

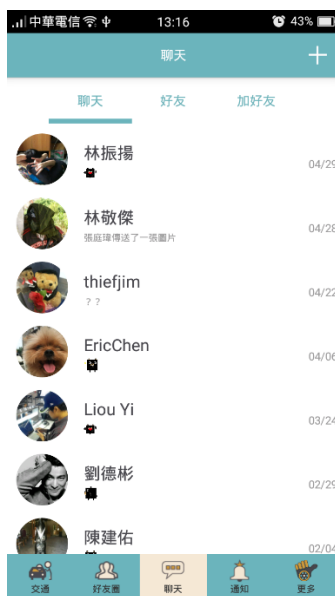
Lab6 清單元件

一、本節目的：

- 理解什麼是 Adapter
- 理解 Adapter 與 ListView 的關係
- 學習如何使用清單元件，Spinner、ListView、Gallery 及 GridView

二、觀念說明：

應用程式最主要的目的便是要傳達某個訊息給使用者，而當這訊息量非常多的時候，單純的畫面就無法容納所有的資訊，尤其是在畫面非常小的手機螢幕上。這時我們往往會使用清單或下拉式選單等方式，讓使用者透過滑動查看更多資訊。這類的列表清單元件對於 Android 應用程式而言就有著舉足輕重的重要性。如以下 APP 畫面，很常看到清單的呈現方式：

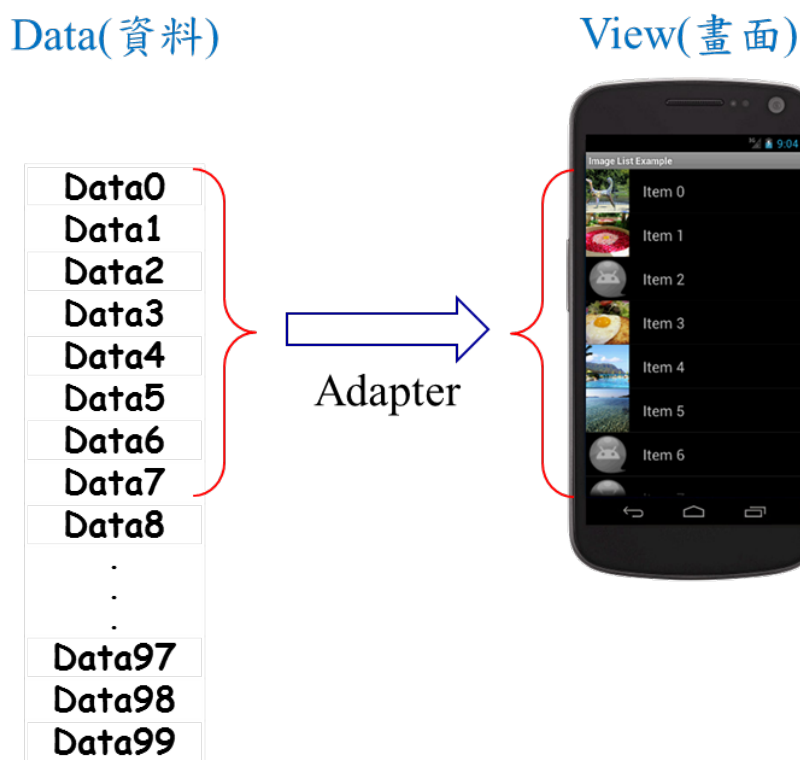


1 Adapter 介紹

清單元件在顯示資料內容上有很特別的設計結構。要顯示的資料一般都是來自於外部的程式，例如透過網路或是其他方式取得資料，我們將這些外部資料稱之為資料來源。

在 Android 中，資料來源的處理與顯示畫面的清單元件是分開來的操作的，只有在需要顯示的時候，才會把資料來源轉成顯示的清單畫面，而負責做這個轉換動作的介面就是 Adapter，而清單元件就像是容器，決定容器內要放入的什麼內容的人就是 Adapter。

用簡單的比喻，清單元件(View)就像是間飯店，資料來源(Data)就像是客人，而 Adapter 類似於接待人員，客人要進到房前需要先詢問接待人員，房間位置，在由接待人員來安排客人進駐到對應的飯店房間。



2 Adapter 繼承類別與使用

在使用清單元件時，每一筆項目(Item)的畫面會需要有對應的 layout(Xml) 來作呈現，而 Android SDK 本身也有提供的現成的圖檔、layout 的資源，我們可以直接使用實現簡單的顯示效果。以下我們用 ListView 元件說明如何使用 ArrayAdapter：

Step1：建立資料來源

```
final String[] data = { "Data1", "Data2", "Data3", "Data4" };
```

Step2：建立 adapter 物件，並放入資料來源與要顯示的項目畫面

```
ArrayAdapter<String> adapter = new ArrayAdapter<>(  
    this, android.R.layout.simple_list_item_1, data);
```

Step3：建立清單元件，並連結 adapter

```
ListView listView = (ListView) findViewById(R.id.listView);  
listView.setAdapter(adapter);
```

Step4：建立項目 (Item) 點擊事件

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
    @Override 回傳第幾個項目被按下  
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
        Toast.makeText(MainActivity.this, data[position], Toast.LENGTH_SHORT).show();  
    }  
});
```

- 1) 我們要先假設擁有一個資料來源，我們要利用 ArrayAdapter，ArrayAdapter 繼承自 BaseAdapter，資料來源需要是陣列格式。
- 2) 我們要產生出 ArrayAdapter 的實體，第一個參數要傳入呼叫對象(即 this，本身物件)，第二個參數傳入一個 layout，這邊我們直接使用 Android SDK 提供的 android.R.layout.simple_list_item_1 現成的 layout(即最簡單的項目風格)，第三個參數傳進資料(即 data)。
- 3) 我們要將 ArrayAdapter 指派給畫面元件，這邊我們實作出一個 ListView，使用 setAdapter() 將 Adapter 做連結。
- 4) 我們要為 ListView 設定點擊事件。由於我們不是要對 ListView 做點擊，而是要對 ListView 裡的項目(item)作點擊，因此這邊我們使用 OnItemClickListener()，OnItemClickListener 內的 onItemClick 方法的第三個參數 position 會回傳按下的項目編號，可以根據該編號從資料來源中取出對應的資料。

3 Adapter 客製化

某些情況下，我們需要更複雜的畫面，可能會包含圖片、文字等，可能 Android SDK 所提供的 layout 範例就無法滿足我們的使用需求，這時我們就需要實作客製化 Adapter。

在實作 Adapter 客製化 上我們有三個準備工作：

- 1) 設計客製化 Data：如果我們要顯示的資料項目需要兩種以上的資料內容(如要顯示標題、內容、圖片等多筆資料)，我們就需要設計對應的一個類別去定義這些資料。

```
class Data{  
    String name,message;  
}
```

與要顯示內容對應的類別宣告

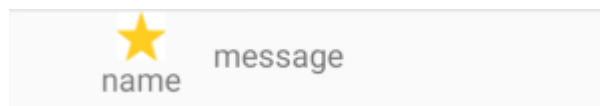
```
data = new Data[10];
```

 用陣列方式宣告自行設計的類別

```
for(int i=0;i<10;i++){  
    data[i]=new Data();  
    data[i].name="name"+i;  
    data[i].message="message"+i;  
}
```

用迴圈去產生資料來源，並放入類別陣列之中。

- 2) 設計客製化 layout：我們需要設計要顯示的項目 layout。用之前章節所教的 layout 設計方法去設計客製化的 Xml。



- 3) 建立客製化的 Adapter：由於我們的資料內容與格式是自行設計的，因此我們也需要自行設計對應的 Adapter。由於我們希望延伸 BaseAdapter 的功能，我們透過 Java 的繼承功能建立一個繼承於 BaseAdapter 的 myAdapter，並藉由覆寫的方式修改其中幾個方法來建立自己的客製化 Adapter。如下所示：

```

public class myAdapter extends BaseAdapter {

    @Override
    public int getCount() { return 0; } 取得資料來源陣列的筆數。

    @Override
    public Data getItem(int position) {return null;} 取得指定項目內的資料。

    @Override
    public long getItemId(int position) { return 0; } 取得指定項目的資料 id。

    @Override
    public View getView(int position, View rowView, ViewGroup parent) {return null;}
} 顯示項目(Item)的資料對應的

```

創立繼承 BaseAdapter 的物件後，我們需要覆寫 BaseAdapter 的幾種方法：

1) public int getCount()

getCount 須回傳要顯示的資料筆數，可直接放入陣列的長度，程式碼覆寫如下：

```

@Override
public int getCount() { return data.length; }

```

2) public Data getItem(int position)

getItem()可得到 position 對應的資料，程式碼覆寫如下：

```

@Override
public Data getItem(int position) {return data[position];}

```

3) public long getItemId(int position)

getItemId ()可得到 position 對應的 id 值，這 id 值應該要為唯一性的編號，一般下非必要的參數，故此處不作修改。

4) public View getView(int position, View rowView, ViewGroup parent)

Adapter 的 getView ()方法，主要是將資料來源(Data)顯示在畫面(View)上，是設計的客製化 Adapter 的核心。如下圖所示，取得項目編號(position)之後，把資料來源(Data)依照房間編號(position)顯示在畫面中。



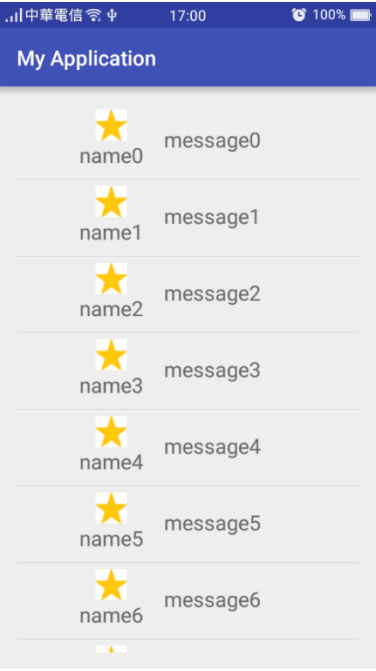
```
@Override                      需要顯示的項目編號
public View getView(int position, View rowView, ViewGroup parent) {
    取得畫面元件
    rowView = getLayoutInflater().inflate(R.layout.m_list, parent, false);

    TextView name = (TextView) rowView.findViewById(R.id.name);
    TextView message = (TextView) rowView.findViewById(R.id.message);
    name.setText(data[position].name);
    message.setText(data[position].message);
    根據項目編號把對應的資料放到畫面元件之中
    return rowView;
}
```

設計完 myAdapter 後，我們將 setAdapter 換成 myAdapter，實作後的結果如下：

```
myAdapter adapter = new myAdapter();

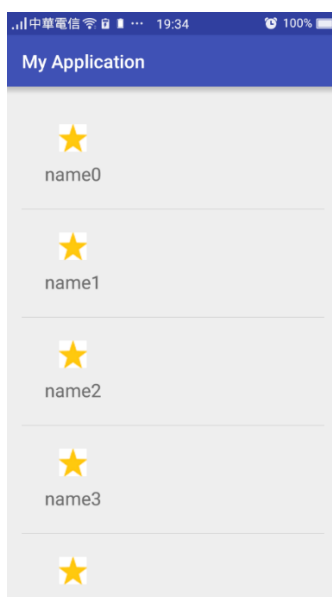
ListView listView = (ListView) findViewById(R.id.listView);
listView.setAdapter(adapter);
```



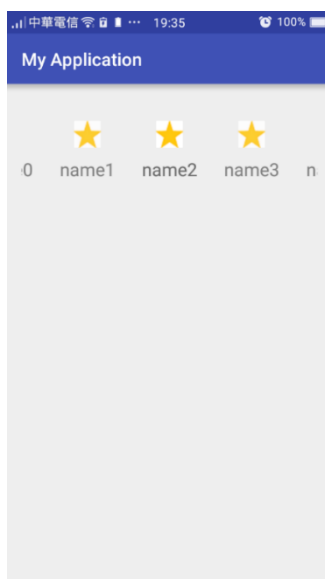
4 清單元件

Android 有提供幾種清單元件以供顯示，由於資料內容都是由 Adapter 決定，清單元件扮演著容器的角色，因此只需要替換連結的清單元件就可以實現不同的資料模式，以下分別作介紹：

- **ListView(縱向清單)：**ListView 是最基本的清單元件，他可以將資料垂直排放，由於大部分的行動裝置都是長比寬高，因此 ListView 能很清楚的顯示資訊。

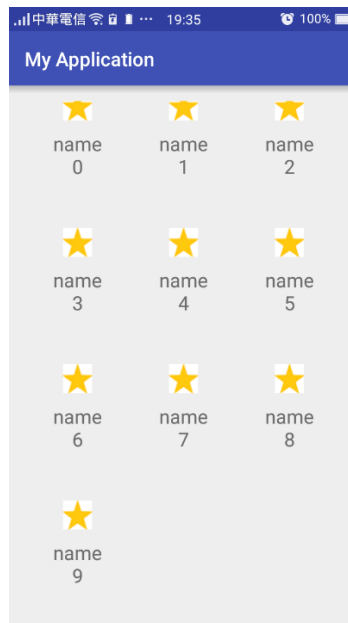


- **Gallery (橫向清單/畫廊)：**Gallery 是可以橫向翻動的清單，在應用程式中常會使用他來展示照片顯示，因此被稱為畫廊。

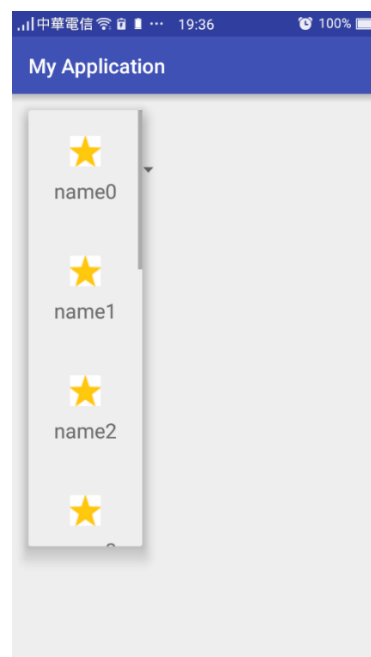
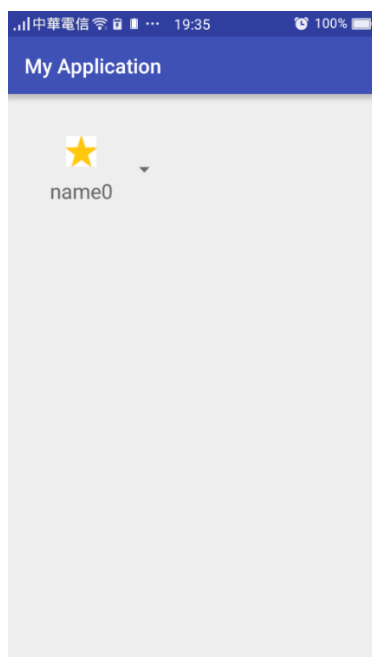


- **GridView (格狀清單)**: GridView 能將內容縮成方形，透過窗戶般的格狀擺放方式，並依照由左至右、由上到下的排列。GridView 比起前述兩個元件，他可以使用 `setNumColumns()` 決定橫向要顯示幾列，如果沒有設定則只會顯示一列。

```
gridView.setNumColumns(3);
```

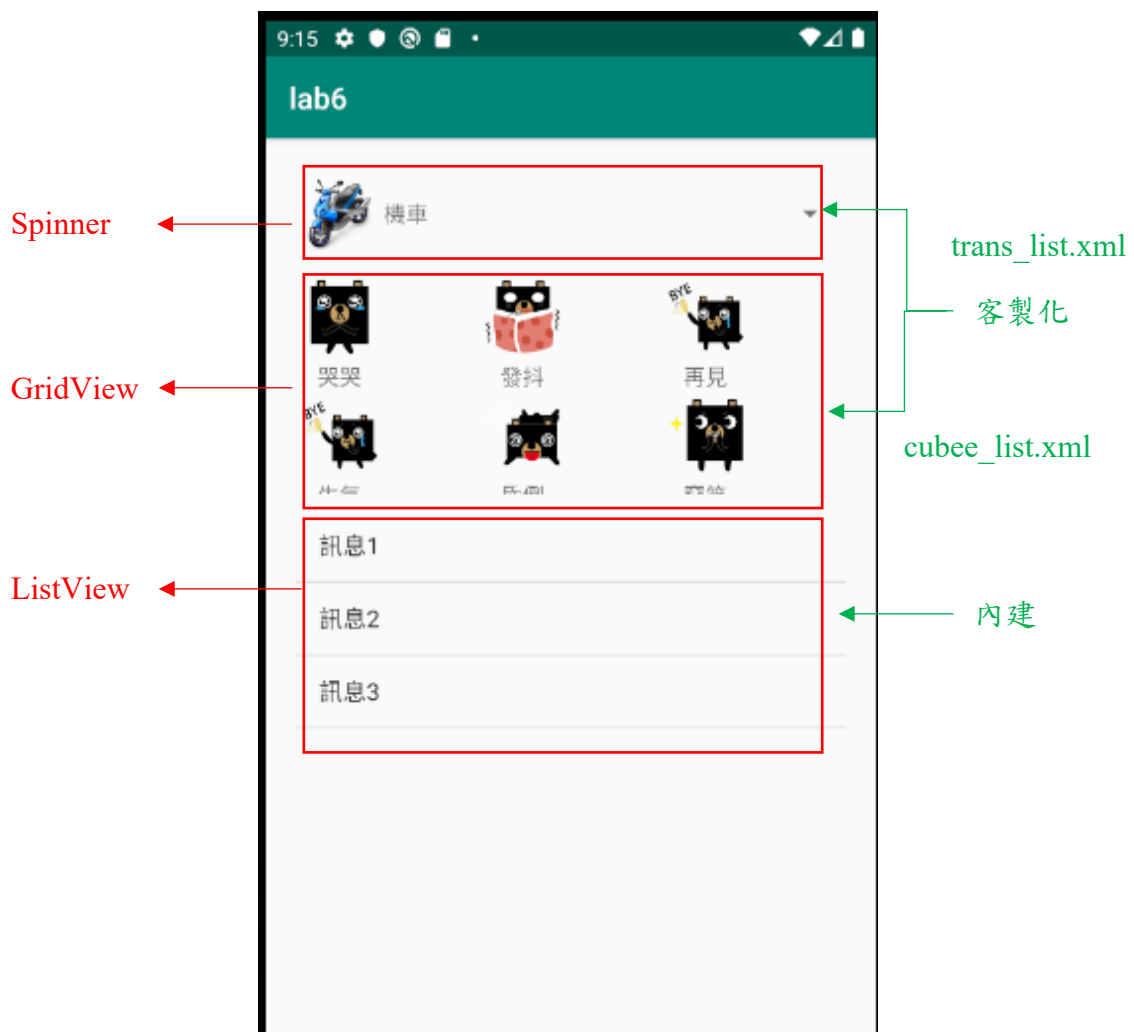
 顯示三列

- **Spinner (下拉式選單)**: Spinner 是下拉式選單，一般下只會占用一個 TextView 元件大小，但被點擊時可以展開清單讓使用者選擇。



三、設計重點:

- 利用 Adapter 靈活的配置 Spinner、ListView 及 GridView
- 設計一含有 Spinner、ListView 及 GridView 的布局
- ListView 使用範例 Xml 顯示清單
- Spinner 與 GridView 分別使用兩種不同的客製化畫面來實作客製化 Adapter 顯示清單

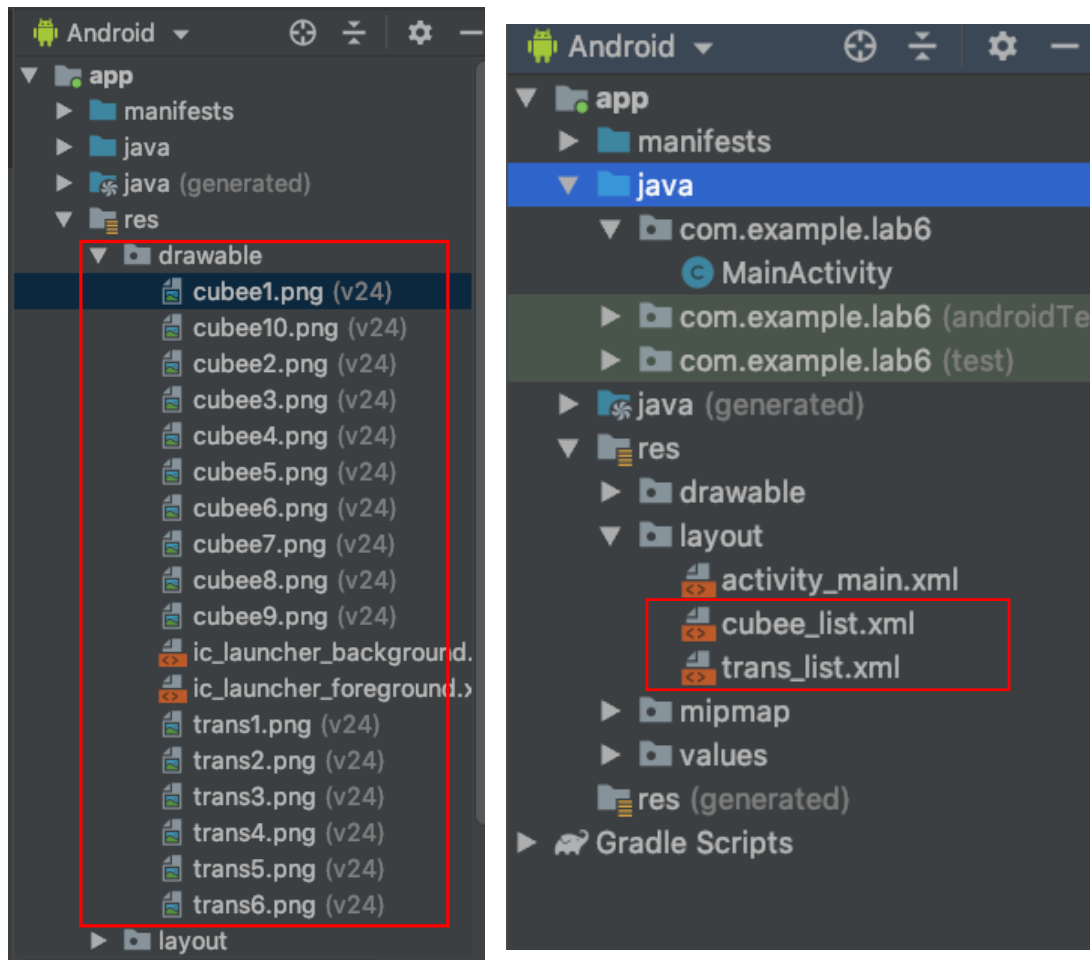


四、設計步驟:

Step1

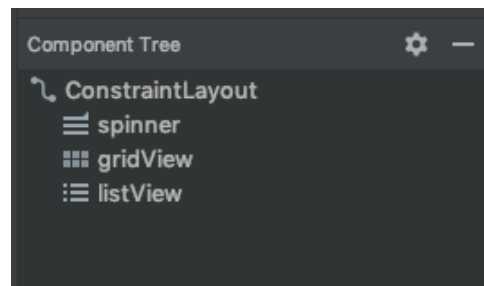
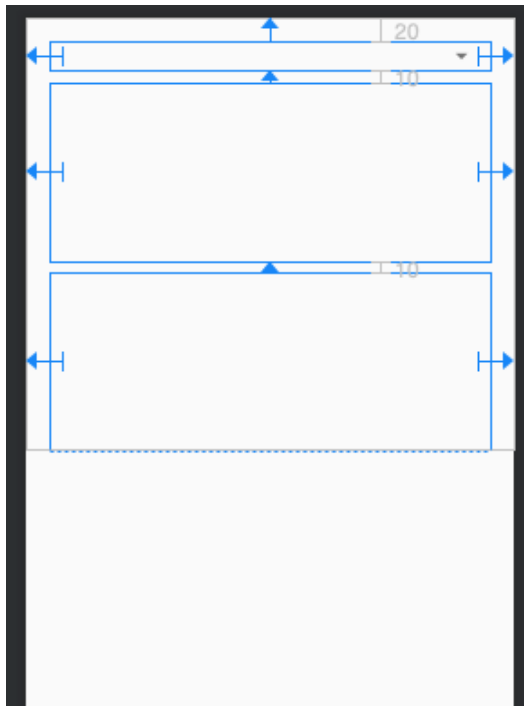
建立專案，並將附件的圖片放於 drawable 底下

以及建立 xml 檔



Step2

繪製 activity_main.xml



對應的 xml 如下：

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    tools:context=".MainActivity">

    <Spinner
        android:id="@+id/spinner"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="20dp"
        android:layout_marginTop="20dp"
        android:layout_marginEnd="20dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

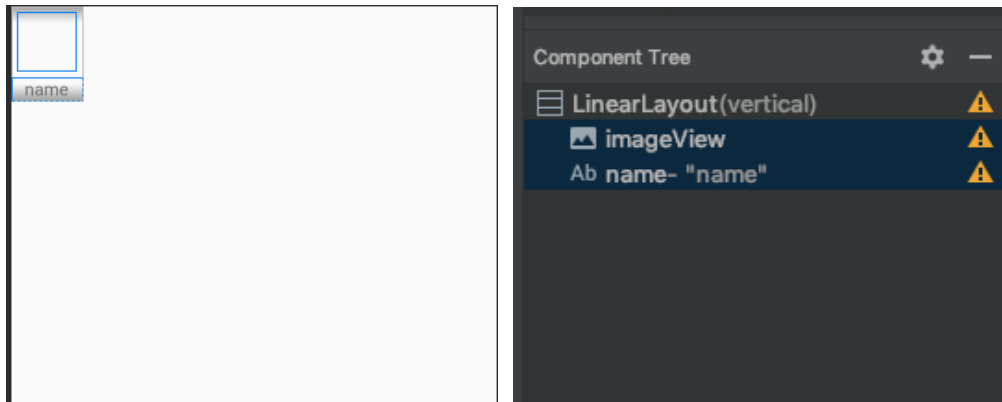
    <GridView
        android:id="@+id/gridView"
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:layout_marginStart="20dp"
        android:layout_marginTop="10dp"
        android:layout_marginEnd="20dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/spinner">

    </GridView>

    <ListView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:layout_marginStart="20dp"
        android:layout_marginTop="10dp"
        android:layout_marginEnd="20dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/gridView" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Step3

繪製 cubee_list.xml，顯示客製化的畫面



對應的 xml 如下：

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

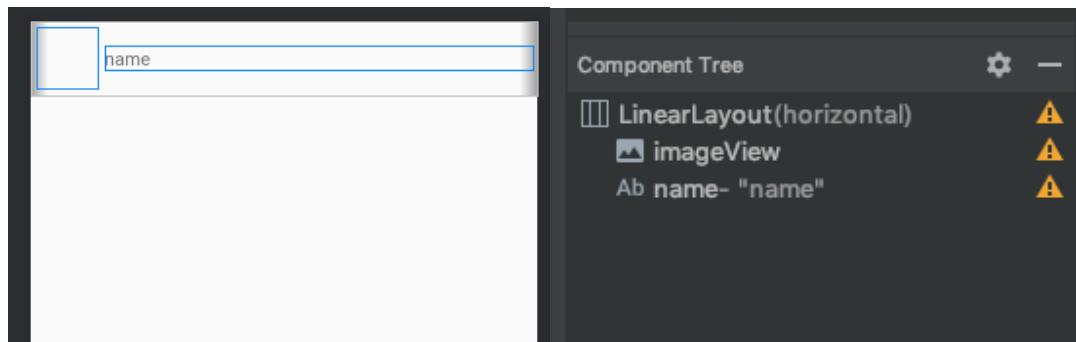
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_margin="5dp"
        />

    <TextView
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="name"
        android:textSize="15sp"
        android:gravity="center"
        />

</LinearLayout>
```

Step4

繪製 trans_list.xml，顯示客製化的畫面



對應的 xml 如下：

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

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_margin="5dp"
    />

    <TextView
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="15sp"
        android:text="name"
        android:gravity="start"
        android:layout_gravity="center"
        android:layout_marginRight="5dp"
    />

</LinearLayout>
```

Step5

編寫 MainActivity，建立一個客製化的類別 Data，包含一張圖片與文字，用於保存之後要顯示於客製化 Adapter 的資料。

```
public class MainActivity extends AppCompatActivity {  
  
    class Data{  
        int photo;    圖片 id  
        String name;  名稱  
    }  
}
```

Step6

建立 myAdapter 來顯示 Spinner 及 GridView 的客製化畫面。由於 Spinner 與 GridView 有各自要顯示的資料與畫面，因此我們需要先把他們的資料在建構 myAdapter 就保存在 myAdapter 中，避免出現取錯資料的問題。

```
public class MyAdapter extends BaseAdapter{  
    private MainActivity.Data[] data;  
    private int view;  
  
    public MyAdapter(MainActivity.Data[] data, int view){  
        this.data = data;  
        this.view = view;  
    }  
  
    @Override  
    public int getCount() {  
        return data.length;  
    }  
  
    @Override  
    public Object getItem(int position) {  
        return data[position];  
    }  
  
    @Override  
    public long getItemId(int position) {  
        return 0;  
    }  
  
    @Override  
    public View getView(int position, View convertView, ViewGroup parent) {  
        convertView = getLayoutInflater().inflate(view, parent, attachToRoot: false);  
        TextView name = convertView.findViewById(R.id.name);  
        name.setText(data[position].name);  
        ImageView imageView = convertView.findViewById(R.id.imageView);  
        imageView.setImageResource(data[position].photo);  
        return convertView;  
    }  
}
```

繼承 BaseAdapter
保存在 myAdapter 之中的資料來源
保存在 myAdapter 之中的畫面

透過建構子儲存資料來源
與畫面到 myAdapter 之中

回傳資料來源筆數

回傳某筆項目

回傳某筆項目 id

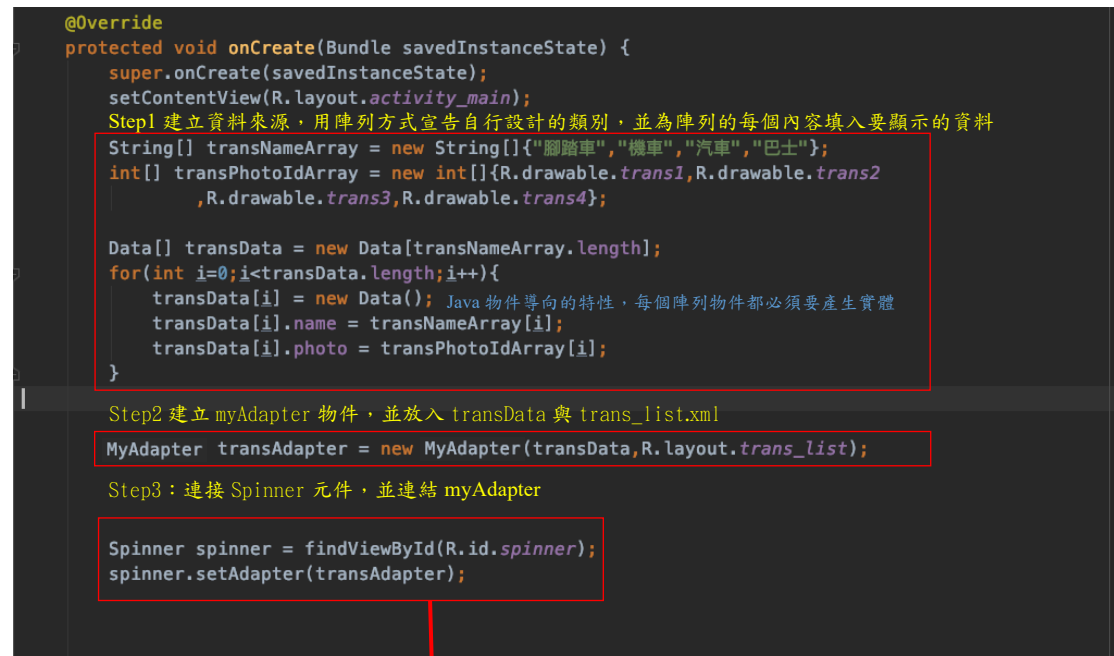
取得畫面元件

需要顯示的項目編號

取得 Xml 畫面
連接 TextView 元件
根據 position 把字串顯示到 TextView
連接 ImageView 元件
根據 position 把圖片顯示到 ImageView

Step7

使用 Spinner，透過客製化 layout 與 myAdapter 顯示 transData(自製的資料)



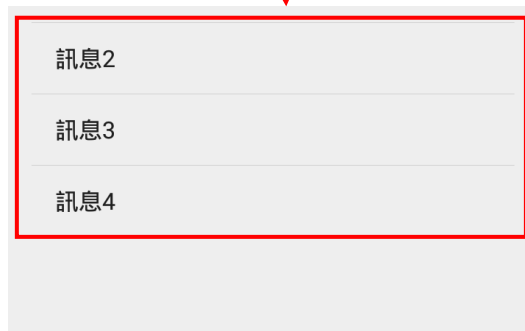
Step7

使用 ListView，透過 Android SDK 提供的現成 layout 與 ArrayAdapter 顯示 messageArray(字串陣列)

```
Step1: 建立資料來源(字串)
String[] messageArray = new String[]{"訊息1", "訊息2", "訊息3", "訊息4", "訊息5", "訊息6"};

Step2: 建立 Adapter 物件，並放入字串與 simple_list_item_1.xml
ArrayAdapter<String> messageAdapter = new ArrayAdapter<>( context: this,
    android.R.layout.simple_list_item_1, messageArray);

Step3: 連接 ListView 元件，並連結 Adapter
ListView listView = findViewById(R.id.listView);
listView.setAdapter(messageAdapter);
```



Step8

使用 GridView，透過客製化 layout 與 myAdapter 顯示 cubeeData (自製的資料)

Step1：建立資料來源，用陣列方式宣告自行設計的類別，將陣列的每個內容填入要顯示的資料

```
String[] cubeeNameArray = new String[]{"哭哭", "發抖", "再見", "生氣", "昏倒", "竊笑",  
    "很棒", "你好", "驚嚇", "大笑"};  
int[] cubeePhotoIdArray = new int[]{R.drawable.cubee1, R.drawable.cubee2  
    , R.drawable.cubee3, R.drawable.cubee3, R.drawable.cubee5, R.drawable.cubee6  
    , R.drawable.cubee7, R.drawable.cubee8, R.drawable.cubee9, R.drawable.cubee10  
    };  
Data[] cubeeData = new Data[cubeeNameArray.length];  
for(int i=0; i<cubeeData.length; i++){  
    cubeeData[i] = new Data();    物件導向的特性，每個陣列物件都必須要產生實體  
    cubeeData[i].name = cubeeNameArray[i];  
    cubeeData[i].photo = cubeePhotoIdArray[i];  
}
```

Step2：建立 myAdapter 物件，並放入 cubeeData 與 cubee_list.xml

```
MyAdapter cubeeAdapter = new MyAdapter(cubeeData, R.layout.cubee_list);
```

Step3：連接 GridView 元件，並連結 myAdapter

```
GridView gridView = findViewById(R.id.gridView);  
gridView.setAdapter(cubeeAdapter);  
gridView.setNumColumns(3);
```

