Lab7
清單元件

本節目的：

- 了解什麼是 Adapter。
- 了解 Adapter 與 ListView 的關係。
- 學習使用清單元件，Spinner、ListView 及 GridView。

7.1 清單列表

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。如下圖 7-1 聊天室列表與通知列表，APP 中很常看到清單的呈現方式。

圖 7-1 聊天室列表（左）與通知列表（右）

7.1.1Adapter 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。如圖 7-2 所示，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。

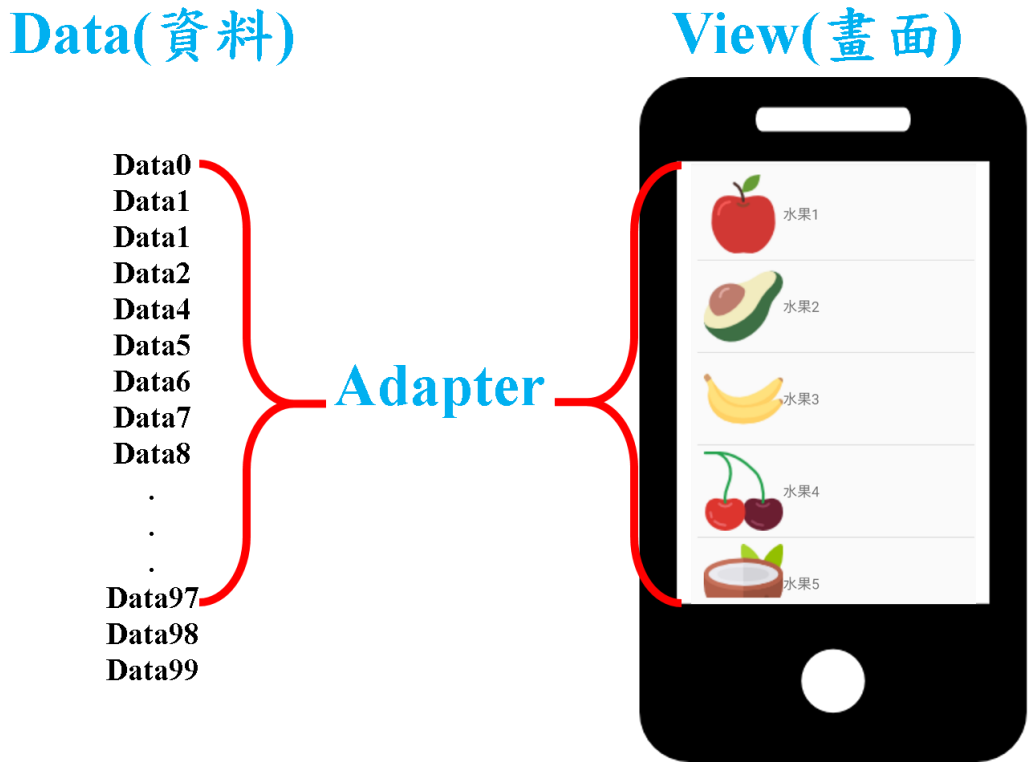

圖 7-2 Data 藉由 Adapter 與 View 溝通

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，再由接待人員來安排客人進駐到對應的飯店房間。

7.1.2Adapter 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
val list_item = arrayListOf("項目 1","項目 2","項目 3","項目 4")
//Step2：建立 adapter 物件，並放入資料來源與要顯示的項目畫面
val adapter = ArrayAdapter<String>(this,
              android.R.layout.simple_list_item_1, list_item)
//Step3：連結 adapter
listView.adapter = adapter
//Step4：建立項目（Item）點擊事件
listView.setOnItemClickListener {
     parent, view, position, id ->
    Toast.makeText(this,"你選的是" +  //回傳第幾個項目被按下
                list_item[position],Toast.LENGTH_SHORT).show()
}
```

Step1 我們要先假設擁有一個資料來源，我們要使用 ArrayAdapter，ArrayAdapter 繼承自 BaseAdapter，資料來源需要是陣列格式。

Step2 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），第三參數傳進資料（即 data）。

Step3 我們要將 ArrayAdapter 指派給畫面元件，這邊我們要將 ListView 的 adapter 與我們創建的 ArrayAdapter 做連結。

Step4 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。

（請勿對 ListView 使用 OnClickListener）

7.1.3Adapter 客製化

某些情況下，我們需要更複雜的畫面，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：

- 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。

```
data class Item(
      val photo: Int, //照片
      val name: String //姓名
)
```

```
//用列表方式宣告自行設計的類別
val item = ArrayList<Item>()
//用迴圈去產生資料來源，並放入類別陣列之中
for(i in 0 until 10)
    item.add(Item(i, "水果${i+1}"))
```

- 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，如圖 7-3 所示。

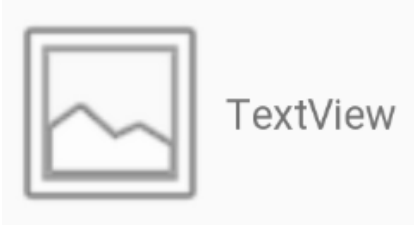

圖 7-3 設計一個帶有圖片與文字的 Layout

- 建立客製化的 Adpater：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，我們透過物件導向的繼承功能建立一個繼承 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：

```
class MyAdapter : BaseAdapter() {
    //取得資料來源陣列的筆數
    override fun getCount() = 0
    //取得指定項目內的資料
    override fun getItem(position: Int) = null
    //取得指定項目的資料 Id
    override fun getItemId(position: Int) = 0L
    //顯示項目（Item）的資料對應的畫面
    override fun getView(position: Int, convertView:
View?, parent: ViewGroup?): View? {
            return convertView
    }
}
```

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 fun getCount() = data.size
```

2) public Data getItem(position: Int)

getItem()可得到 position 對應的資料，程式碼覆寫如下：

```
override fun getItem(position: Int) = data[position]
```

3) public long getItemId(position: Int)

```
override fun getItemId(position: Int) = 0L
```

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(position: Int, convertView: View?, parent: ViewGroup?)

```
override fun getView(position: Int, convertView:
View?, parent: ViewGroup?): View? {
        return convertView
    }
```

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。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，如圖 7-4 所示。

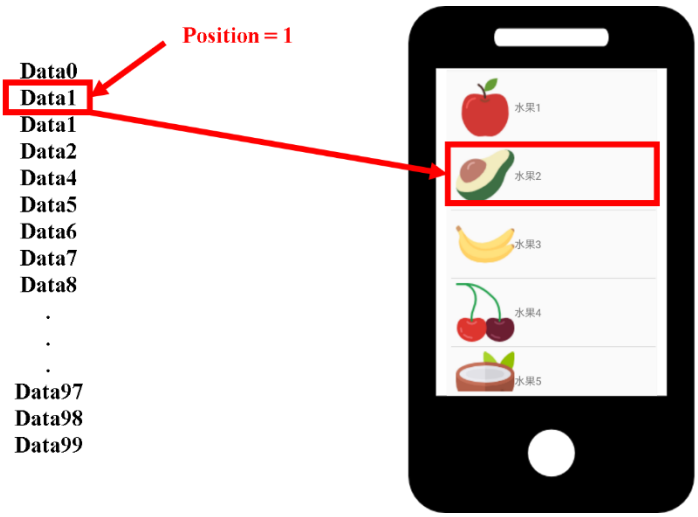

圖 7-4 根據 Position 取得 Data、放到對應的項目（View）

```
override fun getView(position: Int, convertView:
View?, parent: ViewGroup?): View { //需要顯示的項目編號
    val view = View.inflate(parent?.context,
layout, null)
//根據項目編號把對應的資料放到畫面元件之中
view.img_photo.setImageResource(data[position].photo)
    view.tv_name.text = data[position].name

    return view //取得畫面
}
```

設計完 MyAdapter 後，我們將 adapter 連結 MyAdapter，實作後的結果如圖 7-5 所示。

```
listView.adapter = MyAdapter()
```

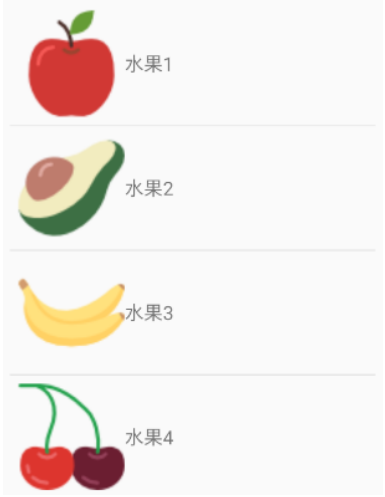

圖 7-5 客製化的 ListView

7.1.4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，如圖 7-6 所示。

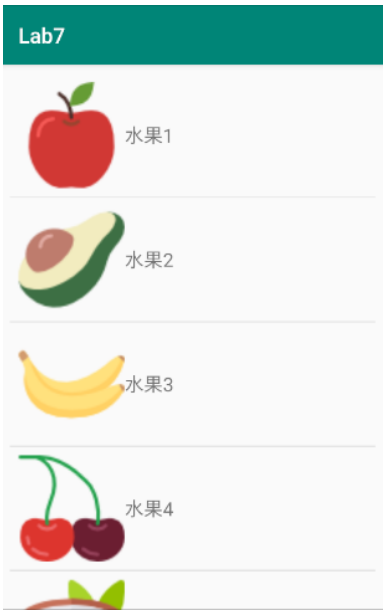

圖 7-6 ListView 範例

- 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，他可以設定 numColumns 決定橫向要顯示幾列，如果沒有設定則只會顯示一列。

```
gridView.numColumns = 3; //顯示三列
```

圖 7-7 GridView 範例

- Spinner（下拉式選單）：Spinner 是下拉式選單，一般狀態下只會占用一個 Item 的大小，但被點擊時可以展開清單讓使用者選擇。

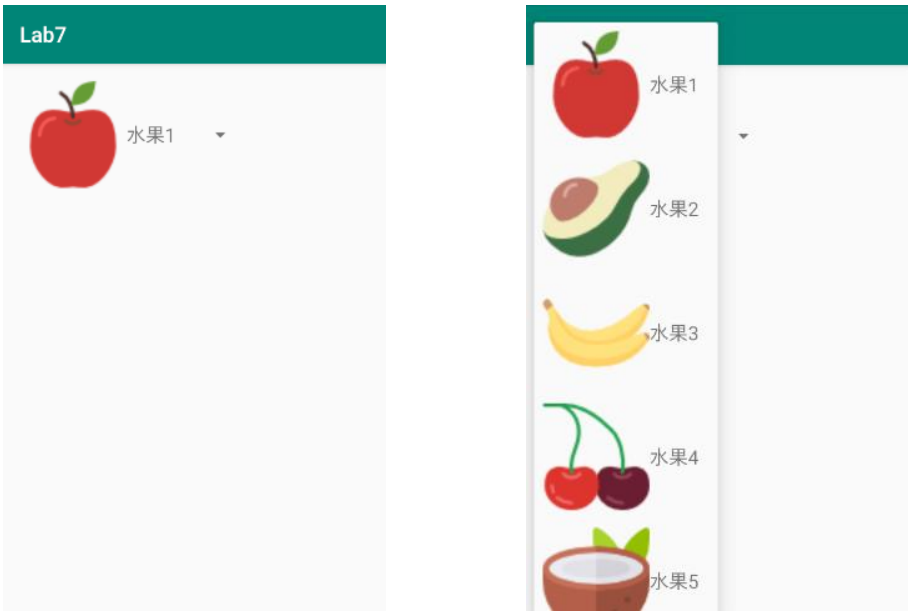

圖 7-8 默認的下拉式選單（左）與展開的選單（右）

7.2 列表實戰

- 利用 Adapter 靈活的配置 Spinner、ListView 及 GridView 如下圖 7-9。
- 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。
- ListView 使用範例 Xml 顯示清單。

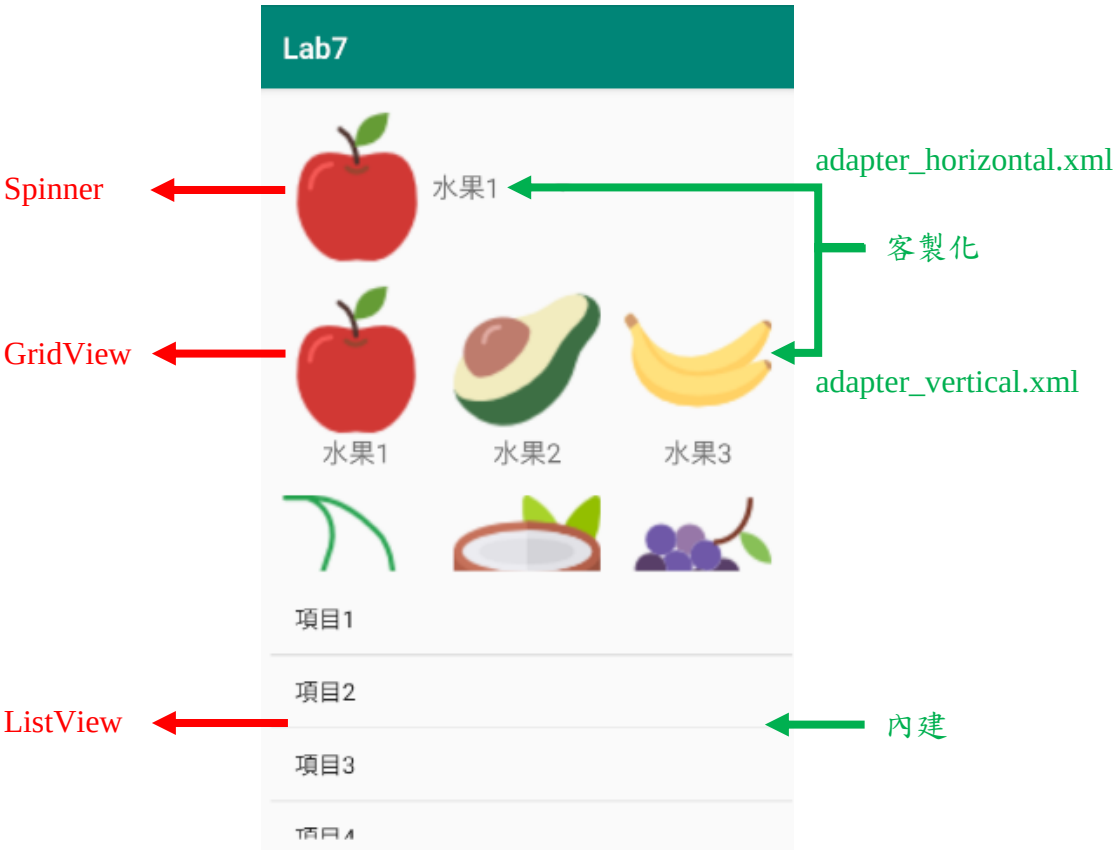

圖 7-9 APP 預覽畫面

7.2.1清單元件畫面設計

Step1 建立專案，並將附件的圖片放於 drawable 底下，如圖 7-10 所示。

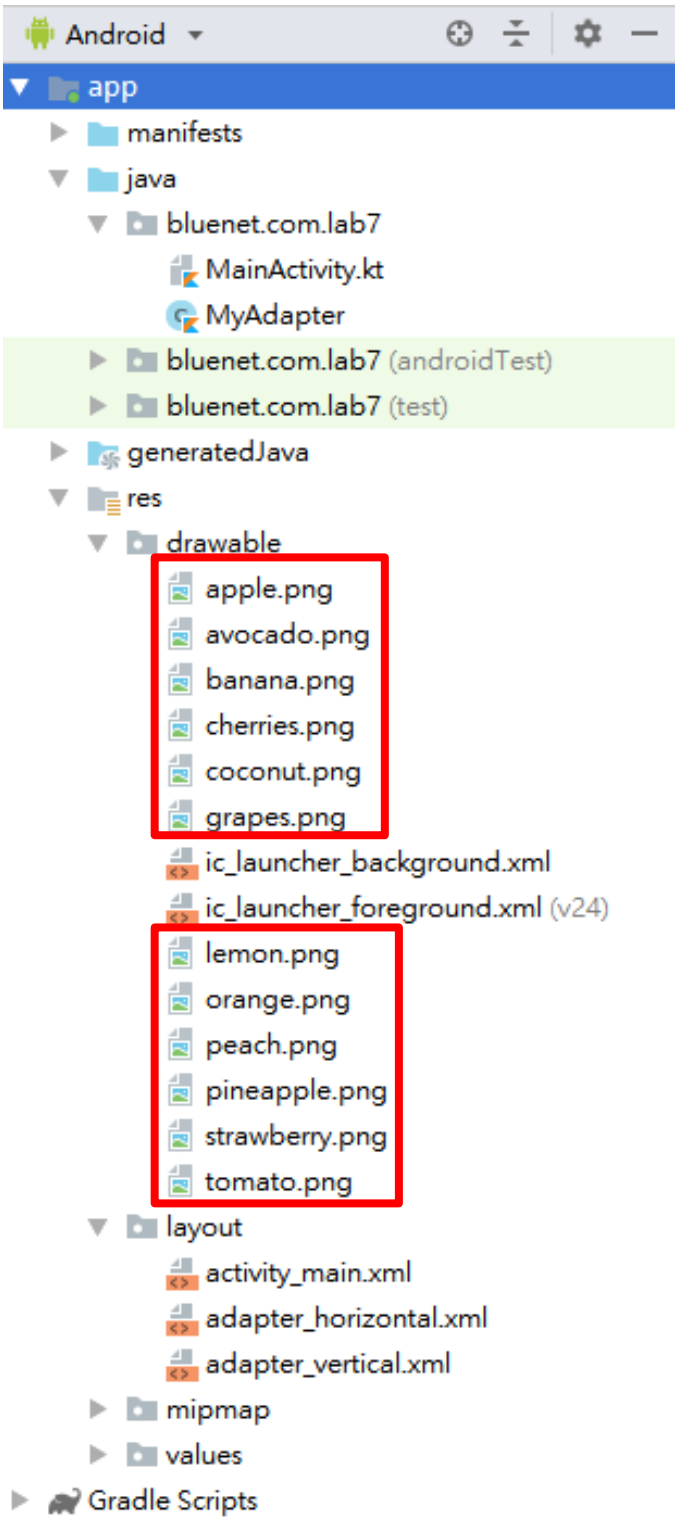

圖 7-10 APP 專案架構

Step2 繪製 activity_main.xml，如圖 7-11 所示。

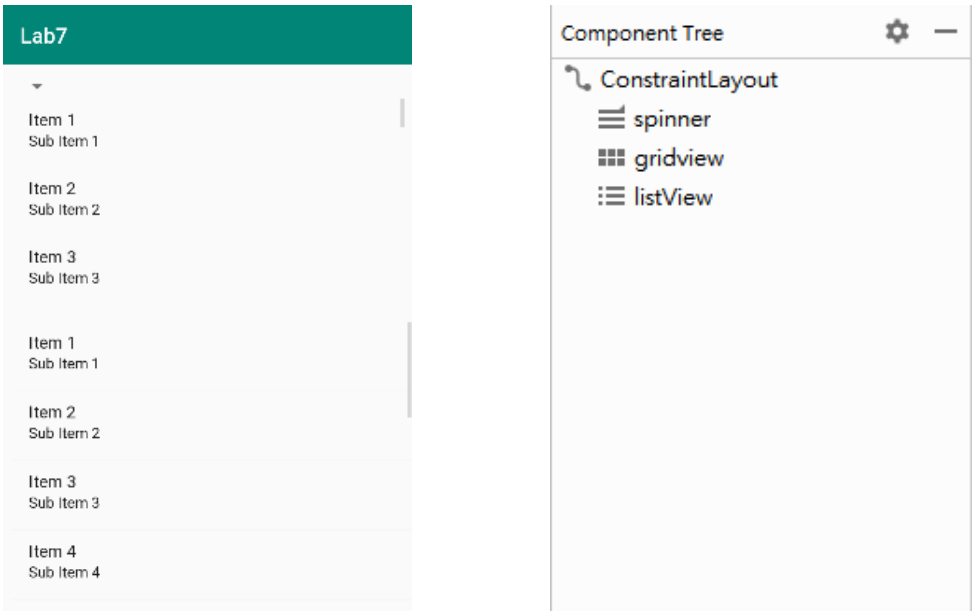

圖 7-11 MainActivity 預覽畫面（左）與布局元件樹（右）

對應的 xml 如下：

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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="match_parent"
        tools:context=".MainActivity">

    <Spinner
            android:id="@+id/spinner"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"/>

    <GridView
            android:id="@+id/gridview"
            android:layout_width="0dp"
            android:layout_height="200dp"
            android:layout_marginEnd="8dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="@+id/spinner"
            app:layout_constraintTop_toBottomOf="@+id/spinner"/>

    <ListView
            android:id="@+id/listView"
```

```
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:layout_marginTop="8dp"
            android:layout_marginStart="8dp"
            android:layout_marginEnd="2dp"
            android:layout_marginBottom="8dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/gridview"
            app:layout_constraintBottom_toBottomOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Step3 繪製 adapter_horizintal.xml，顯示客製化的畫面，如圖 7-12 所示。

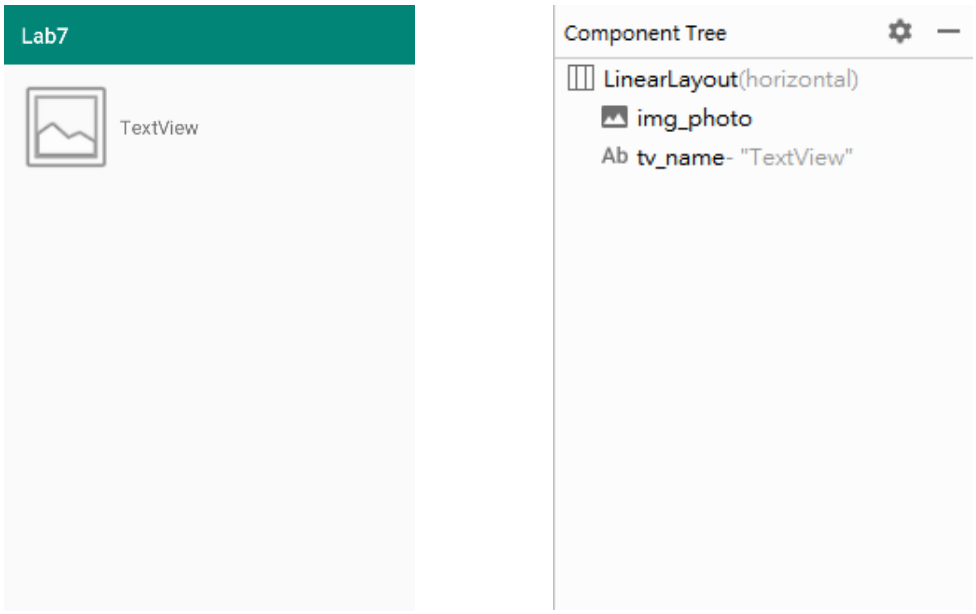

圖 7-12 水平顯示的 Adapter 預覽畫面（左）與布局元件樹（右）

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"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="8dp">

    <ImageView
            android:id="@+id/img_photo"
```

```
            android:layout_width="wrap_content"
            android:layout_height="100dp"
            android:adjustViewBounds="true"
            app:srcCompat="@android:drawable/ic_menu_gallery"/>

    <TextView
        android:id="@+id/tv_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="TextView"
        android:textSize="18sp" />
</LinearLayout>
```

Step4 繪製 adapter_vertical.xml，顯示客製化的畫面，如圖 7-13 所示。

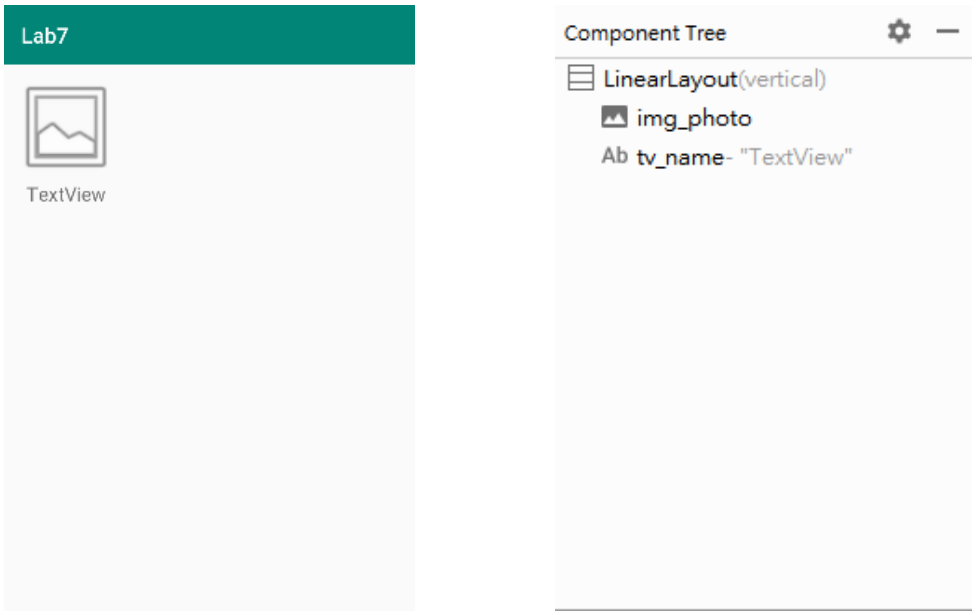

圖 7-13 垂直顯示的 Adapter 預覽畫面（左）與布局元件樹（右）

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"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="8dp">
```

```
    <ImageView
        android:id="@+id/img_photo"
        android:layout_width="wrap_content"
        android:layout_height="100dp"
        android:adjustViewBounds="true"
        android:layout_gravity="center"
        app:srcCompat="@android:drawable/ic_menu_gallery"/>

    <TextView
        android:id="@+id/tv_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="TextView"
        android:textSize="18sp" />
</LinearLayout>
```

7.2.2Adapter 程式設計

Step1 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。

```
data class Item(
      val photo: Int, //照片
      val name: String //姓名
)
```

Step2 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保存在 MyAdapter 中，避免出現取錯資料的問題。

```
class MyAdapter constructor(private val layout: Int, private
val data: ArrayList<Item>) : BaseAdapter() { //繼承 BaseAdapter
    //回傳資料來源筆數
    override fun getCount() = data.size
    //回傳某筆項目
    override fun getItem(position: Int) = data[position]
    //回傳某筆項目 id
    override fun getItemId(position: Int) = 0L
    //取得畫面元件
    override fun getView(position: Int, convertView: View?,
parent: ViewGroup?): View {

        val view = View.inflate(parent?.context, layout, null)
        //根據 position 把圖片顯示到 ImageView
        view.img_photo.setImageResource(data[position].photo)
        //根據 position 把字串顯示到 TextView
        view.tv_name.text = data[position].name
```

```
        return view //取得 Xml 畫面
    }
}
```

Step3 將 drawable 中的 PNG 圖檔資源加入到 values 的 strings.xml 中，建立一個 Integer 陣列，命名為 resourceList，如圖 7-14 所示。

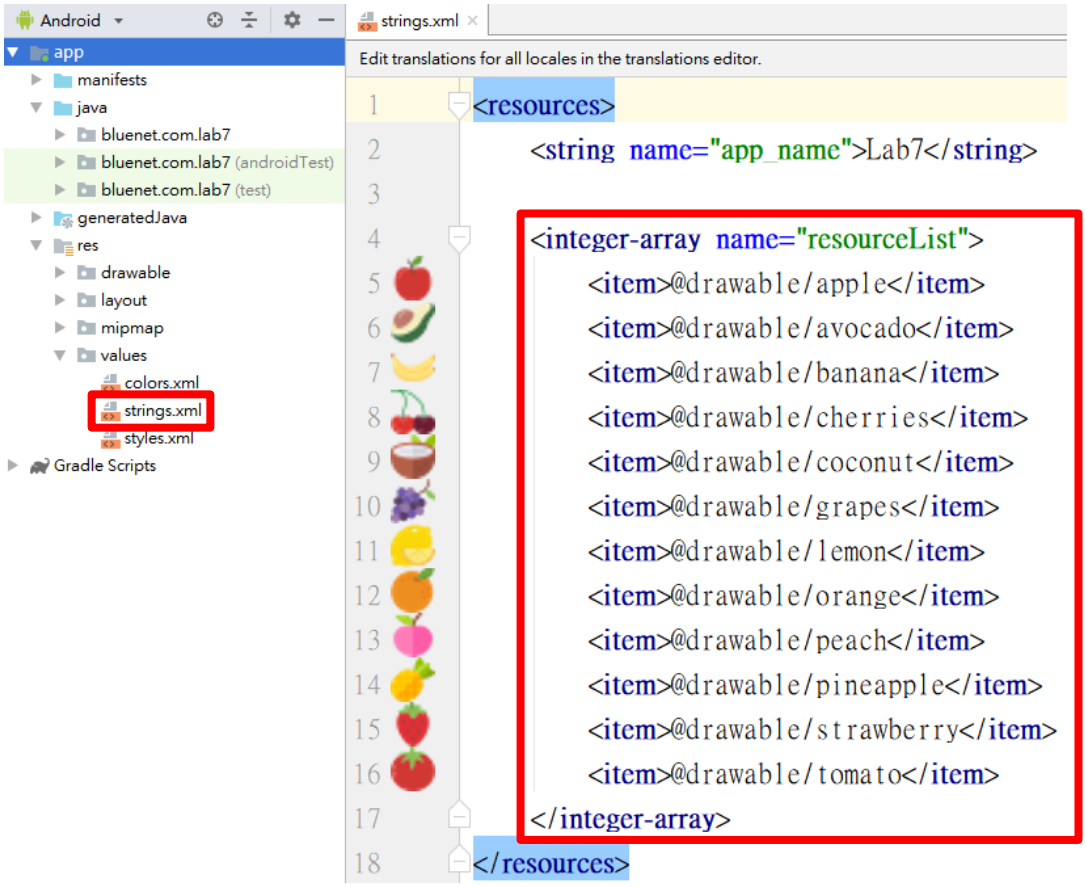

圖 7-14 建立圖檔資源陣列

Step4 將 strings.xml 中的圖檔陣列讀出，存放在 ArrayList 中，透過自定義 Adapter 顯示於 Spinner 與 GridView 中，而 ListView 則使用預設 ArrayAdapter 與 simple_list_item layout 顯示。

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        //建立資料來源（字串），並從 R 類別讀取圖檔資源
        val item = ArrayList<Item>()
        val array =
```

```
                resources.obtainTypedArray(R.array.resourceList)
        for(i in 0 until array.length())
            item.add(Item(array.getResourceId(i,0),水果${i+1}"))
        array.recycle()
        //連結 Adapter，並傳入 adapter_horizontal 作為畫面
        spinner.adapter = MyAdapter(
                            R.layout.adapter_horizontal, item)
        //設定橫向顯示列數
        gridview.numColumns = 3
        //連結 Adapter，並傳入 adapter_vertical 作為畫面
        gridview.adapter = MyAdapter(
                              R.layout.adapter_vertical, item)
        //建立 Adapter 物件，並放入字串與 simple_list_item_1.xml
        listView.adapter = ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, arrayListOf("項目 1","項目
2","項目 3","項目 4","項目 5","項目 6","項目 7","項目 8","項目 9"))
    }
}
```