

# Android Tutorial using Kotlin 第二堂

## （2）設計應用程式使用者介面

[Android Tutorial using Kotlin 第二堂（1）規劃與建立應用程式需要的資源 << 前情](#)

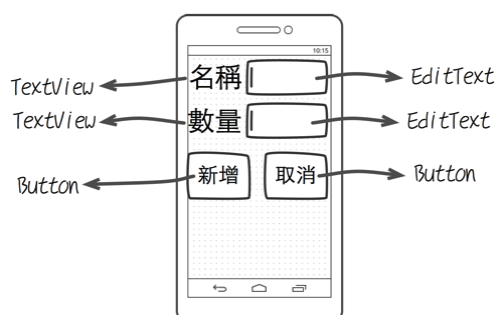
行動裝置提供各種不同應用的程式，讓使用者可以隨時執行一些工作、瀏覽網頁和玩一些遊戲。因為螢幕尺寸的關係，它不會像一般個人電腦上的應用程式，需要設計一些很複雜的操作畫面。透過觸控螢幕的操作，行動裝置應用程式提供簡單與直覺的操作畫面，通常不需要使用說明書，就可以讓使用者順利的使用應用程式。

Android是一個開放的作業系統，這表示所有廠商都可以設計與製造各種使用Android作業系統的行動裝置，這些裝置的螢幕內建的設備，並不是固定的。尤其是螢幕的尺寸和解析度，會讓Android應用程式的畫面設計，跟其它技術比較不一樣。一個好的應用程式，在螢幕是3.4吋、解析度是480X800的裝置上執行，還有在螢幕是4.7吋、解析度是1920X1080的裝置上執行，畫面的畫面看起來應該是一樣的。

Android應用程式的畫面設計，採取一種比較靈活的方式，作法也跟其它技術比較不一樣。這一章說明設計應用程式畫面的重與方式，學習使用各種Android畫面控制項和版面配置元件。

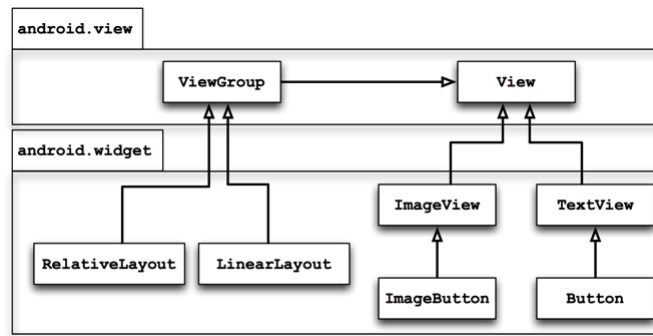
### 6-1 設計使用者介面

一般的Android應用程式，通常需要提供一些畫面，讓使用者執行操作或瀏覽資料。應用程式的畫面是使用一些Android API中畫面物件組合起來的，這些畫面API主要在「android.widget」套件下，這個套件提供各種畫面元件的類別，例如在應用程式畫面一個按鈕，就是一個「Button」類別的物件。認識基本的畫面元件以後，在規劃與設計應用程式的畫面時，你就可以製作像這文件資料：



在上面的規劃資料裡面，你可以決定使用哪些畫面元件，例如用來顯示文字的「TextView」元件，或是讓使用者輸入文字用的「EditText」元件。針對整個畫面的安排，你也會決定控制元件排列方式的畫面配置元件，例如這個畫面適合使用「RelativeLayout」這種畫面配置元件。像按鈕或輸入這類元件，需要在程式碼中設計按鈕的工作與讀取輸入的內容，所以我們取一個名稱，這個工作也可以在規劃應用程式的畫面時就決定好元件的名稱，在設計畫面與撰寫程式的時候，會使用這些元件進行一些設定。

跟一般其它技術的畫面元件類似，Android跟畫面相關的元件在「android.view」和「android.widget」兩個套件中。下圖顯示的套件和部份的元件類別：



上面的類別圖形只有顯示少數幾個元件，在android.view套件下的「View」類別，是所有畫面元件的父類別。同樣在這個套件「ViewGroup」是所有畫面配置元件的父類別。它們各自有很多子類別，提供各種畫面和配置元件。大部份的Android應用程式，應該使用XML格式的畫面配置檔為Activity元件設計畫面，它放在「res\layout」目錄下，這是在這裡說明的主要設計方式

## 6-2 畫面元件的基本設定

Android的畫面元件是View和ViewGroup的子類別，提供各種應用程式畫面需要的元件。在畫面配置檔中設計應用程式的畫面用與這些與元件類別名稱一樣的標籤，再依照需要加入一些設定，所以跟程式設計的作法很不一樣。畫面配置檔一定要放在「res/layout」目錄，檔案名稱必須是小寫的字母和底線，副檔名一定是「.xml」。一個基本的畫面配置檔內容會像這樣：

```

<?xml version="1.0" encoding="utf-8"?>
<!-- 使用Button標籤，設定整個畫面只有一個按鈕元件 -->
<!--
    xmlns:android="..." 這個設定一定要加在第一個標籤，
    後面還有其它標籤的話，就不用再加入這個設定
-->
<Button xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:text="Hello! GUI!" />
  
```

這個畫面配置檔只有在畫面中加入一個按鈕元件，它使用的是「Button」標籤，在API裡面也有一個名為「Button」的類別。好需要的畫面配置檔，就可以在Activity元件的onCreate函式中，呼叫「setContentView」指定元件使用指定的畫面資源：

```

class MainActivity : AppCompatActivity() {

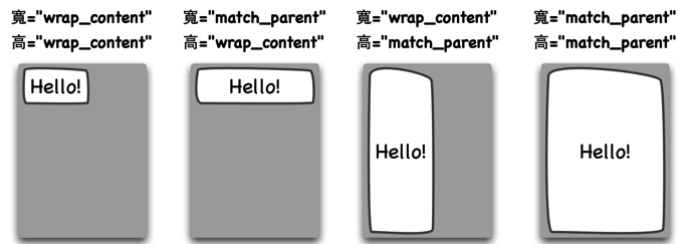
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // 設定畫面配置資源
        // 指定的參數在「R.layout.»後面是檔案名稱
        setContentView(R.layout.activity_main)
    }

}
  
```

不管是設計一個簡單或複雜的畫面，對放到畫面中的元件，都有一些基本而且通用的設定，在大部份的情況下，一定要為畫面加入下列決定大小的設定：

- `android:layout_width`：設定畫面元件的寬度
- `android:layout_height`：設定畫面元件的高度

因為各種Android裝置的螢幕尺寸與解析度並不是一樣的，所以在設定畫面元件的大小時，應該要設定為「`matchparent`」或「`wrapcontent`」。例如把元件的寬度設定為`matchparent`時，這個畫面元件的寬度就會佔滿所用的空間。設定為`wrapcontent`的話，會根據這個畫面元件自動調整為足夠顯示內容的空間。如果畫面中只有一個按鈕元件，不同的設定組合會有不同的效果



這樣的設定方式，讓應用程式的畫面在不同尺寸與解析度的螢幕顯示的時候，看起來會是一樣的。你也可以為畫面元件指定寬與高，在設定的時候使用這些單位，例如「`android:layout_width= "120sp"`」：

- `px`：螢幕畫素。
- `dp`：每英吋畫面，`160dp`為一英吋。
- `sp`：和`dp`一樣，不過會根據裝置設定的字型大小自動調整。
- `in`：英吋。
- `mm`：公厘。

決定畫面元件的大小以後，你可能需要設定與其它元件的間隔距離，這個設定在畫面有比較多元件的時候，可以讓所有元件成一團，畫面看起來會好一些。這些是用來執行與其它元件的間隔距離設定，使用上面說明的單位設定需要的間隔：

- `android:marginTop`：設定上方的間隔。
- `android:marginBottom`：設定下方的間隔。
- `android:marginLeft`：設定左側的間隔。
- `android:marginRight`：設定右側的間隔。
- `android:margin`：設定上、下、左、右為同樣的間隔。

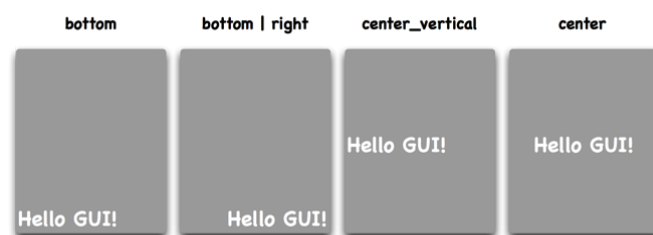
你還可以控制畫面元件內容的間隔距離設定，讓一個元件的內容與它使用的空間有一點間隔，元件本身看起來就不會那麼擁擠。這些是用來執行元件內容間隔距離的設定：

- `android:paddingTop`：設定內容上方的間隔。
- `android:paddingBottom`：設定內容下方的間隔。
- `android:paddingLeft`：設定內容左側的間隔。
- `android:paddingRight`：設定內容右側的間隔。
- `android:padding`：設定內容上、下、左、右為同樣的間隔。

畫面元件還有一個可以控制內容位置的設定，設定名稱是「`android:gravity`」，可以讓你比較容易把內容設定為需要的位置。它的設定值：

- `top`、`bottom`、`left`、`right`：設定畫面元件的內容對齊上、下、左、右。
- `centervertical`、`centerhorizontal`：設定畫面元件的內容對齊垂直或水平的中央。
- `center`：設定畫面元件的內容對齊垂直與水平的中央。
- `fillvertical`、`fillhorizontal`：設定畫面元件的內容佔滿垂直或水平空間。
- `fill`：設定畫面元件的內容佔滿垂直與水平空間。

這些設定值也可以使用組合的方式，例如希望把內容對齊下方的右側，就可以設定為「`bottom|right`」，多個設定值之間使用隔開。這是一個設定效果的範例，它使用一個佔滿整個畫面的`TextView`元件，如果沒有設定的話，預設的內容位置是在左上角，用一些不同的設定控制內容的位置：



如果是可以顯示文字內容的畫面元件，例如文字（`TextView`）、按鈕（`Button`）或輸入（`EditText`）元件，可以為它們加入文與樣式設定：

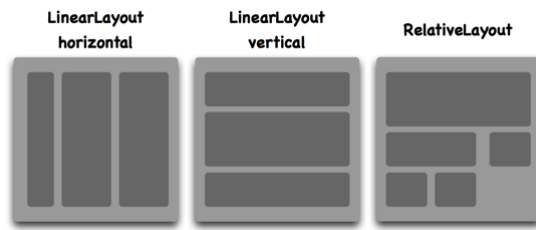
- `android:text`：設定畫面元件的文字內容。
- `android:textSize`：設定文字的大小。
- `android:textAppearance`：使用系統的預設值設定文字的大小，設定的格式為「`?android:attr/設定值`」，有`textAppearanceLarge`（大）、`textAppearanceMedium`（中）和`textAppearanceSmall`（小）三種設定值。
- `android:textColor`：設定文字的顏色。
- `android:background`：設定背景顏色。

## 6-3 使用畫面配置元件

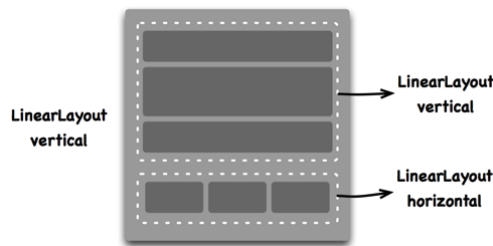
就算是一個簡單的Android應用程式，大部份都會在一個畫面中使用多個畫面元件，組成應用程式需要的畫面，一個畫面只畫面元件的情況應該是不多的。所以你在規劃與設計應用程式的畫面時，除了知道要使用哪一些畫面元件，也要規劃好畫面配置方式，就是決定所有需要的元件如何在畫面上排列。

因為各種Android實體裝置的螢幕尺寸與解析度並不是一樣的，所以畫面元件的排列、位置與大小也應該不是固定的，應用程式同裝置運作的時候，畫面看起來才會一樣。所以Android建議你應該使用「`Layout`」來設計畫面元件的排列方式，`Layout`是`ViewGroup`的子類別，`ViewGroup`是一種容器元件，可以把其它元件放在這些元件裡面，組合成需要的畫面。

在規劃與設計應用程式畫面的時候，應該就可以決定它們該使用哪一種`Layout`。Android在「`android.widget`」套件中提供的「`LinearLayout`」和「`RelativeLayout`」是基本的`Layout`元件。`LinearLayout`可以設定為依照水平（`horizontal`）或垂直（`vertical`）排列。`RelativeLayout`使用畫面元件相對的位置來排列，適合用在比較不規則的畫面元件配置。畫面元件放在這兩種`Layout`中的像這樣：



如果應用程式的畫面比較複雜一些，這樣的Layout應該就不夠用了，所以使用這些Layout的時候，可以使用巢狀的方式組合成複雜的排列。例如這個畫面的排列方式，在最外層使用垂直排列的LinearLayout，裡面又包含上下兩個LinearLayout，它們也依照自己的需求，設定為垂直和水平排列：



ViewGroup還提供一種很常使用的「TableLayout」，它可以把畫面切割為表格，你可以依照畫面的需求設定為像是2X4的區塊，一個區塊都可以放一個畫面元件，如果需要的畫面不是一個固定的表格，也可以調整它們的區塊。使用TableLayout排列的畫面這樣：



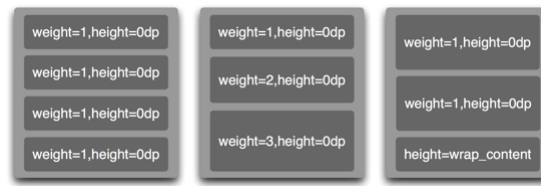
### 6-3-1 LinearLayout

LinearLayout可以提供比較簡單的畫面配置，你可以設定它依照水平（由左到右）或垂直（由上往下）排列畫面元件。如果需這種排列方式，在畫面配置檔中加入LinearLayout標籤，使用「android:orientation」設定排列的方式，「horizontal」為水平「vertical」為垂直：

```
<?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或horizontal" >
    // 放在這裡的畫面元件會依照指定的方式排列
</LinearLayout>
```

放在LinearLayout裡面的元件，還可以設定它們佔用的空間比例，這樣就可以讓這種看起來很單純的排列方式，變得比較靈活。要設定元件佔用的空間比例，使用「android:layout\_weight="比例"」，依照LinearLayout設定為水平或垂直排列，設定元件的

後，它的寬或高就由比例來決定，所以應該把元件的寬或高設定為「0dp」。如果其中有一個元件沒有設定比例，它會依照自與高的設定決定大小，其它空間再由比例決定各自的大小。搭配使用這些設定，就可以很靈活的設定各種畫面的配置：



### 6-3-2 RelativeLayout

應用程式需要的畫面，可能不會都是很規則的排列，如果畫面元件的位置與排列比較複雜與不規則的時候，就比較適合使用 **RelativeLayout** 這種排列方式。它可以讓你決定元件在畫面上位置，還有設定元件與元件之間的相關位置與對齊方式。這種配置使用「**RelativeLayout**」標籤：

```
<?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" >
    // 使用關聯的方式排列畫面元件
</RelativeLayout>
```

放在 **RelativeLayout** 標籤中的畫面元件，可以依照畫面的需求，決定自己在畫面中的位置，也可以設定自己與其它元件的相對位置。這些是在 **RelativeLayout** 標籤中的畫面元件可以使用的設定：

- **android:layout\_位置="@id/元件名稱"**：決定自己在指定元件的相對位置，格式中的「位置」可以使用 **above**、**below**、**toLeftOf** 或 **toRightOf**，依照順序表示把自己放在指定元件的上方、下方、左邊或右邊。
- **android:layout\_align對齊="@id/元件名稱"**：決定自己和指定元件的對齊方式，格式中的「對齊」可以使用 **Top**、**Bottom**、**Left** 或 **right**，依照順序表示把自己的上、下、左或右對齊指定的元件。
- **android:layout\_alignParent對齊="true|false"**：決定自己與容器的對齊方式，格式中的「對齊」可以使用 **Top**、**Bottom**、**Left** 或 **right**，設定為 **true** 的時候，依照順序表示把自己對齊容器的上、下、左或右。
- **android:layout\_center對齊="true|false"**：決定自己與容器對齊中央的方式，格式中的「對齊」可以使用 **Horizontal**、**Vertical** 或 **InParent**，設定為 **true** 的時候，依序表示將自己對齊容器的水平中央、垂直中央或容器中央。

搭配使用這些 **RelativeLayout** 提供的設定，就算是比較複雜或不規則的畫面，也可以很容易完成設計的工作：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout ... >
    <TextView
        android:id="@+id/account"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="ACCOUNT: "
```

```

        android:textSize="24sp" />

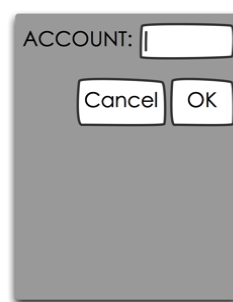
<!-- 把自己放在account元件的右邊 -->
<EditText
    android:id="@+id/account_value"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_toRightOf="@id/account"
    android:paddingLeft="6sp" />

<!-- 把自己放在account_value元件的下方，而且對齊容器右邊 -->
<Button
    android:id="@+id/ok"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_below="@id/account_value"
    android:text="OK" />

<!-- 把自己放在ok元件的左邊，而且上方也對齊它 -->
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignTop="@id/ok"
    android:layout_toLeftOf="@id/ok"
    android:text="Cancel" />
</RelativeLayout>

```

這個畫面配置檔呈現的畫面會像這樣：



### 6-3-3 TableLayout

ViewGroup還提供一種很常使用的「TableLayout」畫面排列方式，它可以把畫面切割為表格，你可以依照畫面的需求，把畫面為像是2X4的區塊，每一個區塊都可以放一個畫面元件，如果需要的畫面不是一個固定的表格，也可以調整它們的區塊。這種置使用「TableLayout」標籤，搭配「TableRow」標籤建立需要的表格。TableLayout會控制畫面元件的寬與高，所以在這個模的畫面元件都不需要設定寬與高，就算加入設定也不會有效果：

```

<?xml version="1.0" encoding="utf-8"?>
<TableLayout

```

```

xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent" >
<!-- 第一列 -->
<!-- TableRow不需要設定寬與高 -->
<TableRow>
    <!-- 所有的元件都不需要設定寬與高 -->
    ...
</TableRow>
<!-- 第二列 -->
<TableRow>
    ...
</TableRow>
</TableLayout>

```

TableLayout會管理與控制畫面元件的大小，預設的情況，畫面元件的寬度都是「wrap\_content」的效果。你可以在TableLayout中加入這些設定，執行元件的寬度與是否隱藏的設定：

- **android:stretchColumns**：放大指定的欄位寬度。第一個欄位是0，可以設定多個欄位，例如「1,3,5」；設定為「\*」表示所有欄位。
- **android:shrinkColumns**：寬度不夠顯示所有內容的時候，指定的欄位會自動縮小。第一個欄位是0，可以設定多個欄位，例如「1,3,5」；設定為「\*」表示所有欄位。
- **android:collapseColumns**：隱藏指定的欄位。第一個欄位是0，可以設定多個欄位，例如「1,3,5」；設定為「\*」表示所有欄位。

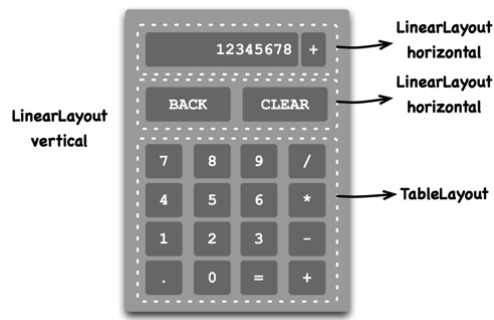
在TableLayout中加入android:stretchColumns的設定，可以讓指定的欄位佔用較大的空間。



### 6-3-4 結合多種畫面排列元件

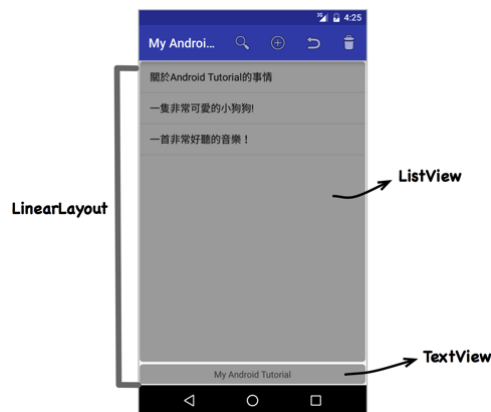
上面說的LinearLayout、RelativeLayout和TableLayout，單獨使用的時候，在設計一些比較複雜的畫面時，就比較不符合需求。只要是Layout元件都可以搭配使用，所以在規畫與設計應用程式的畫面時，如果想要設計一個比較複雜的畫面，應該要把畫面有的元件切割成適合的區塊，為每一個區塊挑選一個合適的排列方式，最後再把它們結合起來。例如像這個計算機應用程式的畫面，就會依照它們的需求使用LinearLayout和TableLayout組合成這樣的畫面：





## 6-4 建立記事本應用程式主畫面

瞭解Android應用程式畫面的設計方式後，現在要回到記事本應用程式，為它設計一個用來顯示所有資料的主畫面：



這個畫面的需求並不會太複雜，使用LinearLayout畫面配置元件就可以了，安排畫面元件的時候，最好幫它們設定邊界，元件擠在一起。開啟「res/values/dimens.xml」，加入需要的尺寸資源：

```
<resources>
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>

    <dimen name="default_padding">6dp</dimen>
    <dimen name="title_txt_size">24sp</dimen>
    <!-- 加入邊界尺寸資源設定 -->
    <dimen name="default_margin">2dp</dimen>
</resources>
```

接下來開啟「res/layout/activity\_main.xml」檔案，把它改為下面的內容：

```
<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"
    tools:context=".MainActivity">
```

```

<ListView
    android:id="@+id/item_list"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:layout_margin="@dimen/default_margin"
    android:dividerHeight="1sp"
    android:background="@drawable/rectangle_drawable"
    android:divider="@color/divider_color" />

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:layout_margin="@dimen/default_margin"
    android:padding="@dimen/default_padding"
    android:background="@drawable/rectangle_drawable"
    android:text="@string/app_name"/>

</LinearLayout>

```

執行這個應用程式，看看它在模擬裝置中顯示的畫面。這個畫面中的**ListView**是用來顯示資料列表的元件，目前還沒有為它設資料，所以看起來是空白的。

為**ListView**元件指定資料的工作必須在**Activity**元件的程式碼執行，所以在畫面配置檔為**ListView**元件使用「**android:id**」為它設個名稱。開啟專案的「**MainActivity.kt**」，修改為下面的內容：

```

package net.macdidi.atk

import android.os.Bundle
import android.support.v7.app.AppCompatActivity
import android.view.Menu
import android.widget.ArrayAdapter
import android.widget.ListView

class MainActivity : AppCompatActivity() {

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

        // 為ListView元件設定三筆資料
        val data = arrayOf("關於Android Tutorial的事情",
            "一隻非常可愛的小狗狗!", "一首非常好聽的音樂!")
        val layoutId = android.R.layout.simple_list_item_1
        val adapter = ArrayAdapter(this, layoutId, data)
        val item_list: ListView = findViewById(R.id.item_list)
        item_list.setAdapter(adapter)
    }
}

```

```

    }

    override fun onCreateOptionsMenu(menu: Menu): Boolean {
        menuInflater.inflate(R.menu.menu_main, menu)
        return true
    }
}

```

完成這個階段的工作了，執行這個應用程式，檢查ListView元件是否顯示三筆範例資料：



相關的檔案都可以在GitHub瀏覽與下載：

# GitHub


<https://github.com/macdidi5/Android-Tutorial-Kotlin>

後續 >> Android Tutorial using Kotlin 第二堂（3）應用程式與使用者的互動

Does Clearly work fine?



Shortcuts: **SHIFT+CTRL+C** to Toggle, **ESC** to Close.

 Give us feedback

Build upon ♥ with Clearly

