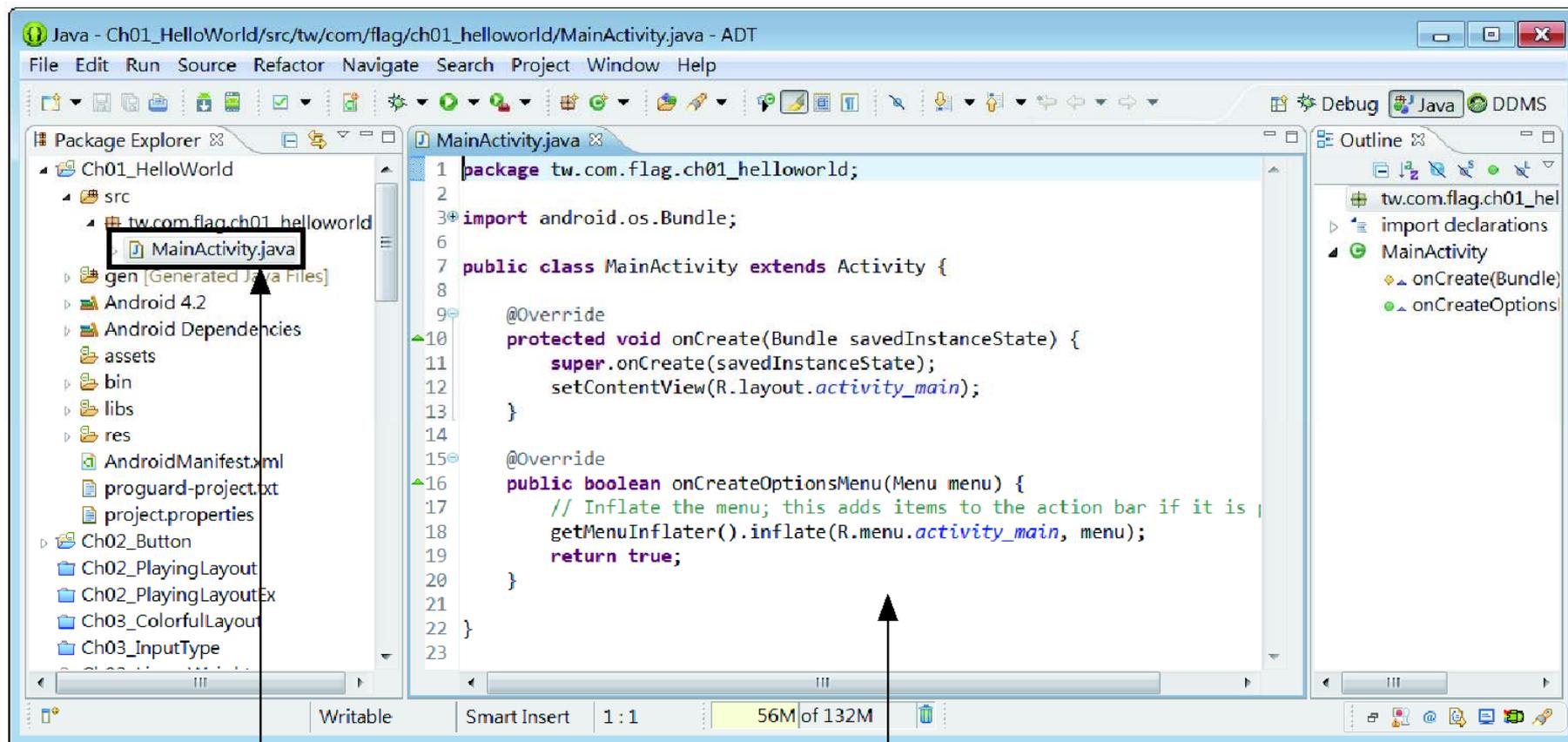


預設以圖形化設計介面顯示

按這裡切換到 XML 文字模式

在中間的佈局編輯區畫面可用滑鼠拉曳、調整元件位置

# 用Java 來寫程式邏輯



自動建立的 Java 程式檔

Java 程式內容

# 最後把視覺設計與程式碼組建 (Build) 起來

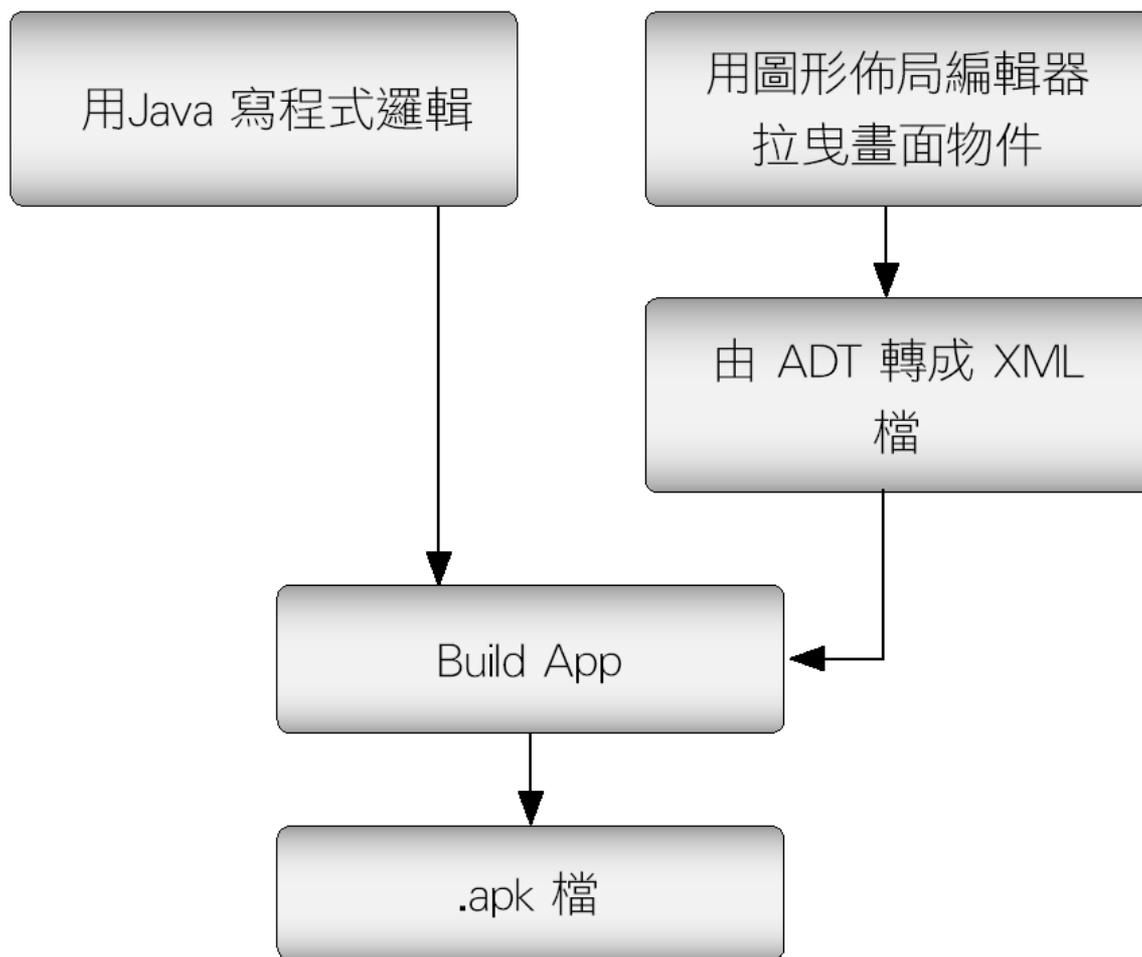


圖 2-3 Android 程式設計的基本流程

## 2-3 認識 **Activity** 的基本程式邏輯

- 初識 MainActivity 框架
- onCreate(): MainActivity 第一件要做的事
- setContentView(): 載入佈局檔
- 資源的 ID

# 初識 MainActivity 框架

Java 程式檔名稱和主程式類別名稱都叫 MainActivity

在新增專案精靈中可在此自訂 Activity Name (此處的 "MainActivity" 為預設值)

```
1 package tw.com.flag.ch01_helloworld;
2
3 import android.os.Bundle;
6
7 public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
```

Activity Name MainActivity  
Layout Name activity\_main  
Navigation Type None

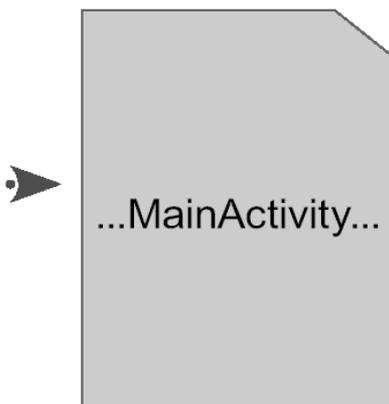
一個叫 MainActivity  
一個叫 activity\_main  
真搞昏我老人家...

Layout Name 就是視覺佈局檔名稱

# 初識 MainActivity 框架

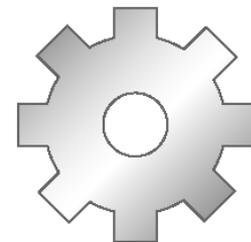


使用者啟動手機的 Android App，系統會檢視 AndroidManifest.xml 的內容



AndroidManifest.xml：  
記錄了程式一開始要啟動的 Activity

系統會載入 MainActivity 類別並建立物件、開始執行



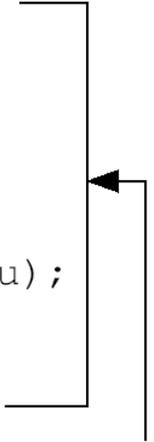
# onCreate(): MainActivity 第一件要做的事

## 程式 2-1 新建專案時 ADT 自動產生的 Java 程式框架

```
01 package tw.com.flag.ch01_helloworld; ← 我們之前設定的套件 (package)
02                                     名稱, 這是 Java 的標準語法
03 import android.os.Bundle;
04 import android.app.Activity;
05 import android.view.Menu;
06
07 public class MainActivity extends Activity {
08
09     @Override
10     protected void onCreate(Bundle savedInstanceState) {
11         super.onCreate(savedInstanceState); ← 先呼叫父類別的
                                           onCreate() 來做該做的事
```

# onCreate(): MainActivity 第一件要做的事

```
12     setContentView(R.layout.activity_main); ← 然後我才來
13 }                                           做我要做的
14
15 @Override
16 public boolean onCreateOptionsMenu(Menu menu) {
17     // Inflate the menu; this adds items to the
18     // action bar if it is present.
19     getMenuInflater().inflate(R.menu.activity_main, menu);
20 }
21
22 }
```



這一部分暫時用不到...

# onCreate(): MainActivity 第一件要做的事

```
onCreate() {  
... 加入自己的程式 ...  
}
```

MainActivity.java

Android 系統載入/啟動 Activity 後，  
會呼叫 onCreate() 方法

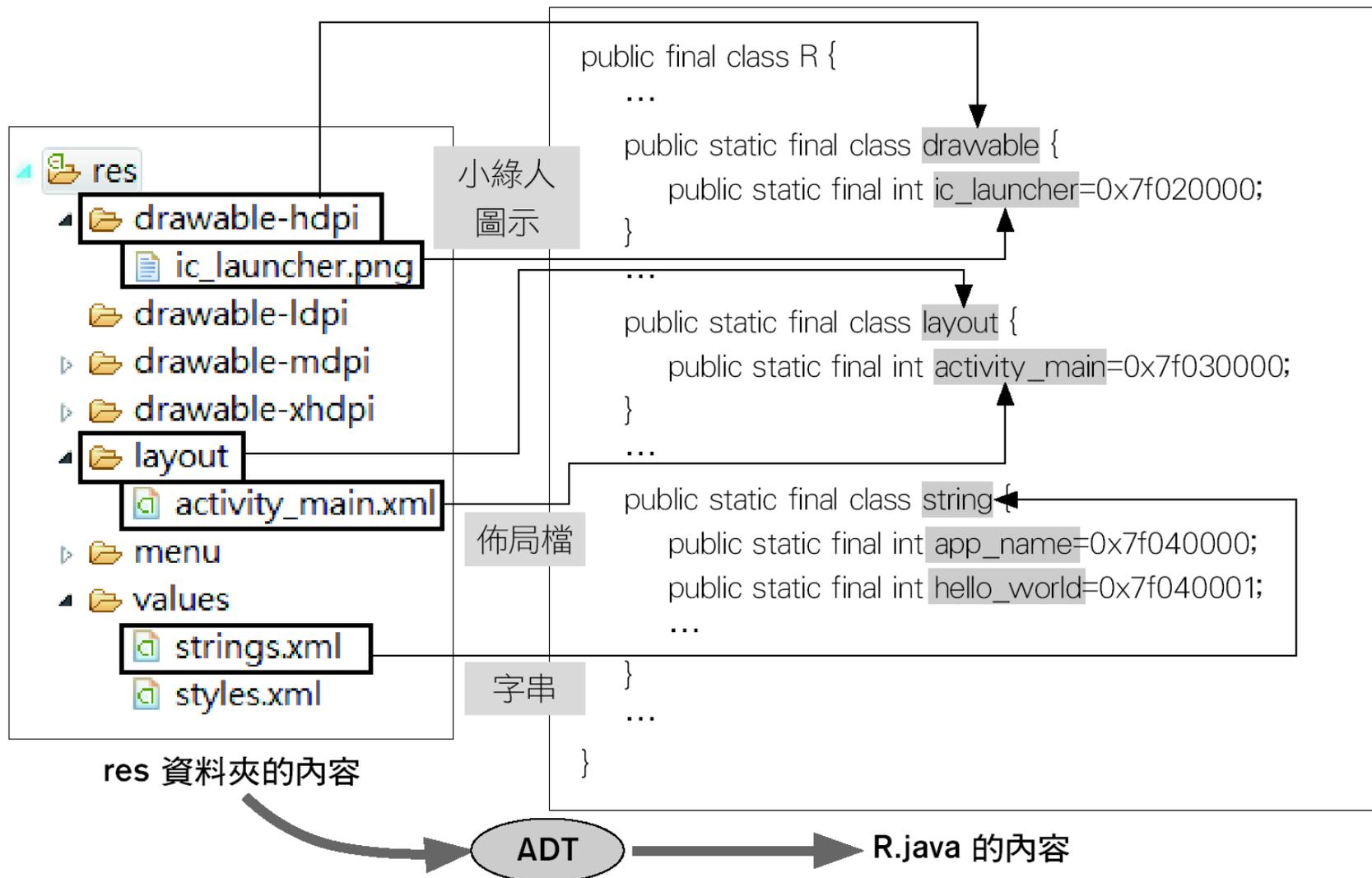


# setContentView() : 載入佈局檔

```
setContentView(R.layout.activity_main);
```

# 資源的 ID

R.java 在專案的 gen/package 名稱/目錄下



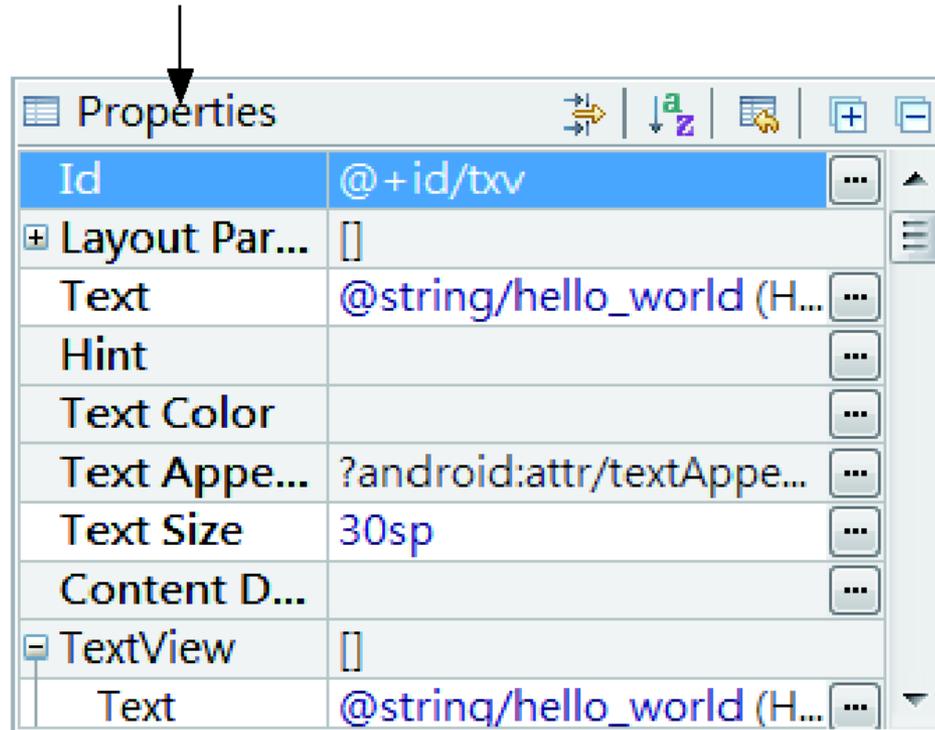
# 資源的 ID

`R.layout.activity_main` ← 可取得 layout 中的 `activity_main.xml` 佈局檔  
`R.string.app_name` ← 可取得 `strings.xml` 中的 `app_name` 字串內容

請自行和上頁  
R.java 內容比對

## 2-4 元件的佈局與屬性設定

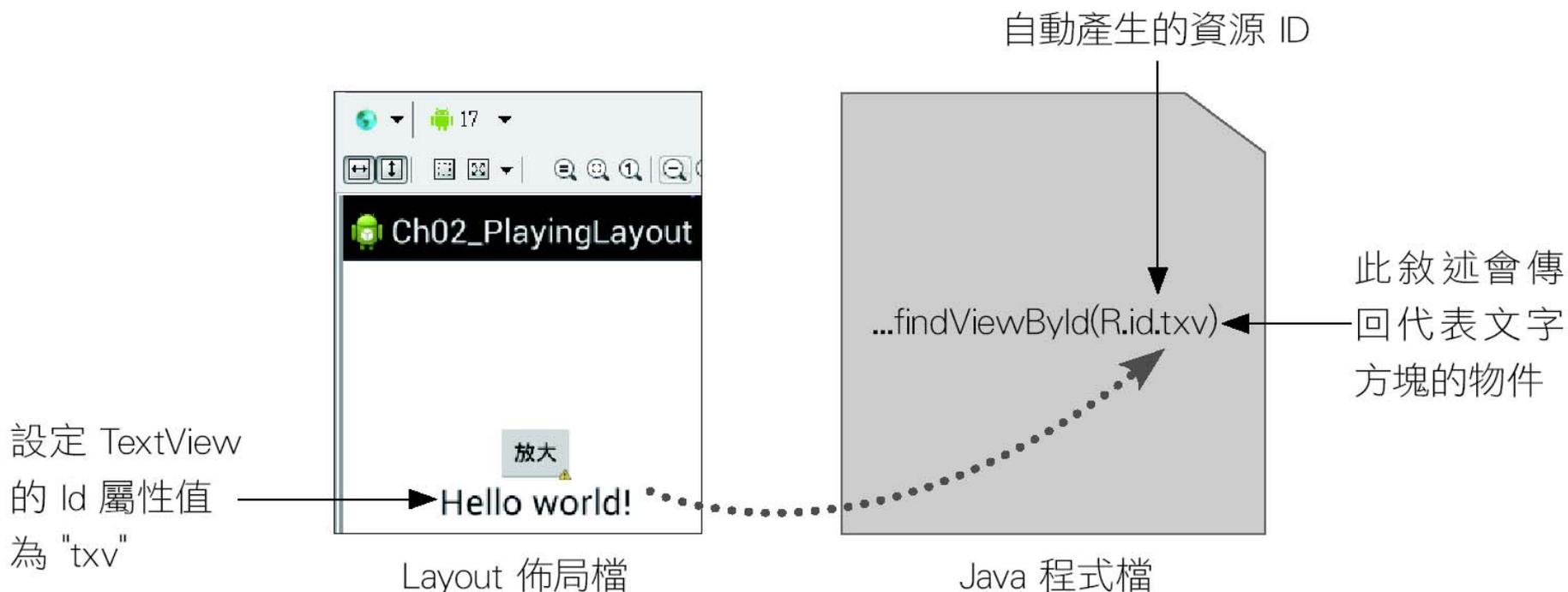
在這裡



# 元件的佈局與屬性設定

- Id 屬性
- findViewById() 方法
- 常見的屬性類型

# findViewById() 方法



# findViewById() 方法

```
// 取得 TextView 物件  
TextView myTxv = (TextView) findViewById(R.id.txv);  
// 呼叫 TextView 類別的 setText() 方法  
myTxv.setText("請問芳名");
```

它傳回的是 View 基礎類別的物件

轉換成 TextView 類別

設定 (set) 元件顯示的文字 (Text) 為 "請問芳名"

# 常見的屬性類型

屬性	屬性值舉例	說明
Id	@+id/button1	通知 ADT 建立新的資源 ID
Above	@id/txv	將元件放置在畫面中名稱為 txv 的元件上方
On Click	bigger	表示當元件被『按一下』後, 要執行 Activity 中定義的 bigger() 方法
Text	@string/hello_world	將 res/values 資料夾下的 strings.xml 檔中名稱為 hello_world 的字串內容顯示在元件上
Text Size	30sp	將元件上顯示的文字設定為 30sp 大小
Src	@drawable/ic_launcher	將 res/drawable-XXX 資料夾下主檔名為 ic_launcher 的圖形顯示在元件上

\* "@+id/名稱" 代表如果名稱不存在, 就要建立此 id 名稱。而 "@id/名稱" 則是要使用此名稱。

# 常見的屬性類型

- 在屬性中直接設值
- 在屬性中設參照 (Reference)
- 在屬性中設方法的名稱

```
public void bigger(View v) {  
    ...// 放大 TextView 文字的程式碼  
}
```

# 常見的屬性類型

設計時

放大

- 1 指定按鈕的 On Click 屬性值為 "bigger"

執行時

放大



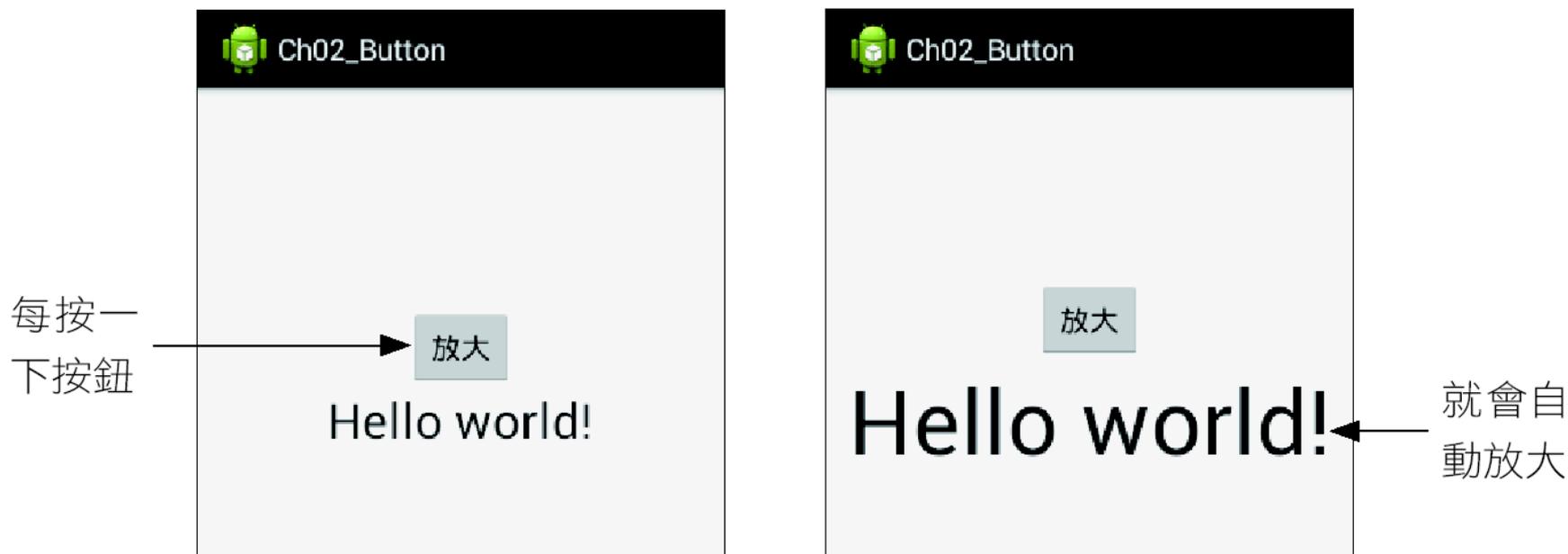
- 1 使用者按一下按鈕

- 2 在 Activity 類別中新增 bigger() 方法

```
bigger(){  
...  
}
```

- 2 Android 會呼叫 Activity 類別中的 bigger() 方法

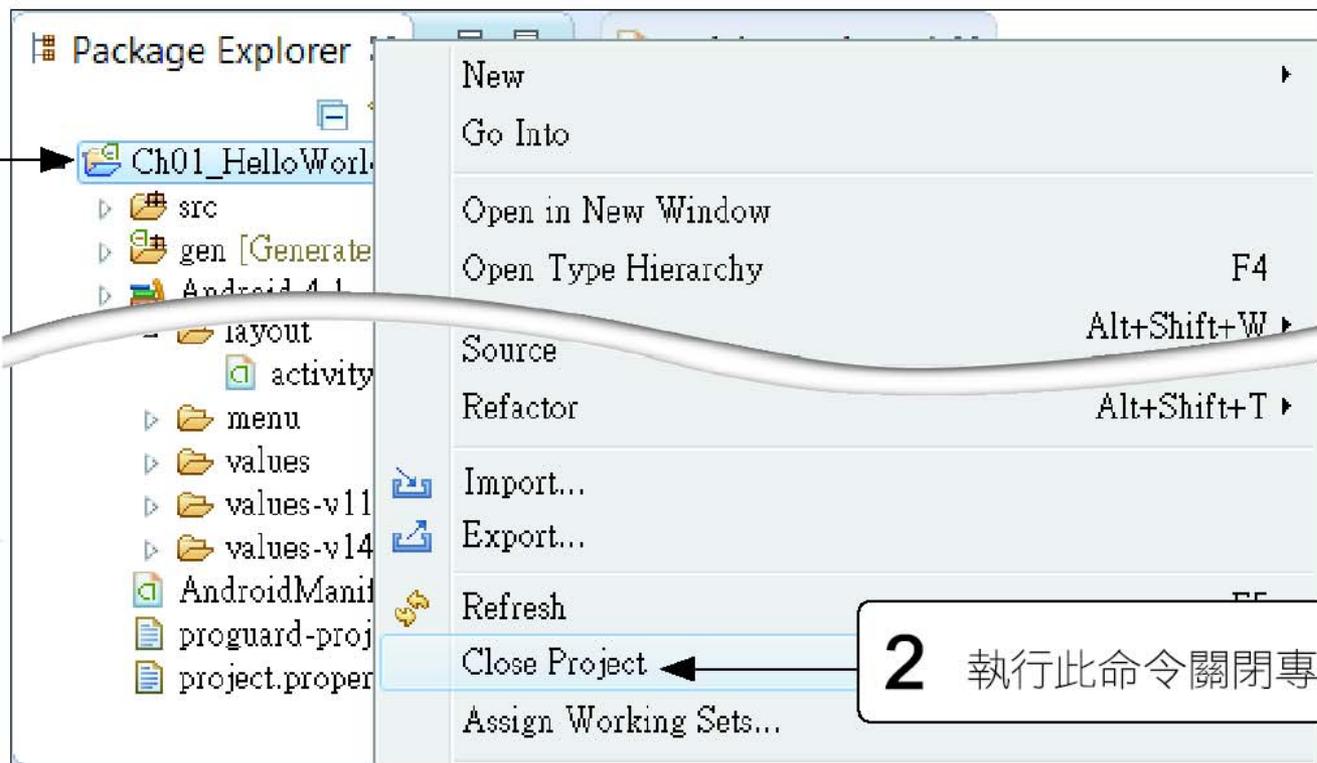
## 2-5 開始動手寫程式



# 範例2-1：按一下按鈕就放大顯示的文字

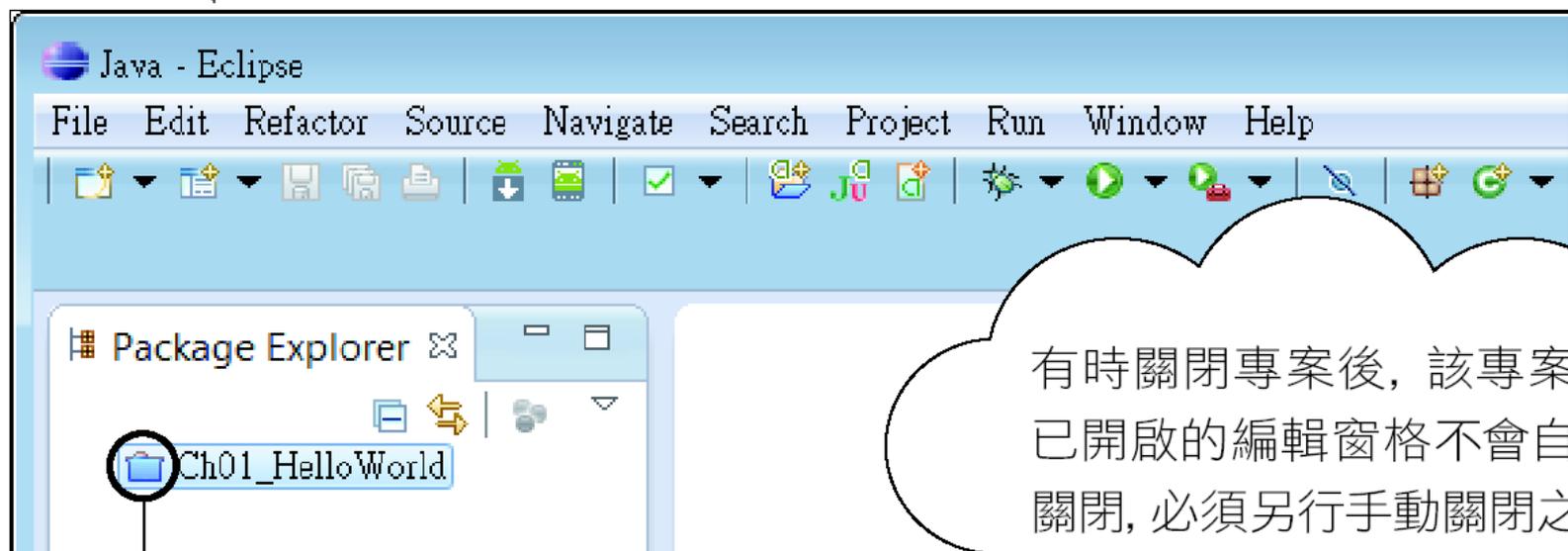
## step 1

- 1 在開啟中的專案上按下滑鼠右鈕



- 2 執行此命令關閉專案

# 按一下按鈕就放大顯示的文字



專案已經關閉了

有時關閉專案後，該專案中已開啟的編輯窗格不會自動關閉，必須另行手動關閉之

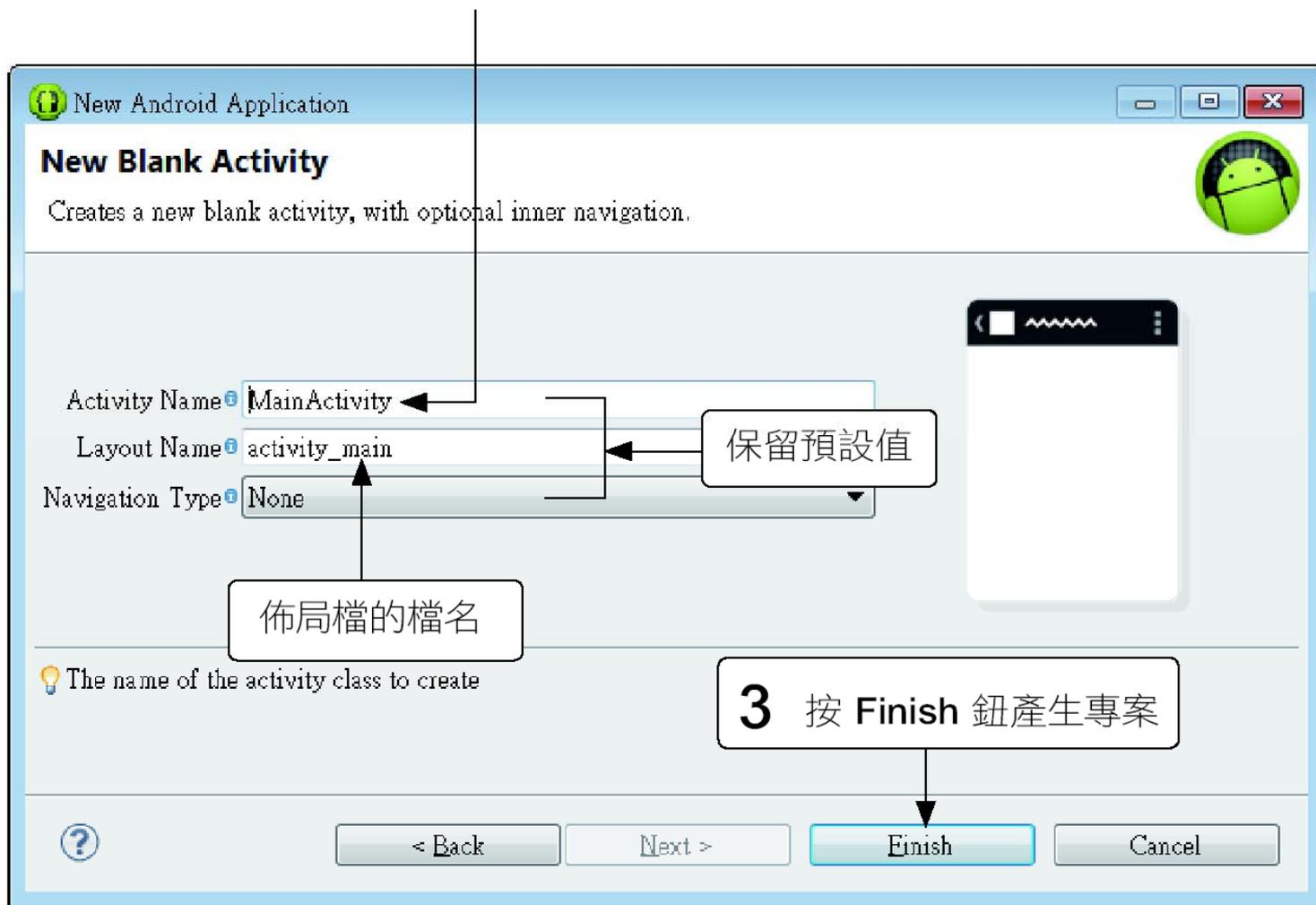
# 按一下按鈕就放大顯示的文字

## step 2

Application Name	Ch02_Button
Project Name	Ch02_Button
Package Name	tw.com.flag.ch02_button

# 按一下按鈕就放大顯示的文字

主 Activity 類別名及 Java 程式檔名



# 按一下按鈕就放大顯示的文字

預設包含這兩個元件

預設會進入圖形佈局設計畫面, 如果不小心切換到 XML 文字畫面, 請按左下角的 Graphical Layout 就可切回來

# 按一下按鈕就放大顯示的文字

- **RelativeLayout**：RelativeLayout 表示是透過『相對 (Relative) 位置』來規劃畫面上元件的位置 (包括『元件與元件』或『元件與 Layout』之間的相對位置)。下一章會介紹另一種佈局元件。
- **TextView 元件**：可稱為『文字標籤』、『文字方塊』，它的用途就是用來顯示一段文字，例如預設顯示 "Hello world!" 字串。若您畫面中的 "Hello world!" 是出現在其他位置，請先用滑鼠將其拖曳到畫面正中央，以利後續操作。

# 按一下按鈕就放大顯示的文字

## step 3

按這兩個鈕可放大或縮小預覽畫面

1 在 "TextView" 上按一下

2 屬性窗格出現 TextView 元件的屬性清單

畫面中的元件在四邊與角落出現方形控點, 表示是目前選取的元件

Outline 窗格

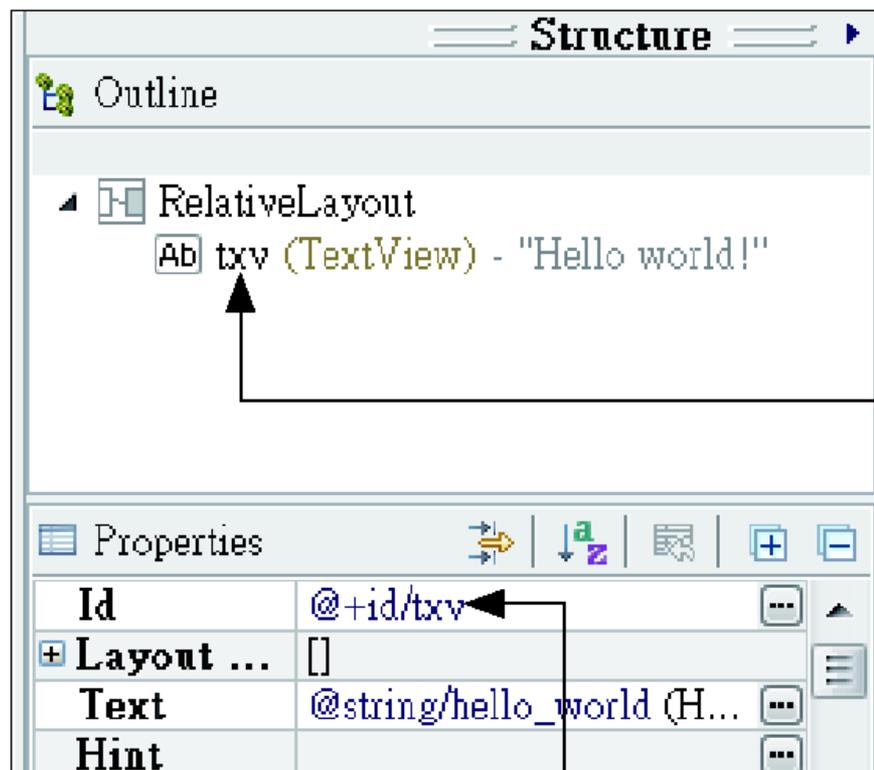
Properties

Layout Parameters	
To Left Of	...
To Right Of	...
Above	...
Below	...
Align Baseline	...
Align Left	...
Align Top	...
Align Right	...
Align Bottom	...
Align Parent Left	<input type="checkbox"/> ...
Align Parent Top	<input type="checkbox"/> ...
Align Parent Rig...	<input type="checkbox"/> ...
Align Parent Bott...	<input type="checkbox"/> ...
Center In Parent	<input type="checkbox"/> ...

虛線表示元件在 RelativeLayout 中的對齊位置 (本例是將元件「置中」)

# 按一下按鈕就放大顯示的文字

## step 4



2 在 Outline 窗格會顯示新的名稱:"txv"

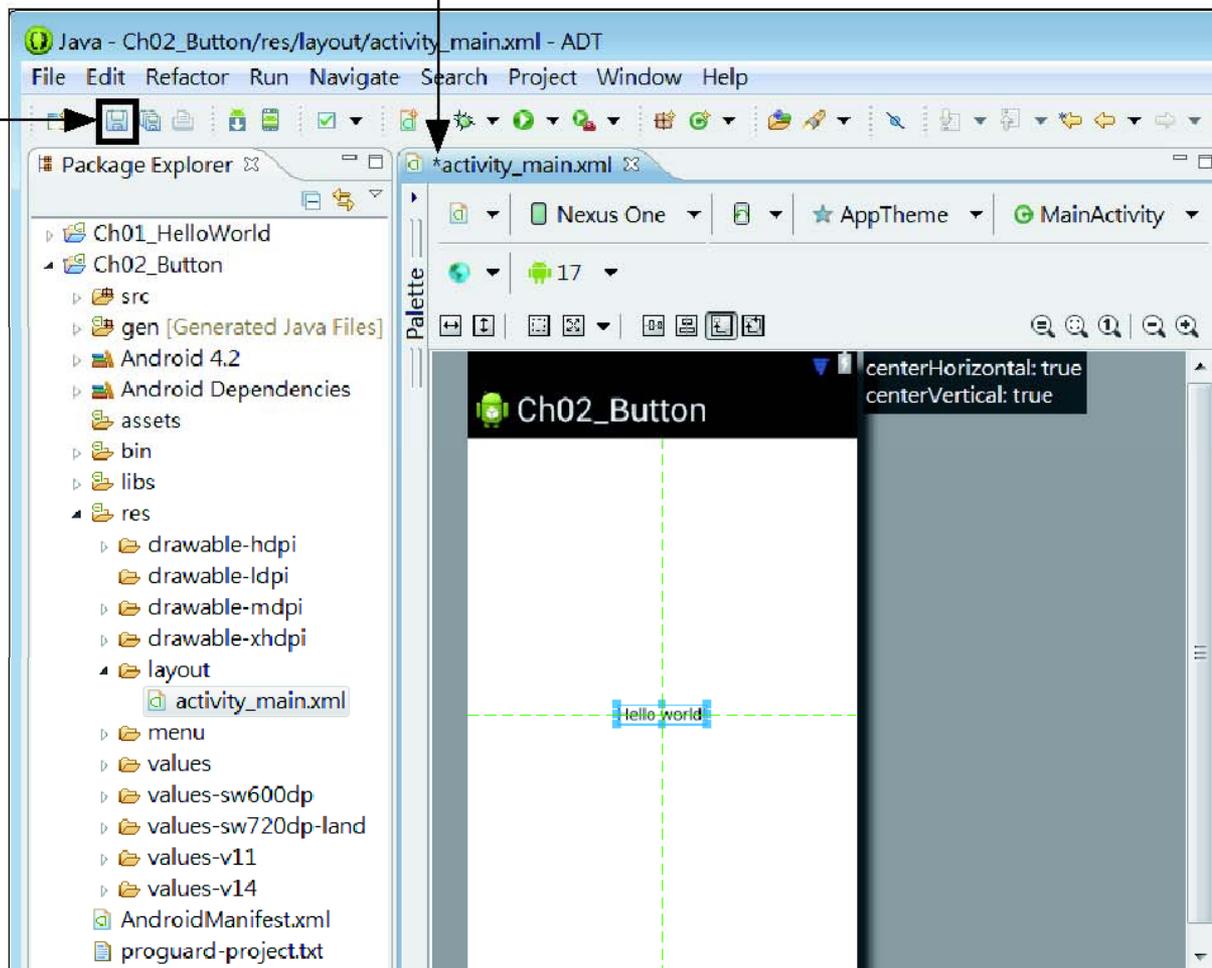
1 在 Id 屬性欄位填入 "@+id/txv"  
將 TextView 元件命名為 "txv"

# 按一下按鈕就放大顯示的文字

## step 5

請按此鈕 (或按  
**Ctrl + S** 鍵) 存檔

存檔後, 前頭的 '\*' 符號會消失

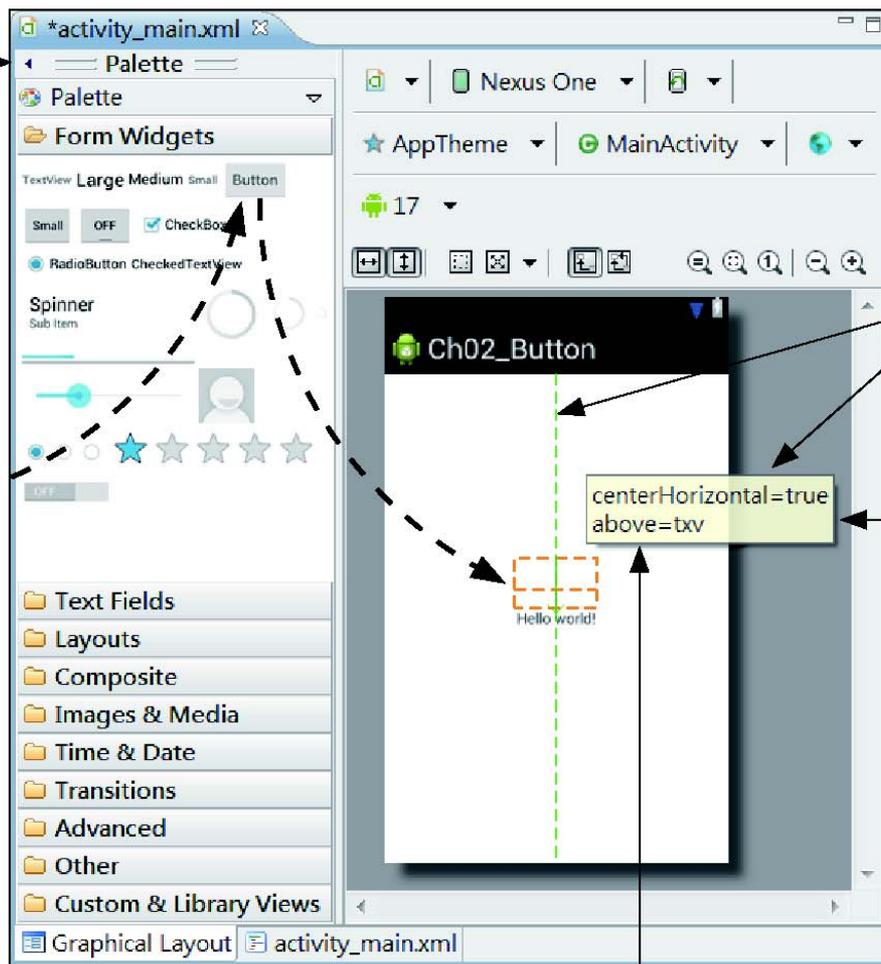


# 按一下按鈕就放大顯示的文字

## step 6

如果 **Palette** 是被收起的狀態，可按標題上的三角箭頭圖示將其展開

- 1 按住 **Button** 不放，將元件拉曳到 **TextView** (就是 Hello World 文字) 的『正』上方放開

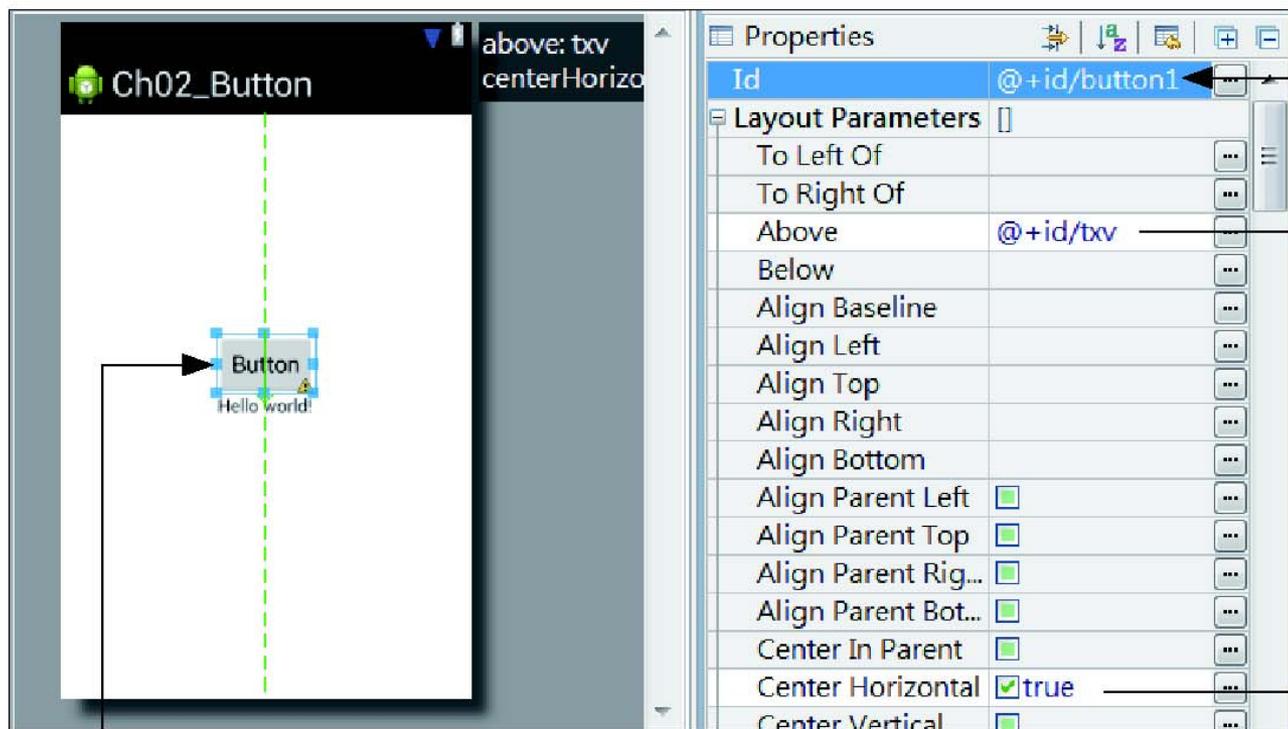


放開滑鼠鈕前，編輯器會用線條及浮動視窗提示元件的 Layout 相關屬性

表示元件『水平置中』

表示元件置於 txv 元件之上 (above)

# 按一下按鈕就放大顯示的文字



新加入元件的預設 Id (通常是『元件類型』加『流水號』1, 2, 3...)

放開滑鼠鈕, 屬性窗格會出現和先前浮動視窗內相同的設定 (若看不到, 可按 **Layout Parameter** 前的加號圖示)

2 在新加入的元件上按滑鼠右鈕, 執行『Edit Text』命令

# 按一下按鈕就放大顯示的文字

The image illustrates the process of changing a button's text in an Android application. On the left, the 'Resource Chooser' dialog is shown with 'Project Resources' selected. A text box contains '放大' (Magnify), and the 'Resolved Value' is also '放大'. A callout box with the number '3' points to this text box, containing the text '輸入要顯示在按鈕上的文字: "放大"'. Below the dialog, a callout with the number '4' points to the 'OK' button, with the text '按 OK 鈕'. On the right, an Android emulator window titled 'Ch02\_Button' shows a button with the text '放大' and a yellow warning triangle icon. A callout with the number '1' points to the warning icon, with the text '此數字代表警告的數量'. Another callout points to the new text '放大' on the button, with the text '出現新的文字'. A large grey arrow points from the Resource Chooser dialog to the emulator window.

3 輸入要顯示在按鈕上的文字: "放大"

4 按 OK 鈕

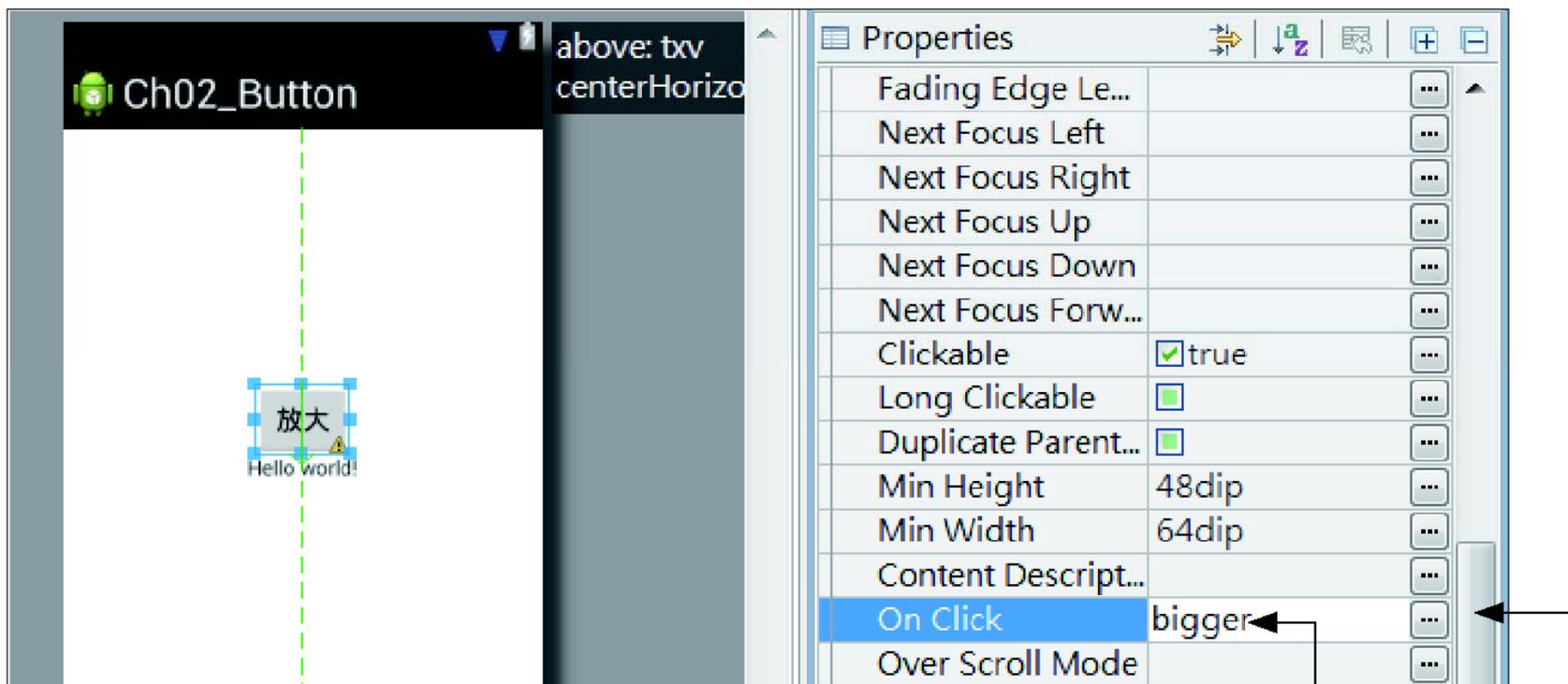
此數字代表警告的數量

出現新的文字

# 按一下按鈕就放大顯示的文字

## step 7

1 將屬性清單捲動到最後面, 找到 **On Click** 屬性



2 在此欄位輸入 "bigger"

# 按一下按鈕就放大顯示的文字

step 8

1 展開 src 及其下的套件名稱

2 雙按 MainActivity.java

```
1 package tw.com.flag.ch02_button;
2
3 import android.os.Bundle;
4
5
6
7 public class MainActivity extends Activity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13    }
14
15    @Override
16    public boolean onCreateOptionsMenu(Menu menu) {
17        // Inflate the menu; this adds items to the acti
18        getMenuInflater().inflate(R.menu.activity_main,
19        return true;
20    }
21
22 }
23
24
```

3 用滑鼠在類別的大括弧前 1 行  
按一下, 再按 **Enter** 鍵新增一行

# 按一下按鈕就放大顯示的文字

```
activity_main.xml  *MainActivity.java
1 package tw.com.flag.ch02_button;
2
3 import android.os.Bundle;
4
5
6
7 public class MainActivity extends Activity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13    }
14
15    @Override
16    public boolean onCreateOptionsMenu(Menu menu) {
17        // Inflate the menu; this adds items to the acti
18        getMenuInflater().inflate(R.menu.activity_main,
19        return true;
20    }
21
22    int size=30; // 字型大小, 初始值 30(sp)
23 }
24
```

4 加入一個變數用來記錄字型大小

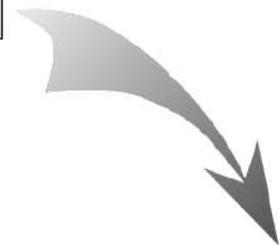
# 按一下按鈕就放大顯示的文字

```
21  
22     int size=30;    // 字型大小, 初始值 30(sp)  
23     public void bigger(View v){  
24 }
```

輸入過程中, 會出現紅色波浪表示錯誤, 稍後再排除

5 輸入方法的定義

6 按 Enter 鍵



# 按一下按鈕就放大顯示的文字

```
21  
22     int size=30;    // 字型大小, 初始值 30(sp)  
23     public void bigger(View v){  
24  
25     }  
26 }
```

ADT 會自動加入  
結尾的大括弧

再次提醒, On Click 屬性所對應的方法, 必須為 public (公開的方法)、void (無傳回值)、有 1 個 View 類別的參數 (參數名稱無限制)。

# 按一下按鈕就放大顯示的文字

## step 9

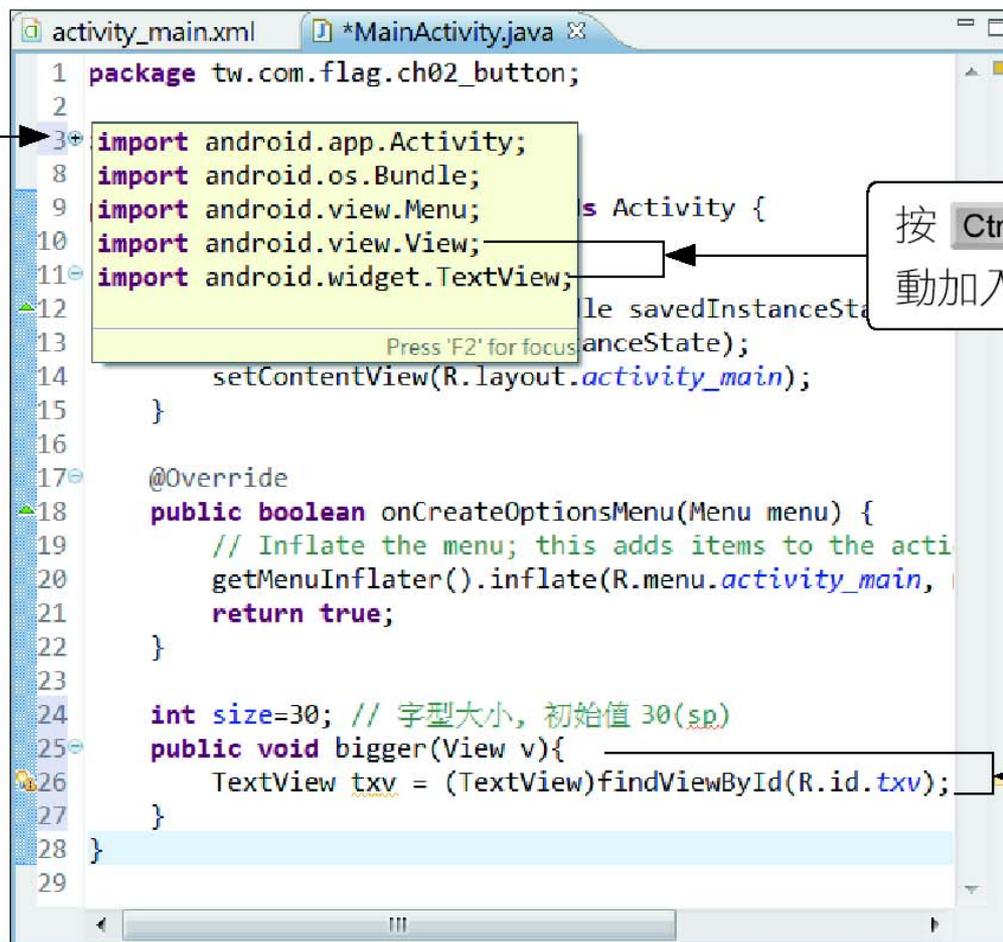
```
22     int size=30; // 字型大小, 初始值 30(sp)
23     public void bigger(View v){
24         TextView txv = (TextView)findViewById(R.id.txv);
25     }
26 }
27
```

1 輸入這一行程式

2 按 **Ctrl** + **Shift** + **O** 鍵

# 按一下按鈕就放大顯示的文字

- 3 將滑鼠移到第 3 行程式前面的加號圖示，會顯示所有的 import 敘述



```
1 package tw.com.flag.ch02_button;
2
3+ import android.app.Activity;
8 import android.os.Bundle;
9 import android.view.Menu;
10 import android.view.View;
11- import android.widget.TextView;
12
13
14 setContentView(R.layout.activity_main);
15 }
16
17 @Override
18 public boolean onCreateOptionsMenu(Menu menu) {
19     // Inflate the menu; this adds items to the acti
20     getMenuInflater().inflate(R.menu.activity_main,
21     return true;
22 }
23
24 int size=30; // 字型大小, 初始值 30(sp)
25 public void bigger(View v){
26     TextView txv = (TextView)findViewById(R.id.txv);
27 }
28 }
29
```

按 **Ctrl** + **Shift** + **O** 鍵會自動加入這 2 行 import 敘述

這 2 行的紅色波浪線都消失了

# 按一下按鈕就放大顯示的文字

## step10

1 按 **Enter** 新增 1 行, 並輸入 "txv."

```
25 public void bigger(View v){
26     TextView txv = (TextView)findViewById(R.id.txv);
27     txv.
28 }
29 }
```

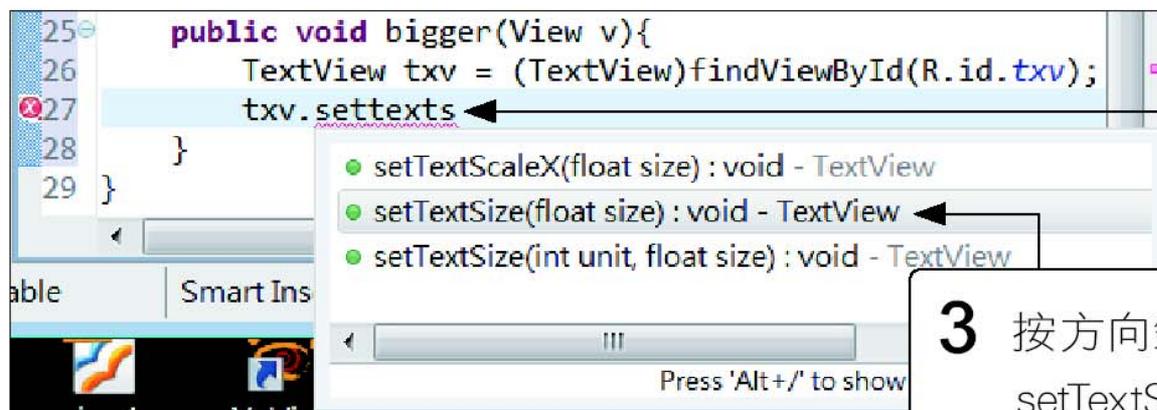
- addChildrenForAccessibility(ArrayList<View> children)
- addFocusables(ArrayList<View> views, int direction)
- addFocusables(ArrayList<View> views, int direction, int focusableMode)
- addOnAttachStateChangeListener(OnAttachStateChangeListener listener)
- addOnLayoutChangeListener(OnLayoutChangeListener listener)
- addTextChangedListener(TextWatcher watcher)
- addTouchables(ArrayList<View> views) : void
- animate() : ViewPropertyAnimator - View
- announceForAccessibility(CharSequence text)
- append(CharSequence text) : void - TextView
- append(CharSequence text, int start, int end) : void - TextView

Smart Ins  
Jump Next | T  
Press 'Alt+/' to show Template Proposals

一輸入小數點時, 就會跳出清單, 列出 TextView 所有可用的方法和常數

# 按一下按鈕就放大顯示的文字

```
25 public void bigger(View v){
26     TextView txv = (TextView)findViewById(R.id.txv);
27     txv.settexts
28 }
29 }
```

A screenshot of an IDE showing a code completion menu. The code in the background is: 

```
25 public void bigger(View v){
26     TextView txv = (TextView)findViewById(R.id.txv);
27     txv.settexts
28 }
29 }
```

The code completion menu is open, showing three options: 

- setTextScaleX(float size) : void - TextView
- setTextSize(float size) : void - TextView
- setTextSize(int unit, float size) : void - TextView

Arrows point from the text 'settexts' in the code to the first option, and from the text 'setTextSize()' in the instruction box to the second option.

**2** 繼續輸入 "settexts"  
以篩選清單內容 (不  
需顧慮大小寫)

**3** 按方向鍵移到我們要用的  
setTextSize(), 再按 **Enter** 鍵

# 按一下按鈕就放大顯示的文字

4 不需理會提示的內容, 直接輸入 "++size"

```
25 public void bigger(View v){
26     TextView txv = (float size) findViewById(R.id.txv);
27     txv.setTextSize(size);
28 }
29 }
```

自動加入的內容

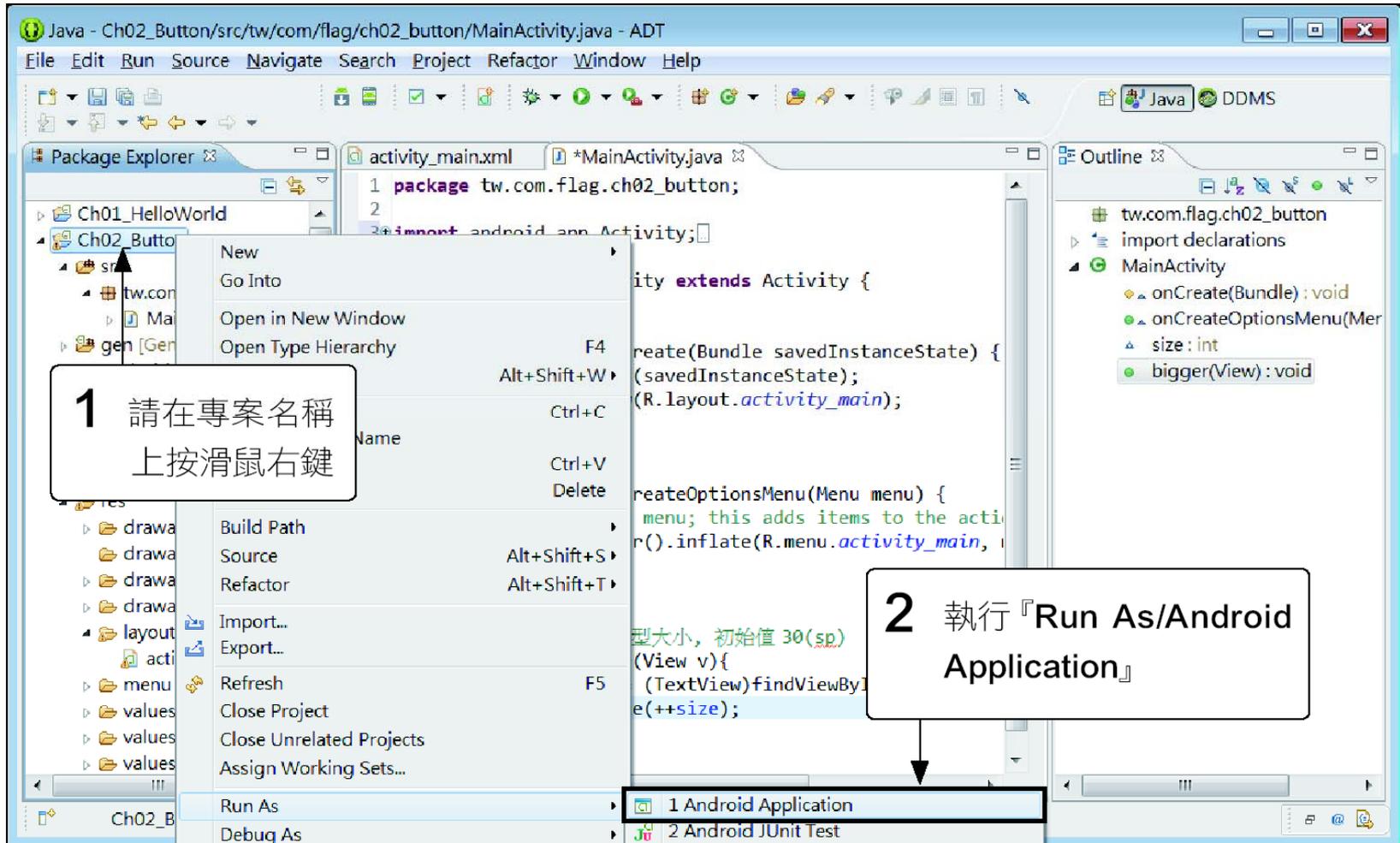
```
25 public void bigger(View v){
26     TextView txv = (TextView) findViewById(R.id.txv);
27     txv.setTextSize(++size);
28 }
```

5 補上分號, 完成自訂方法的內容

表示將 size 變數的值遞增 (加 1), 再指定為新的字型大小

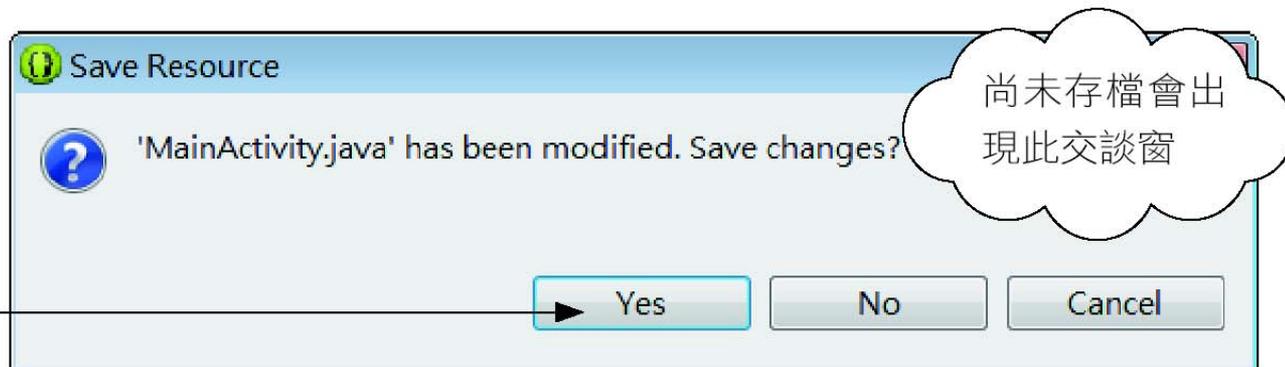
# 按一下按鈕就放大顯示的文字

## step11

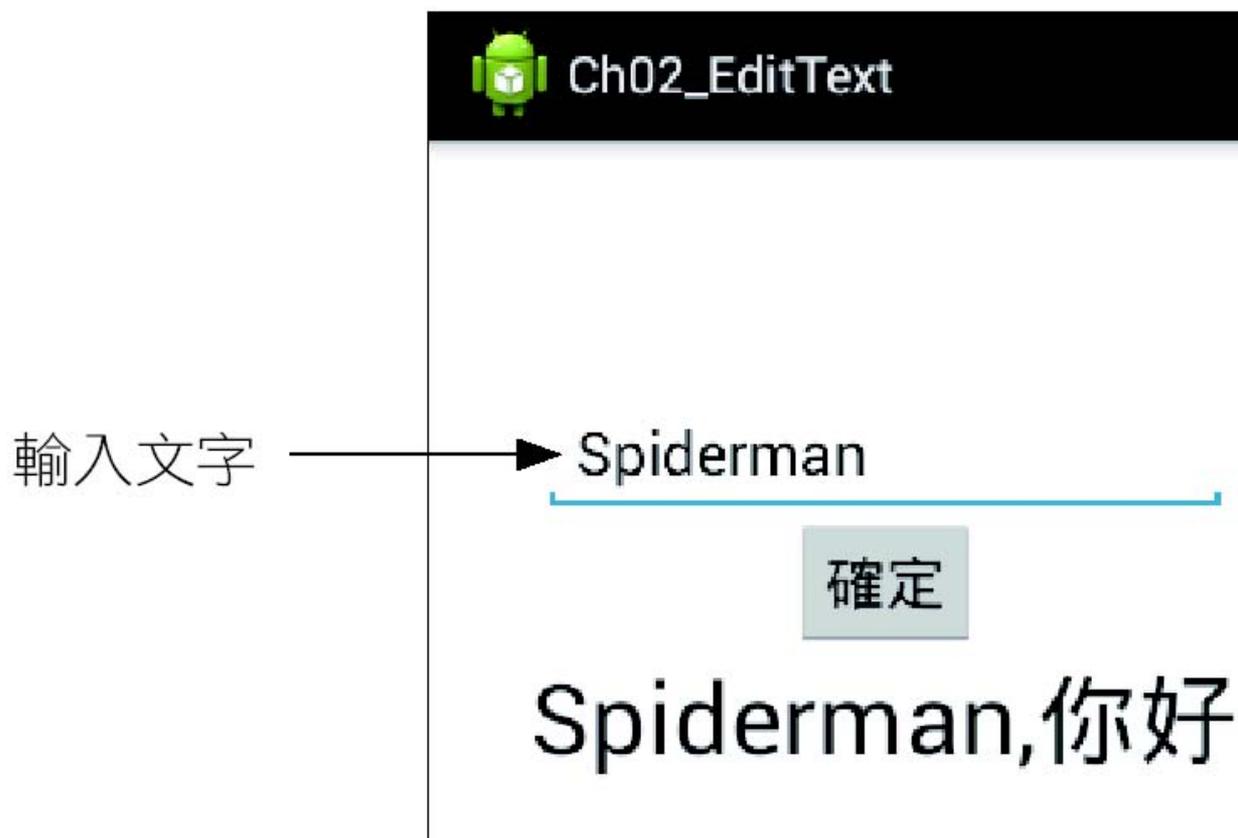


# 按一下按鈕就放大顯示的文字

3 按 Yes 繼續



## 2-6 輸入欄位 EditText 元件



# 輸入欄位 EditText 元件

- `getText()`：取得使用者輸入的文字
- `setText()`：設定 `TextView` 顯示的文字

# getText()：取得使用者輸入的文字

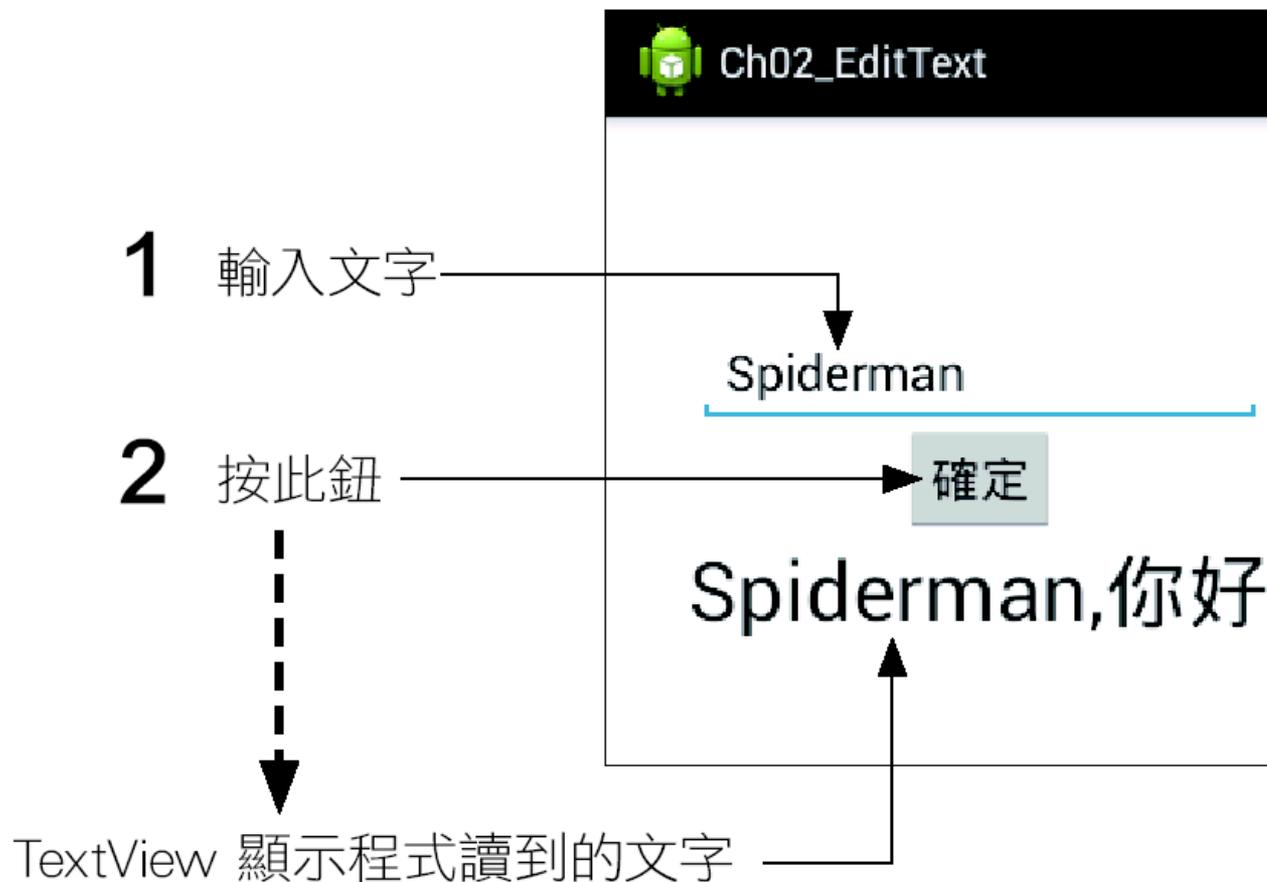
```
EditText edit = (EditText) findViewById(R.id.edit); ←  
                若佈局中的元件 Id 屬性命名為 edit  
String str = edit.getText().toString(); ← 取得文字
```

有在佈局中設定  
Id 屬性 (命名)  
的元件才會自動  
產生資源 ID 喔!

# setText()：設定TextView 顯示的文字

```
txv.setText("您好!");
```

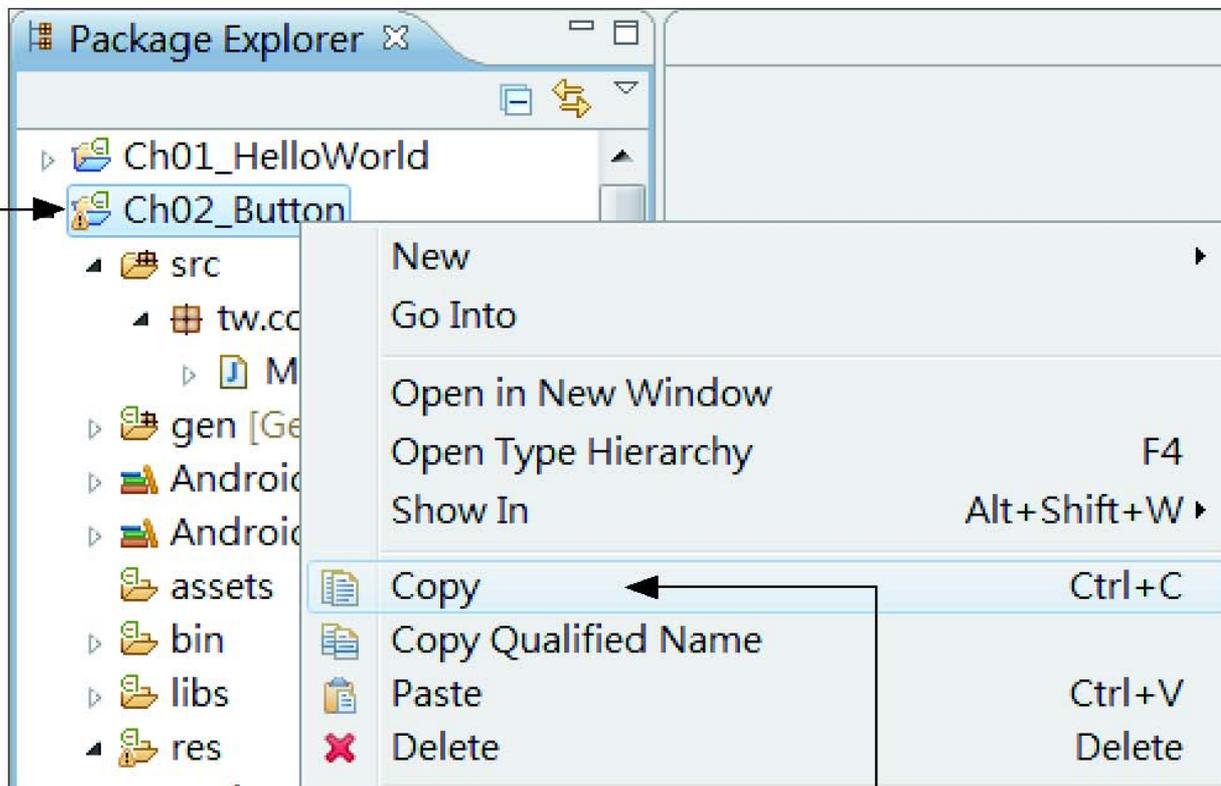
# 範例2-2：加入 EditText 元件



# 加入 EditText 元件

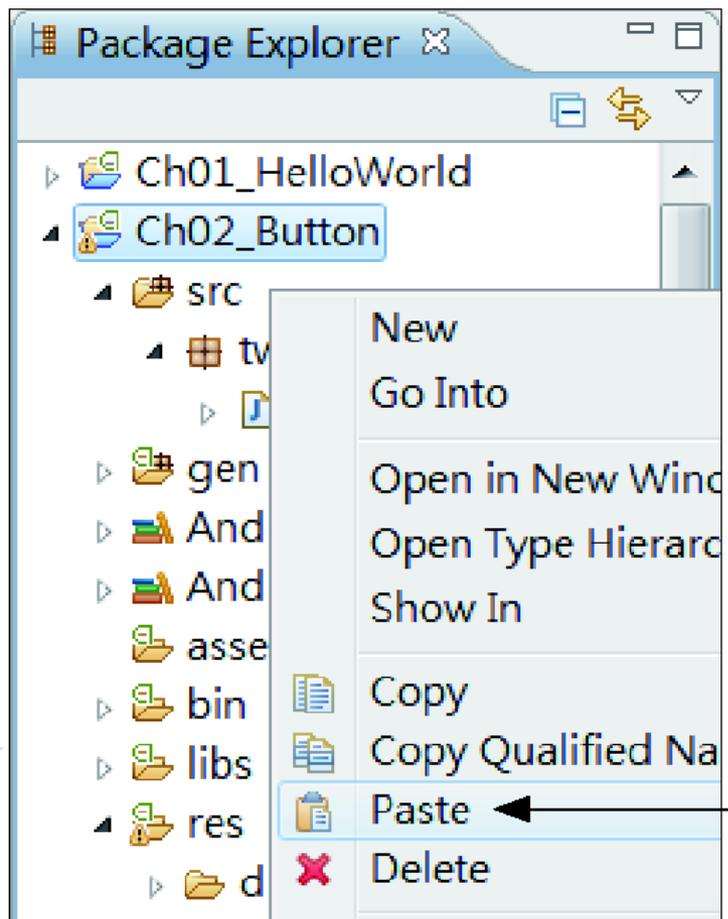
## step 1

專案開啟時，  
才能進行複製的動作



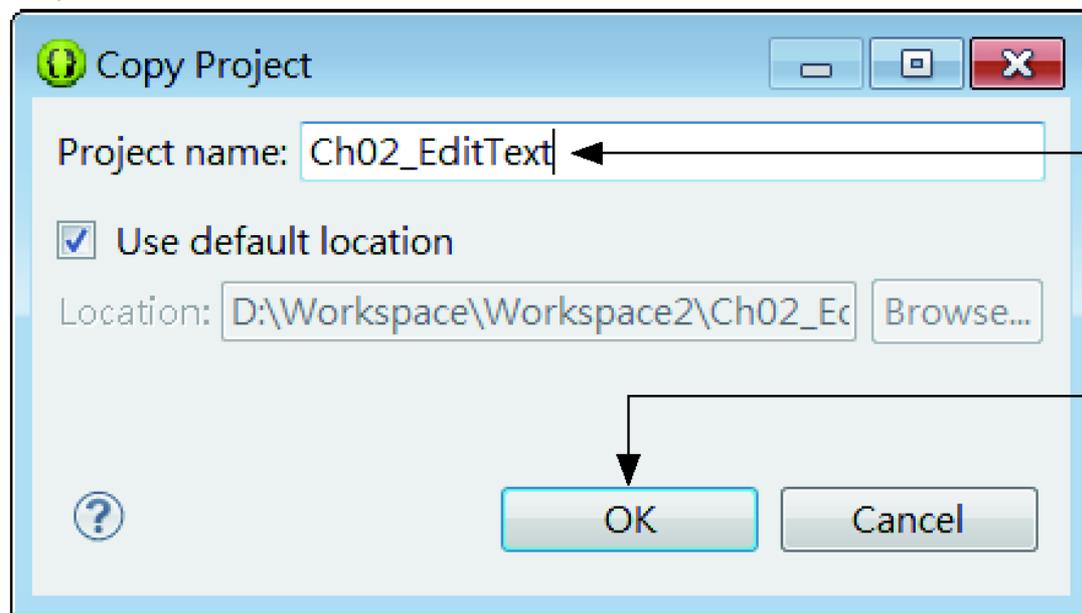
1 在現在專案上按滑鼠右鈕, 執行『Copy』命令

# 加入 EditText 元件



2 在空白處上按滑鼠右鈕，執行『Paste』命令

# 加入 EditText 元件

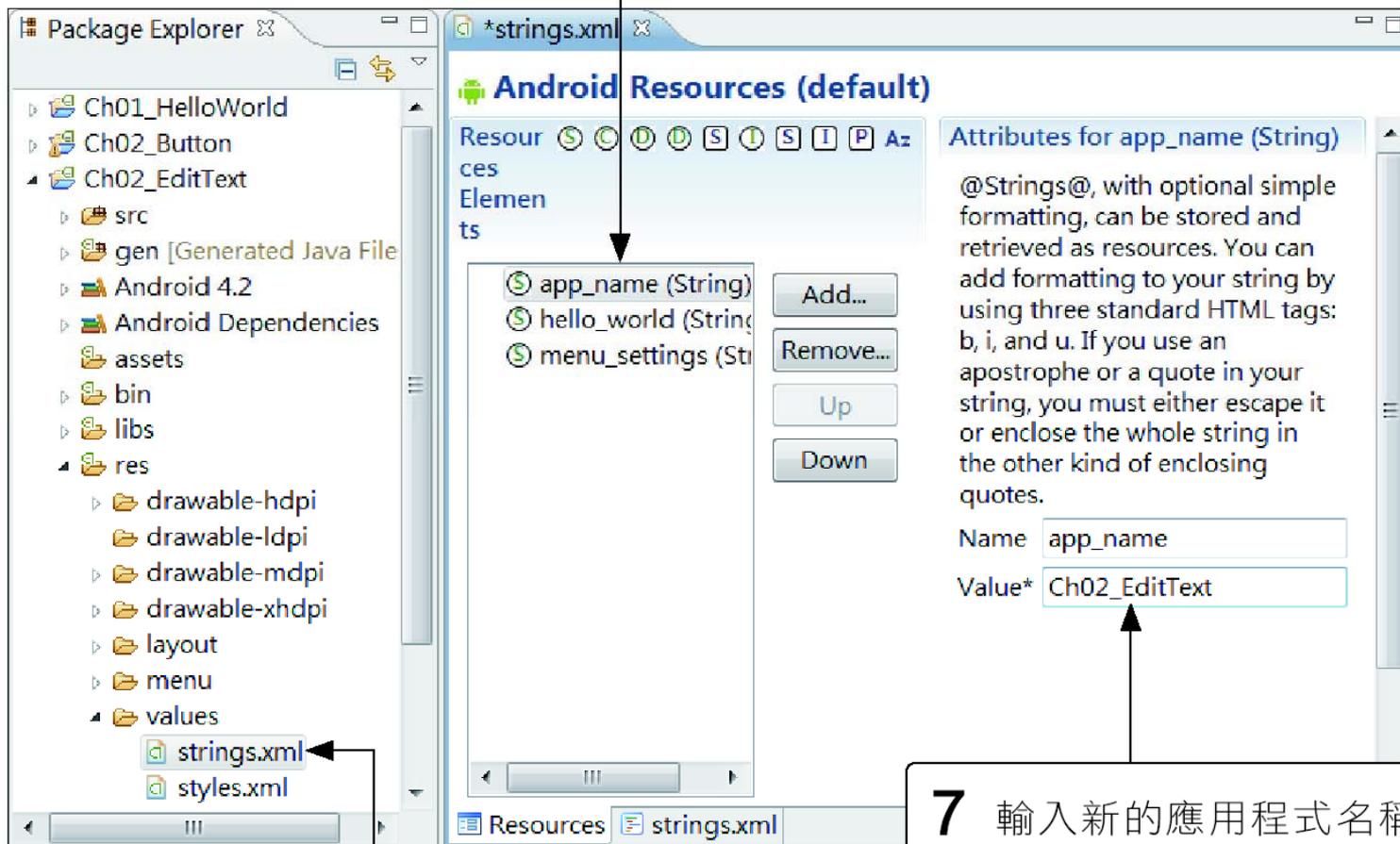


**3** 輸入新專案名稱  
"Ch02\_EditText"

**4** 按 OK 鈕

# 加入 EditText 元件

6 選 app\_name (應用程式名稱)



5 開啟 res/values/strings.xml

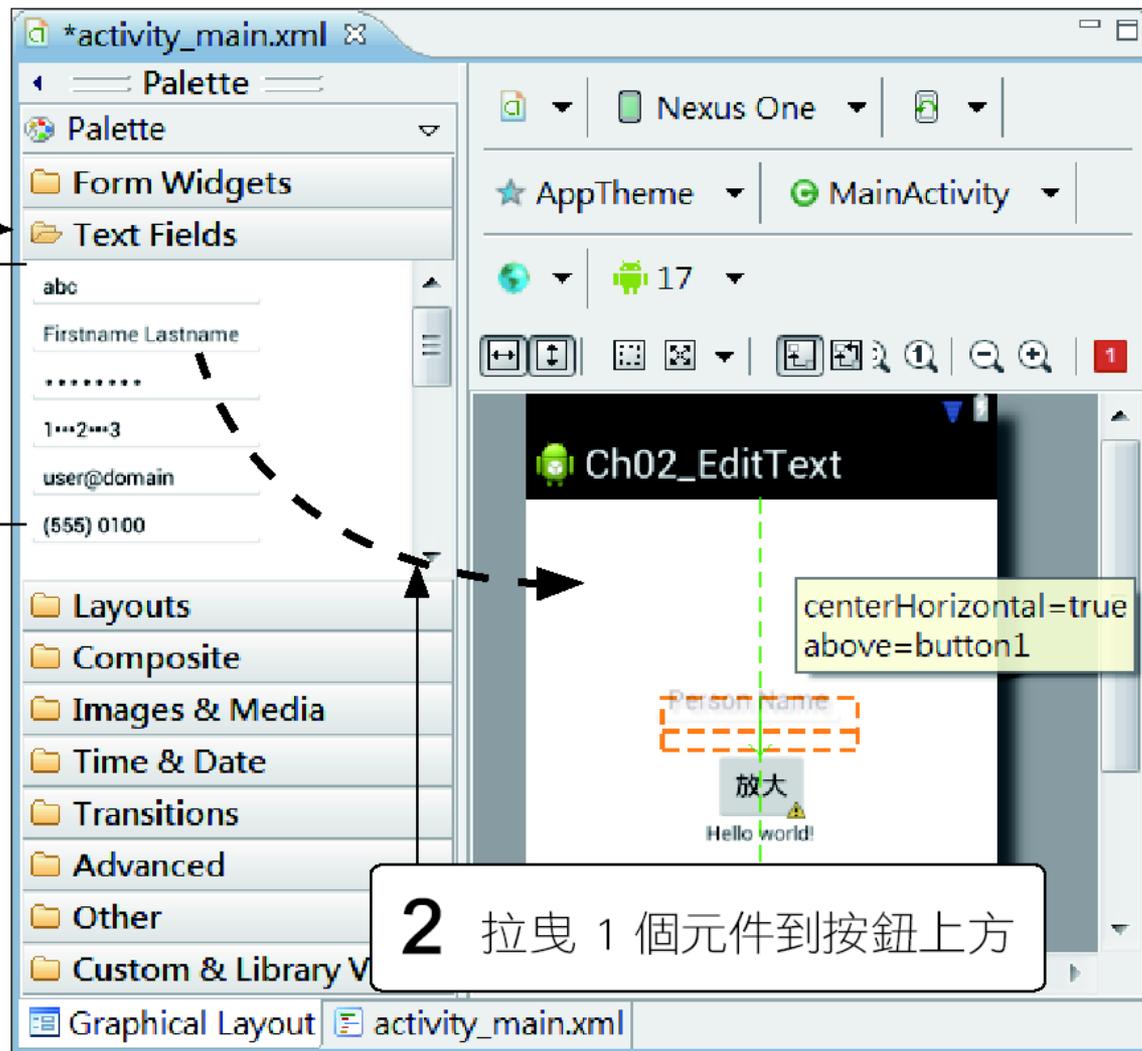
7 輸入新的應用程式名稱並按 **Ctrl** + **S** 存檔

# 加入 EditText 元件

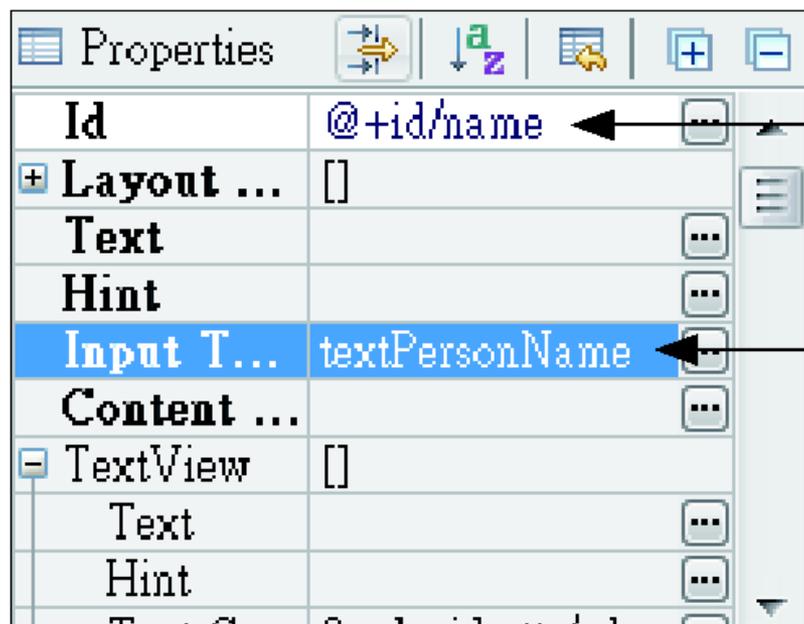
## step 2

1 在 **Text Fields** 項目上按一下展開其內容

這些都是 EditText 元件, 但 Input Type 屬性不同 (參見下圖)



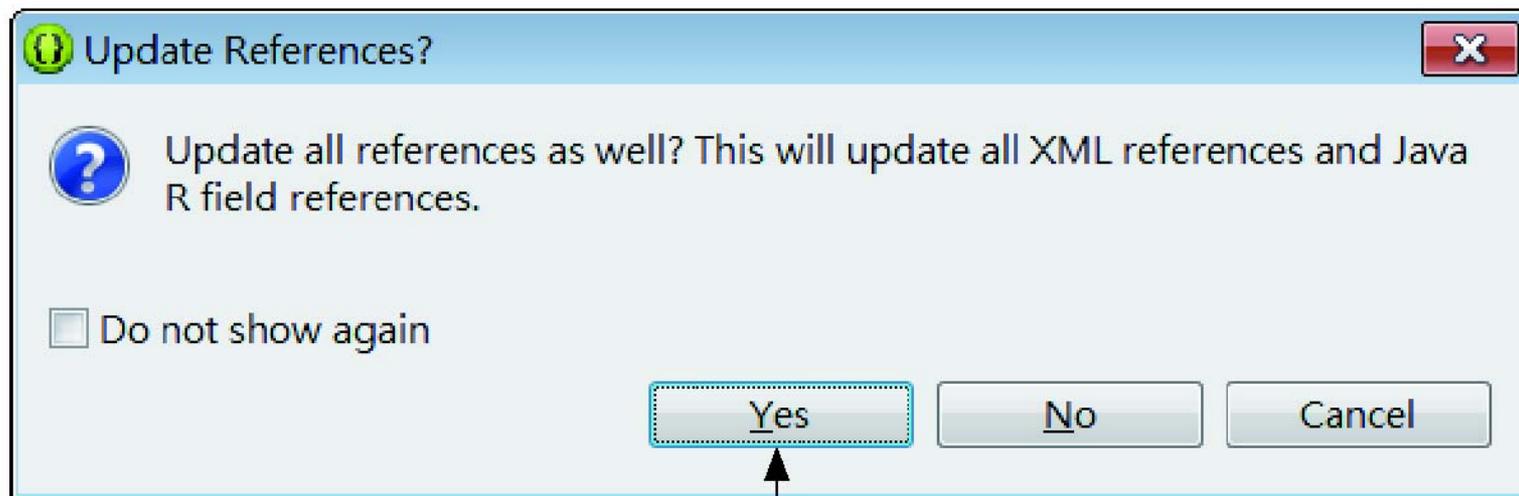
# 加入 EditText 元件



**3** 修改 Id 為 name

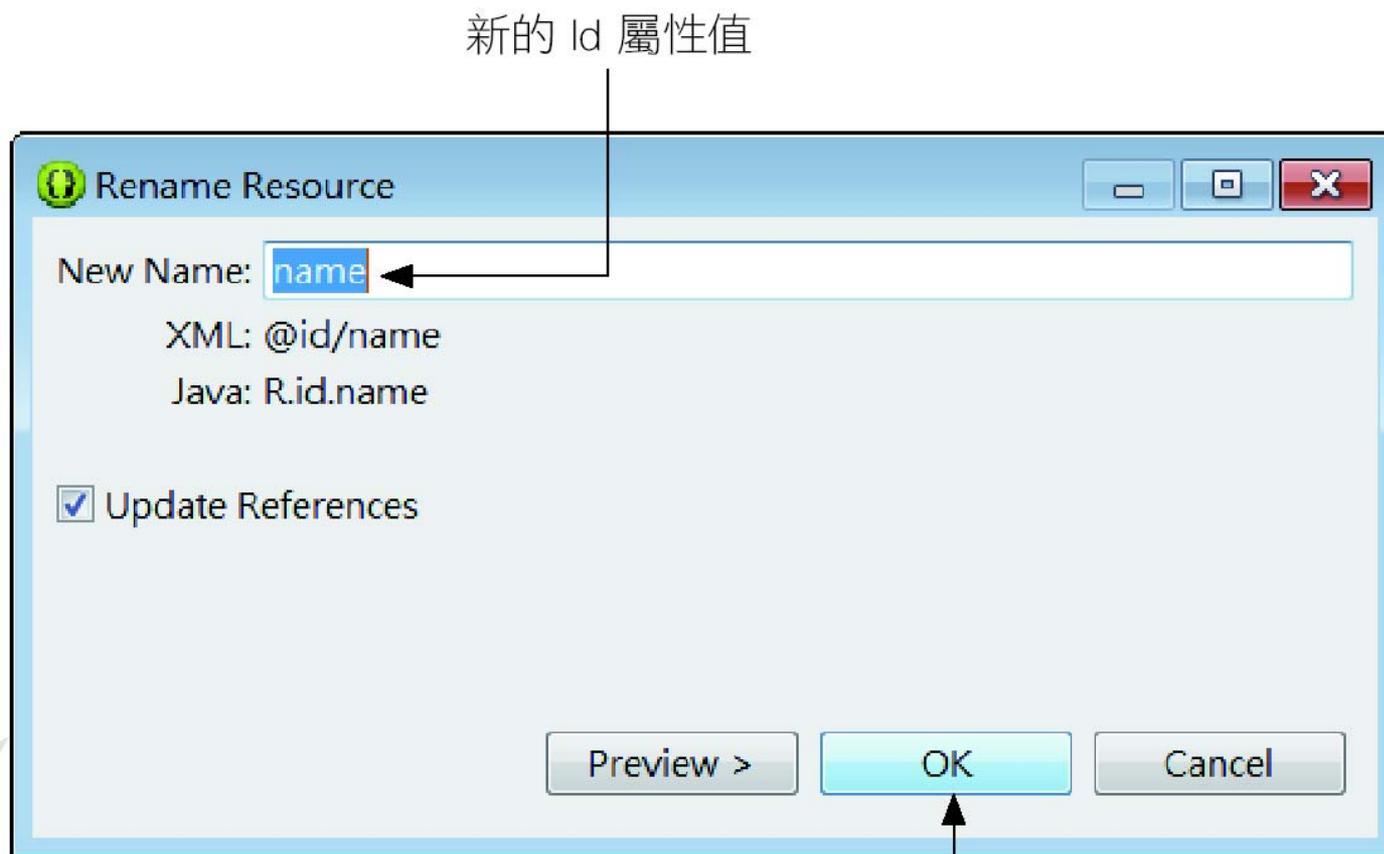
剛剛拉曳加入的項目, 其 Input Type 屬性預設為 "textPersonName", 表示是用於輸入『人名』的欄位

# 加入 EditText 元件



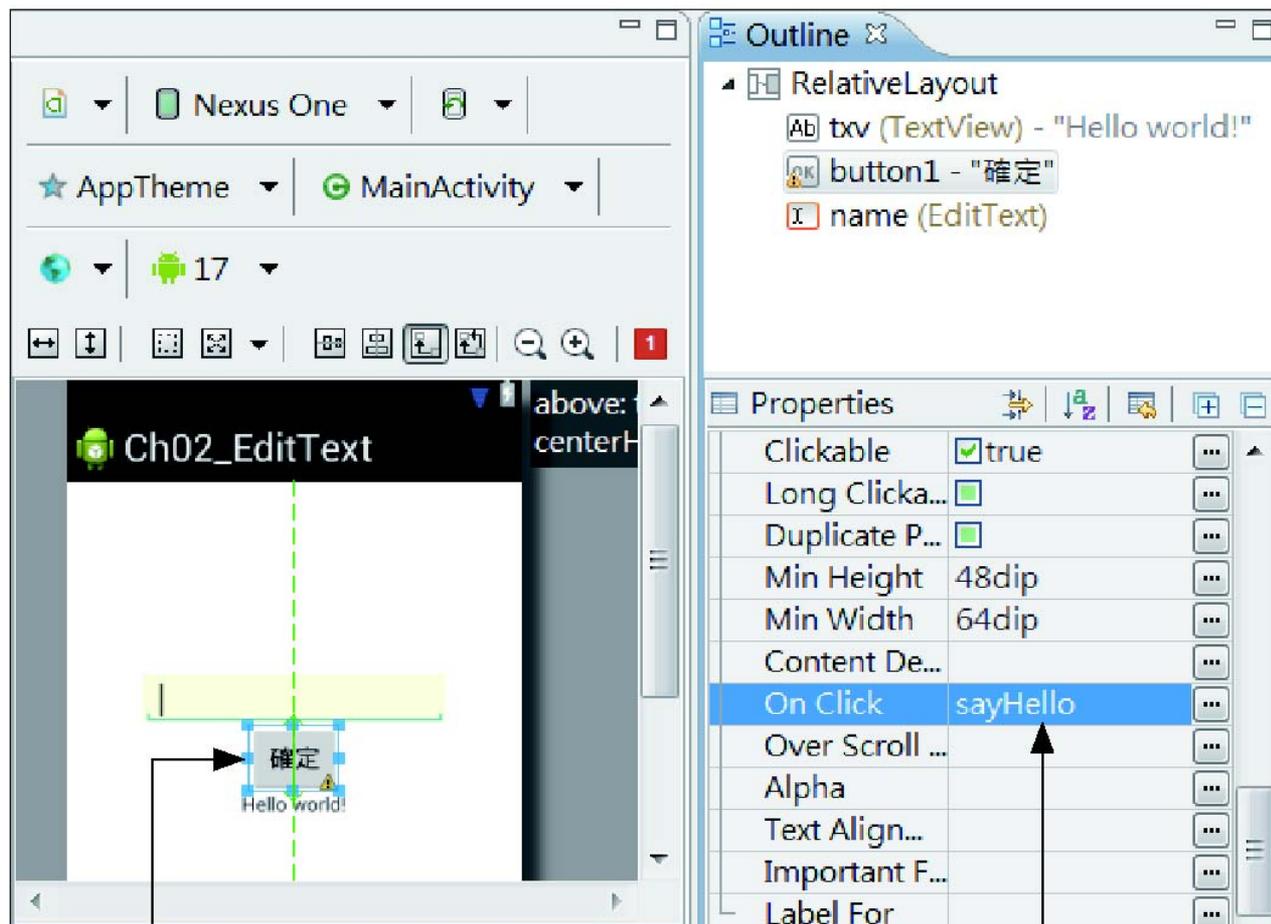
4 按 **Yes** 鈕讓 ADT 自動修改所有參照到舊名稱的地方

# 加入 EditText 元件



5 按 OK 鈕確認修改 Id 值

# 加入 EditText 元件



6 修改按鈕文字

7 設定按鈕的 On Click 屬性為 "sayHello"

# 加入 EditText 元件

## step 3

自訂的 sayHello() 方法

```
25 public void sayHello(View v){
26     // 取得代表佈局中 EditText 及 TextView 的物件
27     EditText name = (EditText)findViewById(R.id.name);
28     TextView txv = (TextView)findViewById(R.id.txv);
29     // 設定 TextView 文字大小為 30sp,
30     // 並將 EditText 文字串接自訂訊息再設定給 TextView
31     txv.setTextSize(30);
32     txv.setText(name.getText().toString() + ", 您好!");
33 }
34 }
```

在輸入程式的過程中，可按 **Ctrl** + **Shift** + **O** 加入必要的 import 敘述匯入所需的類別

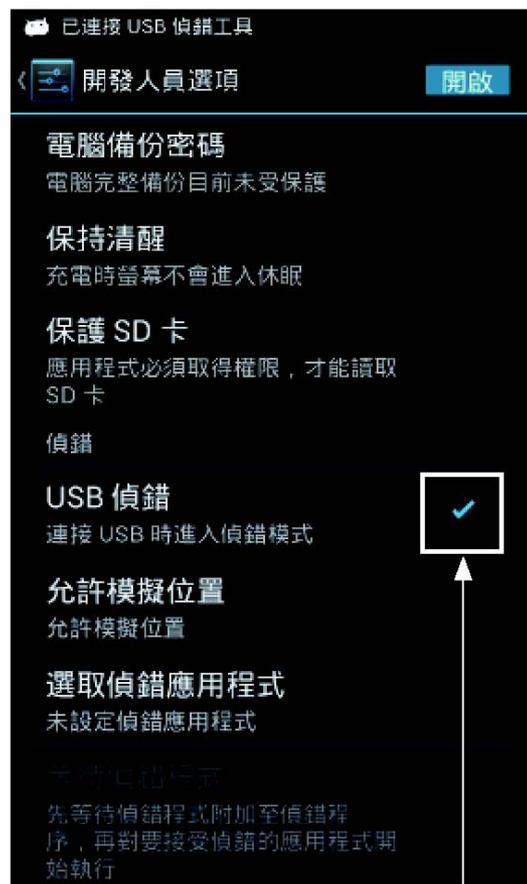
## 2-7 使用USB 線將程式部署到手機上執行

- 開啟手機偵錯功能
- 透過 USB 將 Android App 傳送到手機安裝與執行
- 執行已安裝的程式
- 修改專案的套件名稱

# 開啟手機偵錯功能

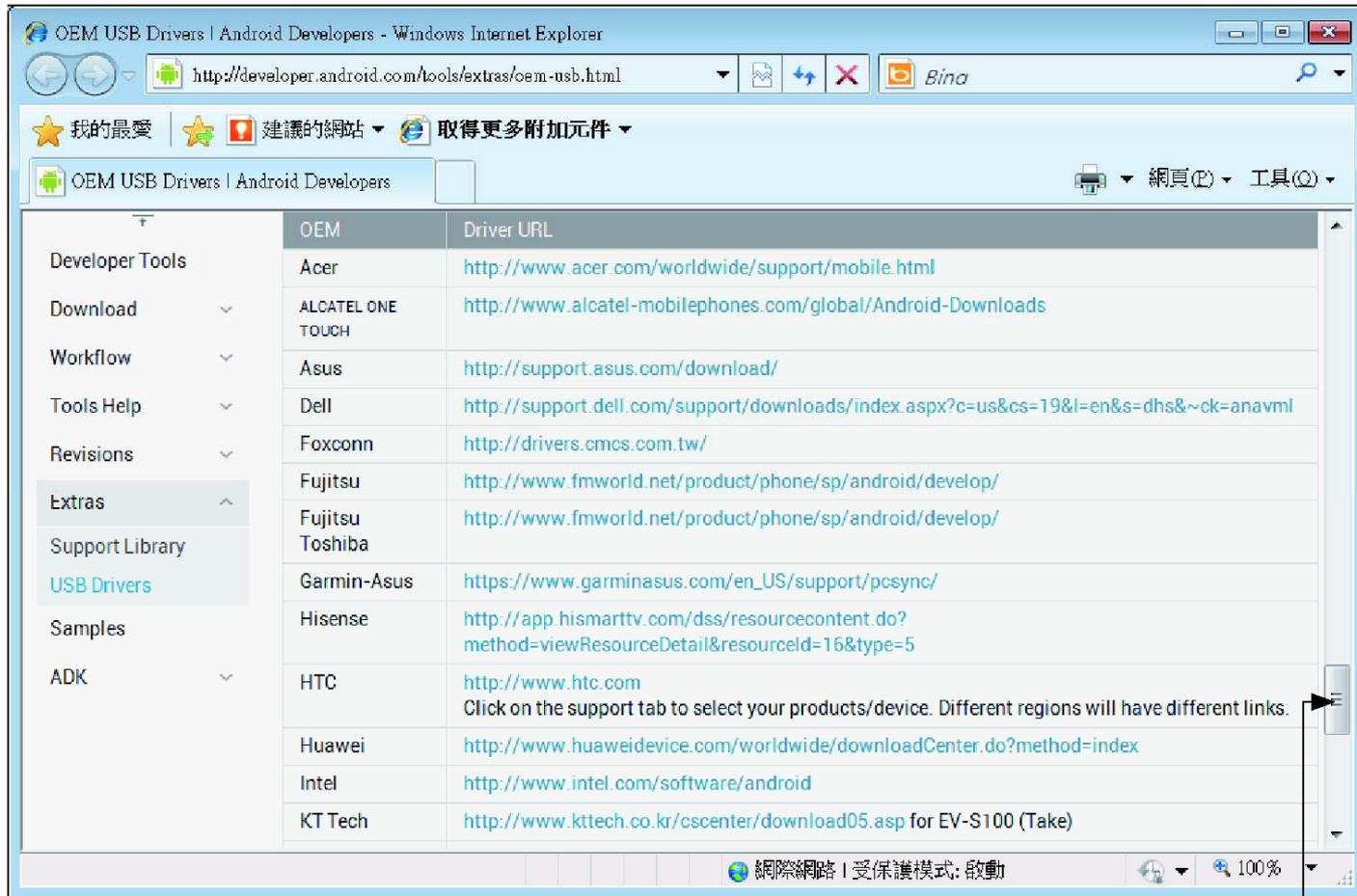


- 1 在設定項目清單中，選開發人員選項項目



- 2 勾選 USB 偵錯並按確定鈕，啟用此設定表示允許 ADT 透過 USB 將程式傳送到手機

# 透過 USB 將 Android App 傳送到手機安裝與執行



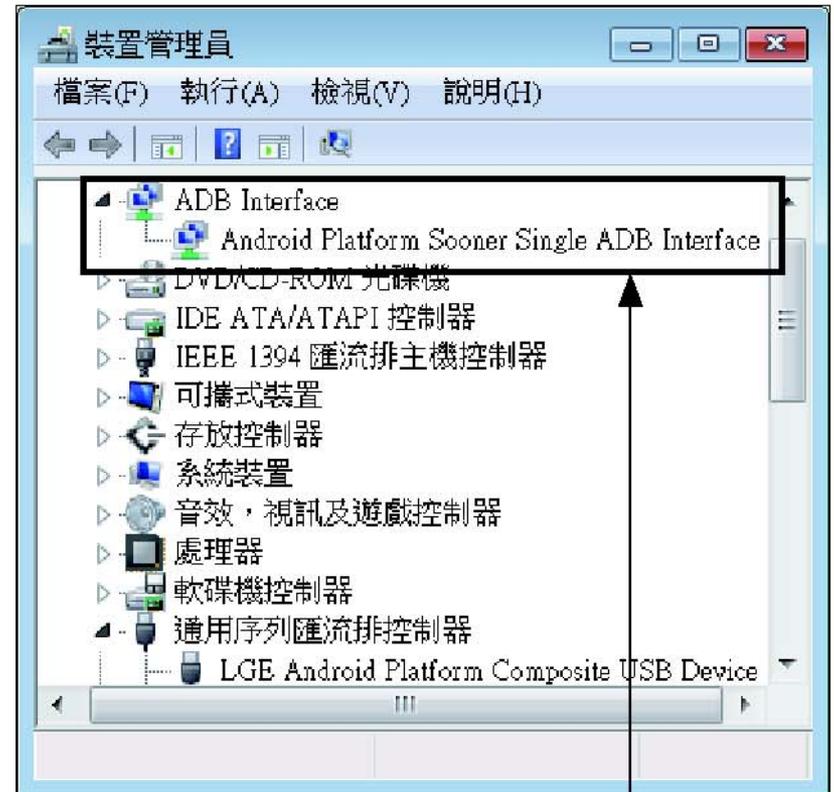
將網頁向下捲動, 即可看到各廠商的下載網址連結

# 透過 USB 將 Android App 傳送到手機安裝與執行

不同手機裝置，  
其出現的位置  
也可能不同



本例為 Asus PadFone2 手機

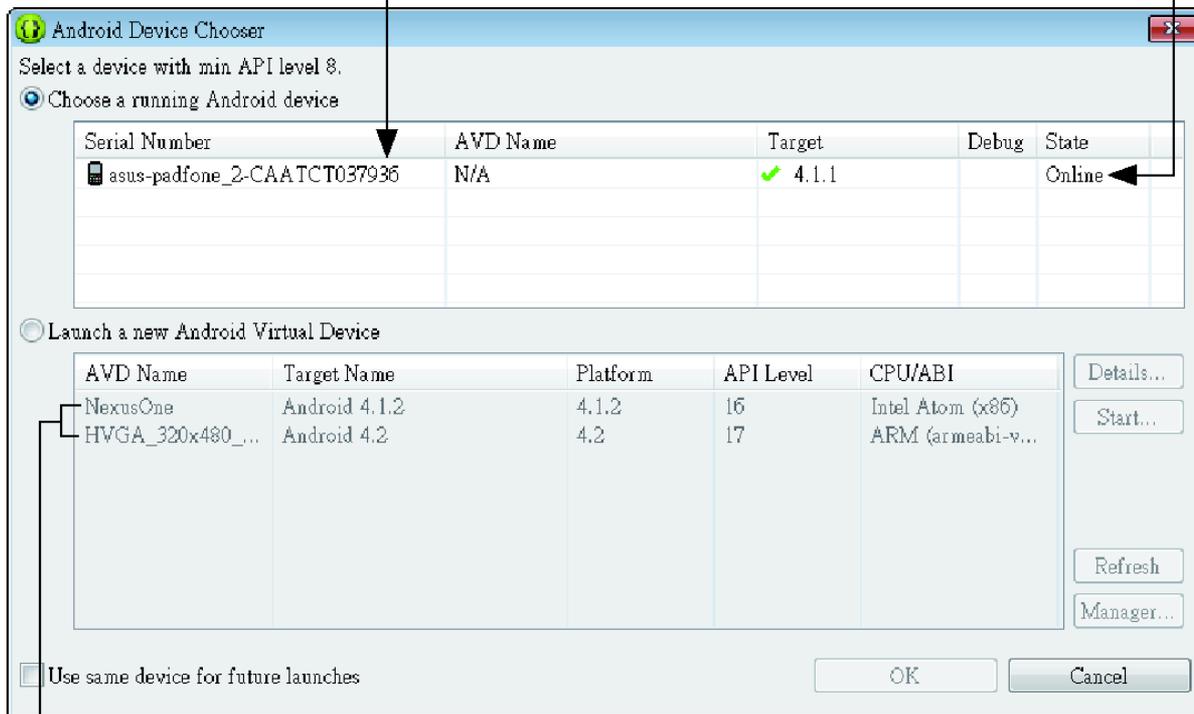


本例為 LG 手機

# 透過 USB 將 Android App 傳送到手機安裝與執行

雙按手機裝置名稱即可將程式安裝到手機上, 且會自動執行

若此處顯示 "Offline", 請將連接手機的 USB 線拔除, 再重新接回, 待顯示 "Online" 才能上傳程式



稍等一下, 就可在手機螢幕上看到程式的執行畫面了

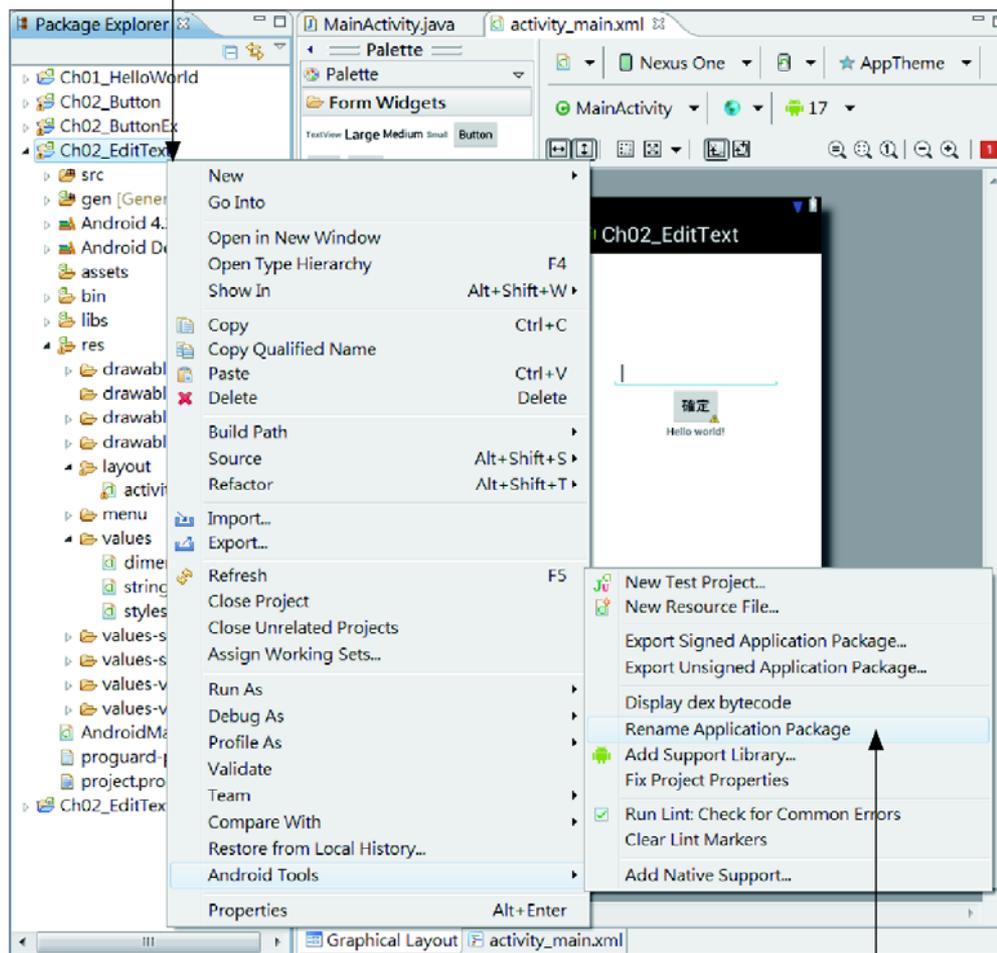
此為尚未啟動的模擬器, 若模擬器已啟動, 也會列在視窗上半的清單

# 執行已安裝的程式



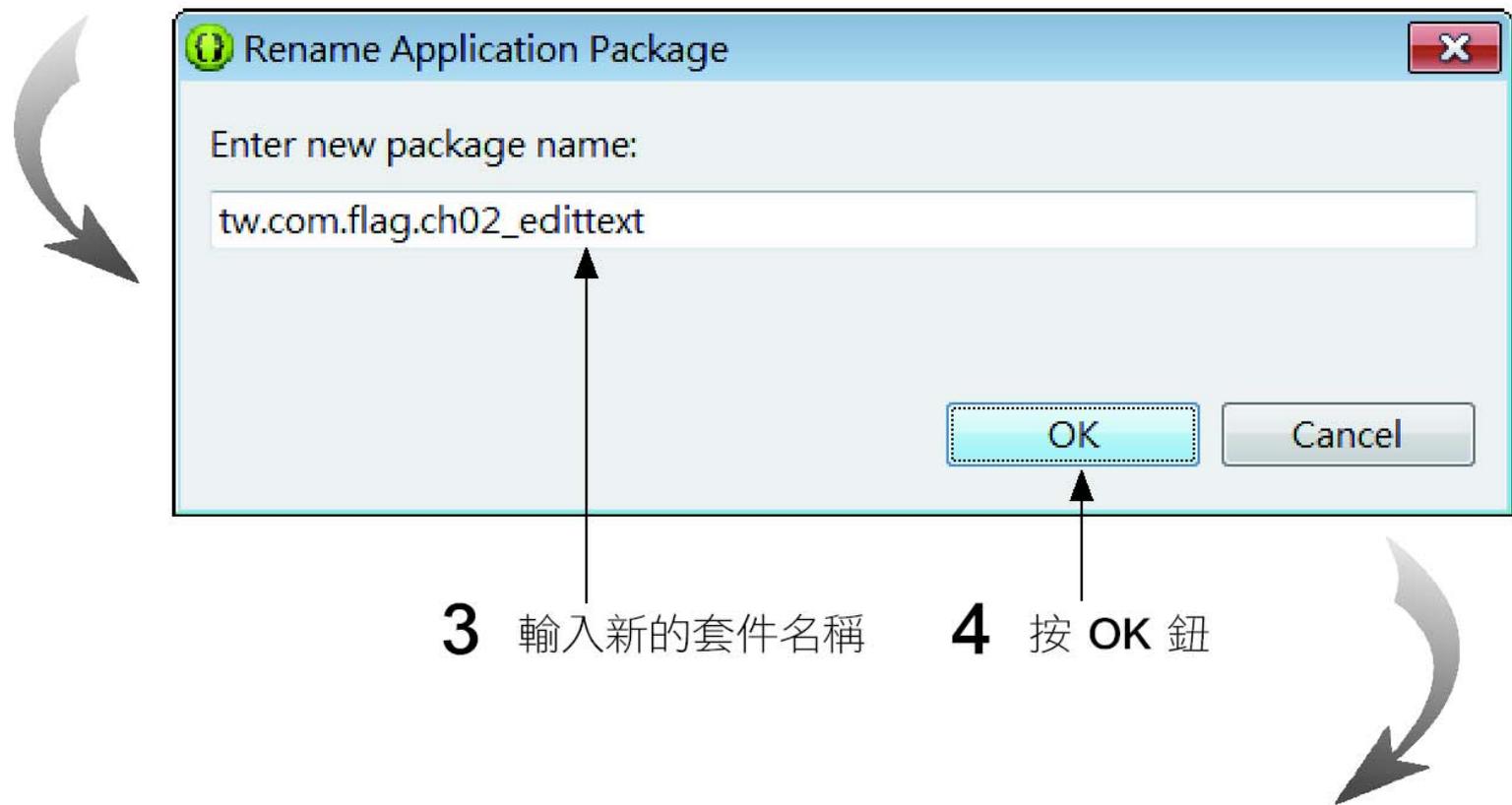
# 修改專案的套件名稱

1 在複製產生的新專案上按滑鼠右鈕

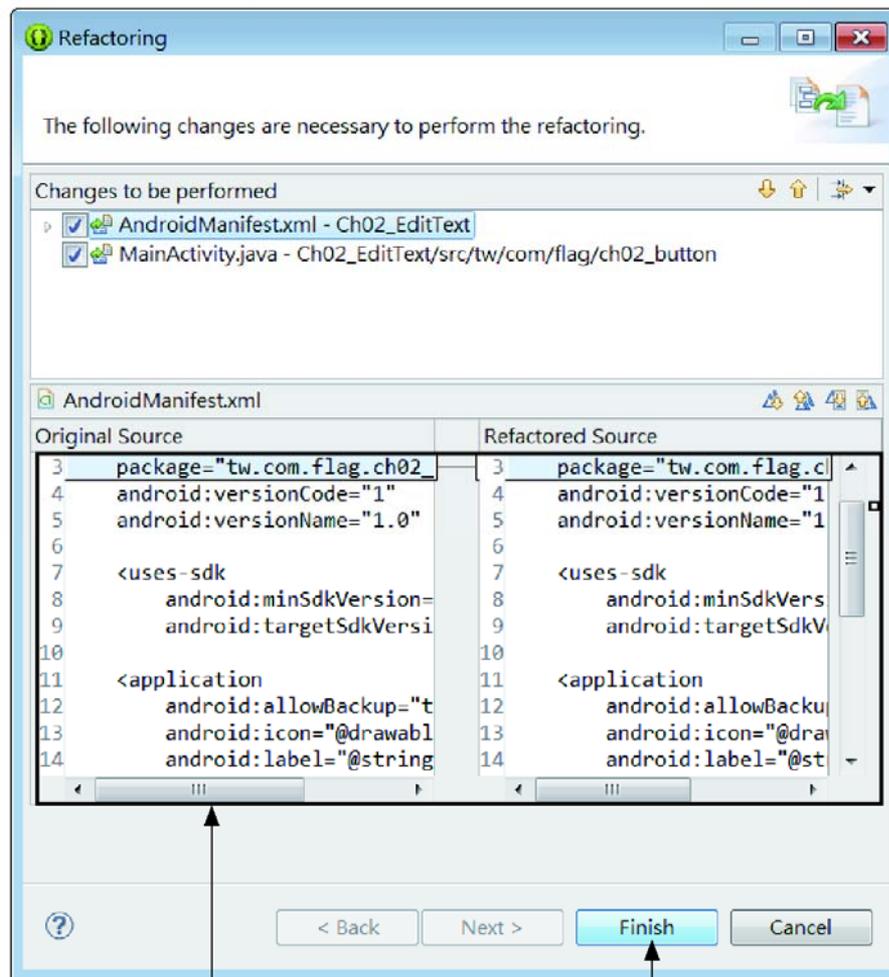


2 執行『Android Tools/Rename Application Package』命令

# 修改專案的套件名稱



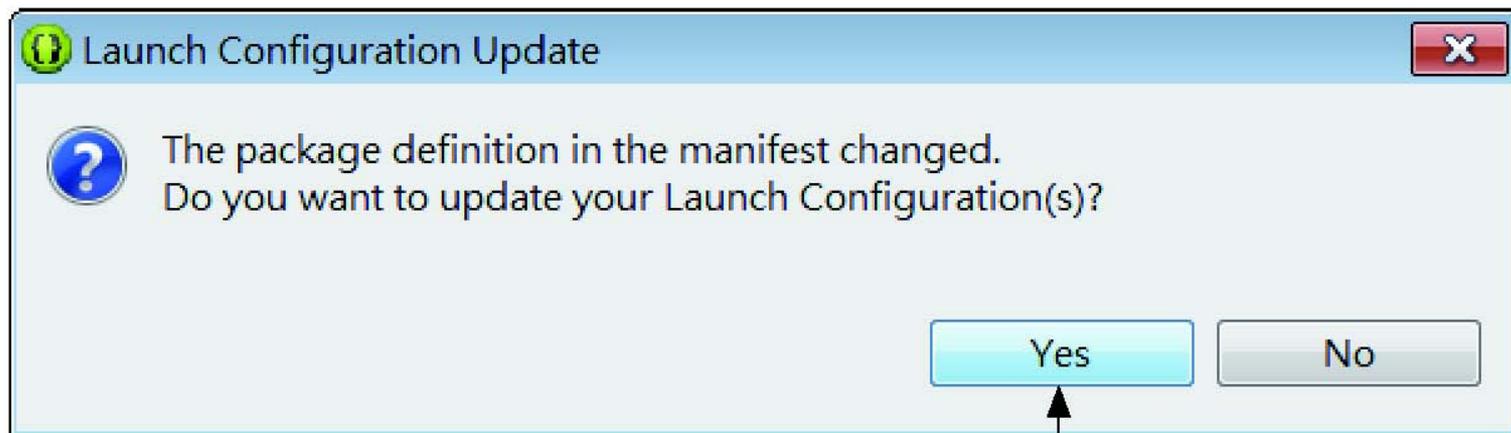
# 修改專案的套件名稱



此處可預覽檔案變動的內容

5 按 Finish 鈕

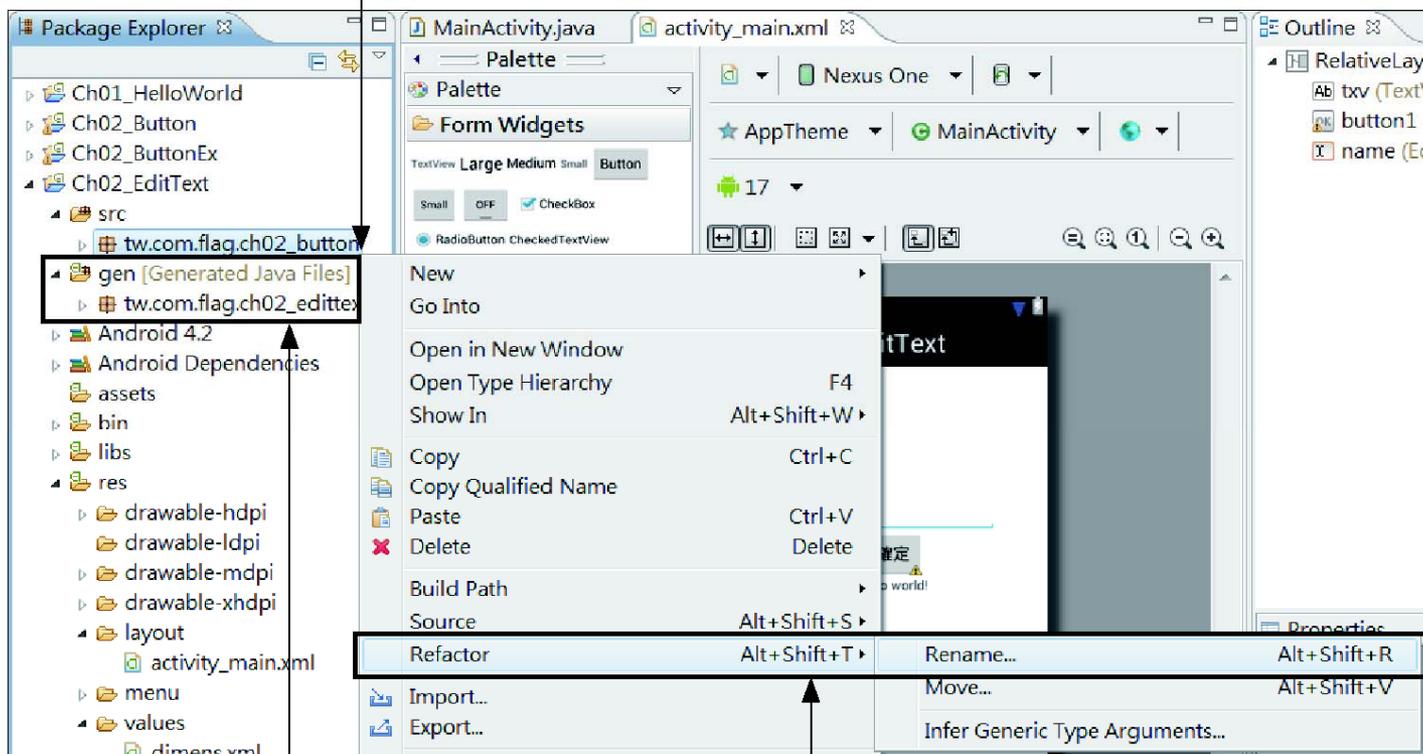
# 修改專案的套件名稱



6 按 Yes 鈕

# 修改專案的套件名稱

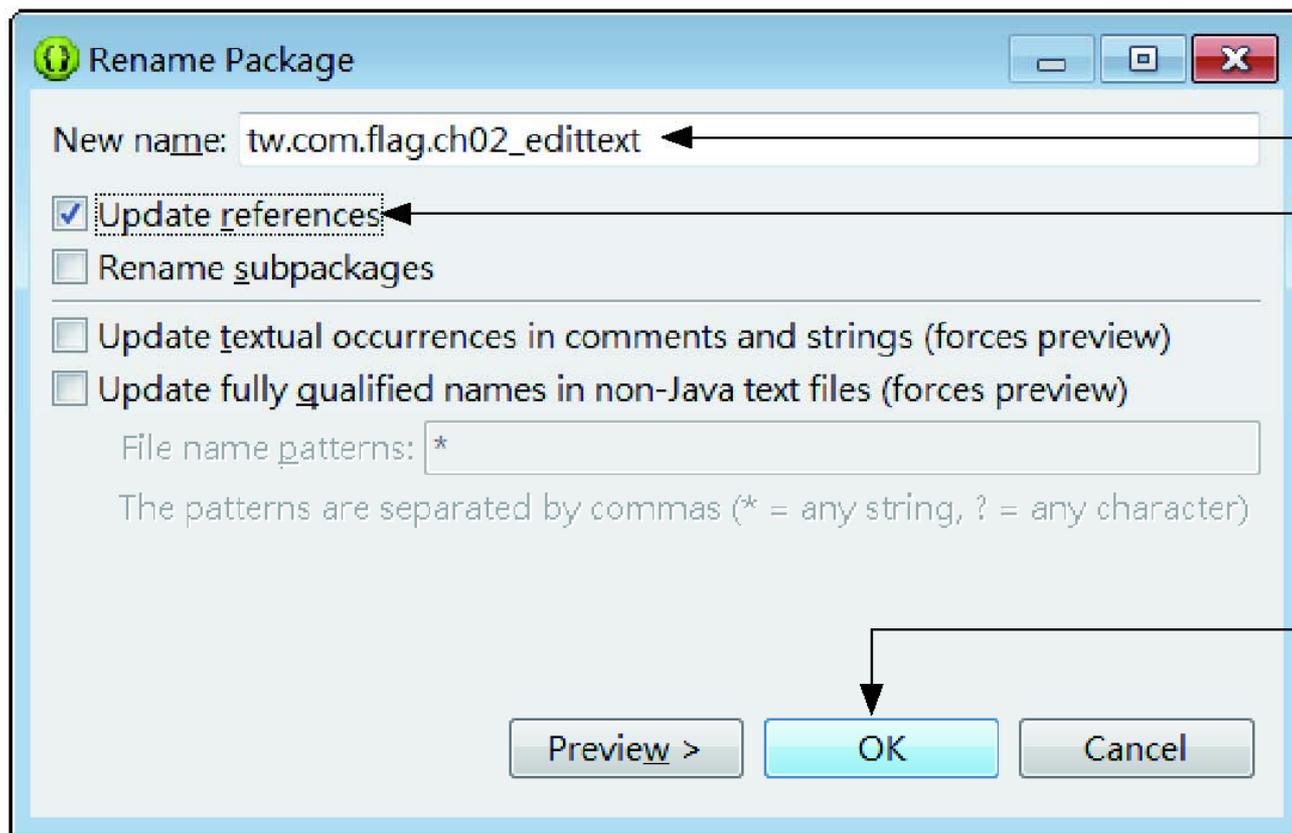
1 展開 src 資料夾, 在套件名稱上按滑鼠右鈕



gen 資料夾是由 ADT 維護, 所以其內的套件名稱已更新

2 執行『Refactor/Rename』命令

# 修改專案的套件名稱

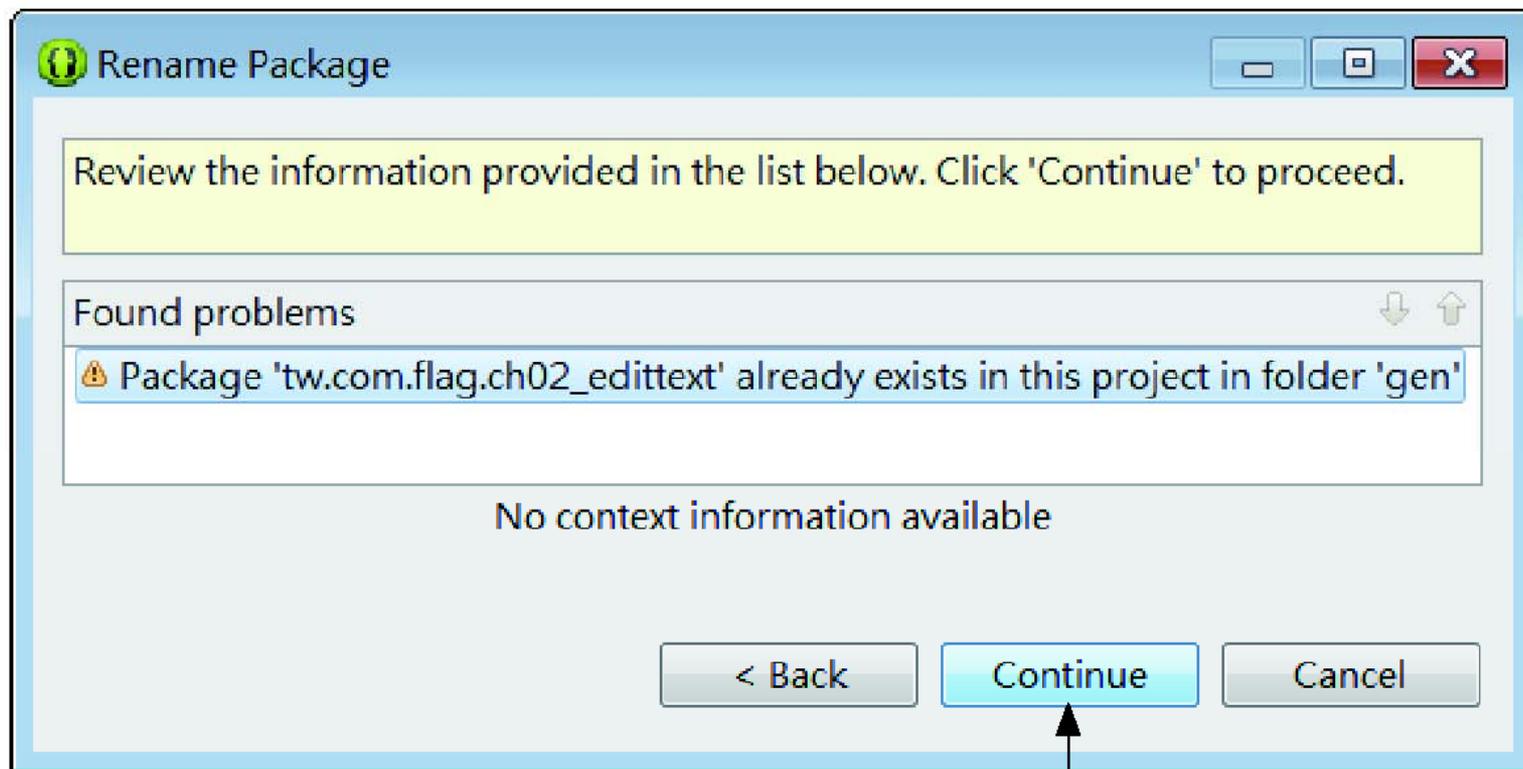


3 輸入新的套件名稱

4 只需勾選第 1 項

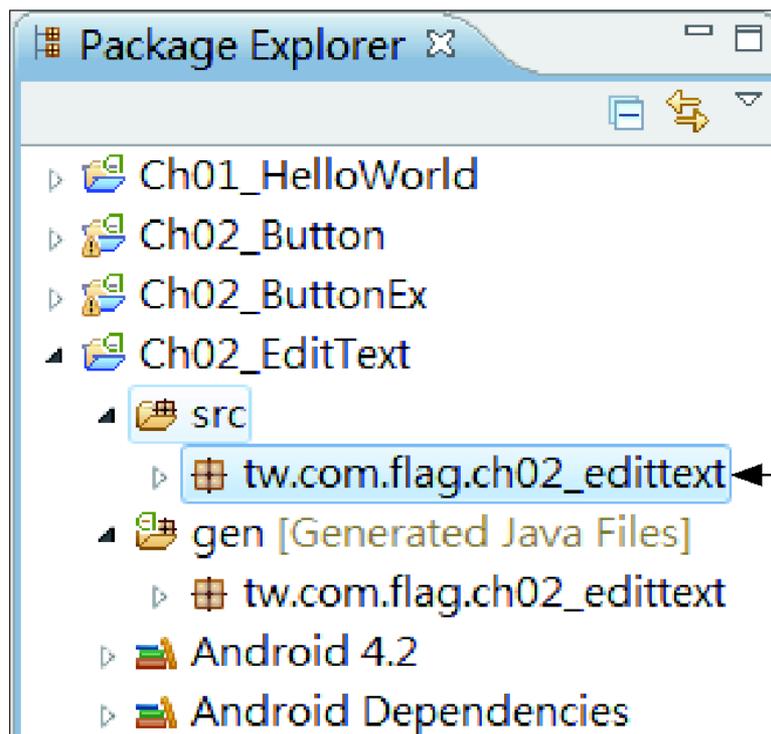
5 按 OK 鈕

# 修改專案的套件名稱



6 按 **Continue** 鈕確認修改套件名稱

# 修改專案的套件名稱



← 修改完成