Module IV
Android Advanced User Interface

- Menus
- AdapterView

Menus (1/2)

- Android提供3種Menu功能
  - Options Menu：按Menu實體按鈕
  - Context Menu：久按指定的元件不放而產生
  - Submenu：子選項

- 產生Options Menu方法如下：
  - 使用XML檔案建立Menu元件
    - <menu>標籤：建立Menu元件
    - <item>標籤：建立Menu的選項
      - android:id – 識別代號
      - Android:icon – 選項圖示
      - android:title – 選項上呈現的文字
  - 改寫與Options Menu相關的方法
    - onCreateOptionsMenu()：顯示menu時會呼叫此方法
    - onOptionsItemSelected()：選項被點選時會呼叫此方法

Menus (2/2)

- 產生Context Menu與Options Menu大同小異，唯一需注意的是要先呼叫registerForContextMenu(View view)指定在哪個元件上長按才會出現目錄

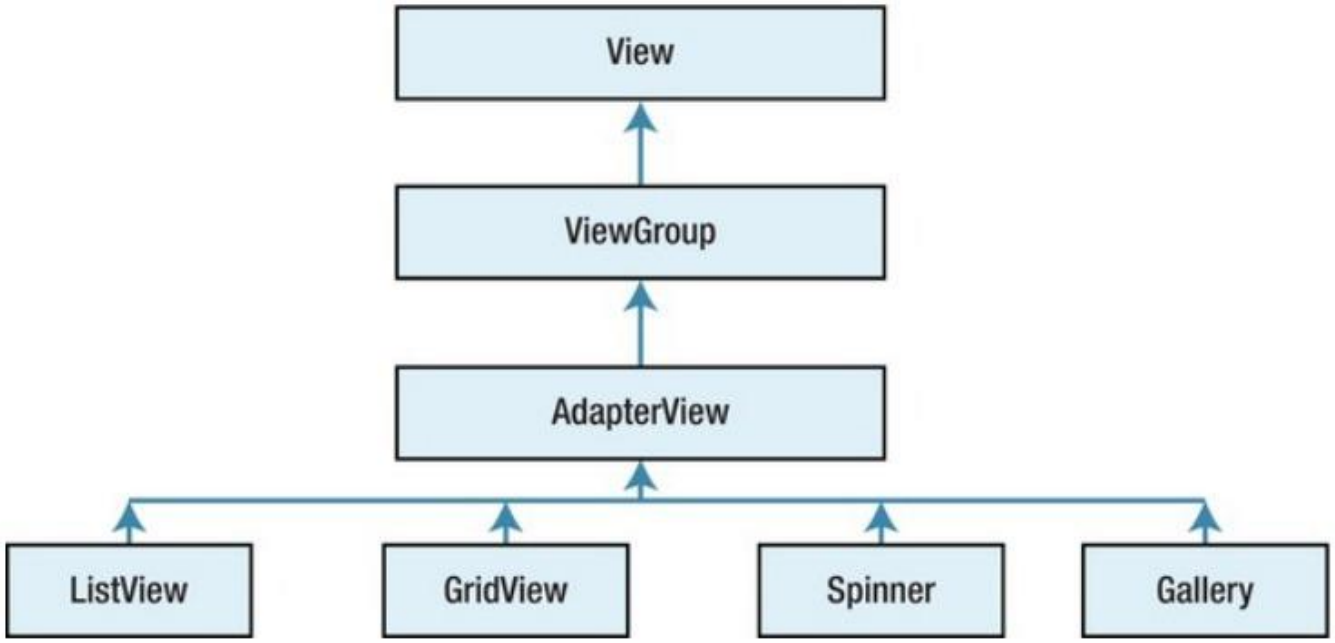
- 改寫Context Menu相關的方法
  - onCreateContextMenu()：顯示menu時會呼叫此方法
  - onContextItemSelected()：選項被點選時會呼叫此方法

- Submenu即為巢狀目錄，可透過xml檔案設定之外，也能用程式碼動態產生

範例：MenusEx

Android AdapterView (1/2)

- Class Diagram



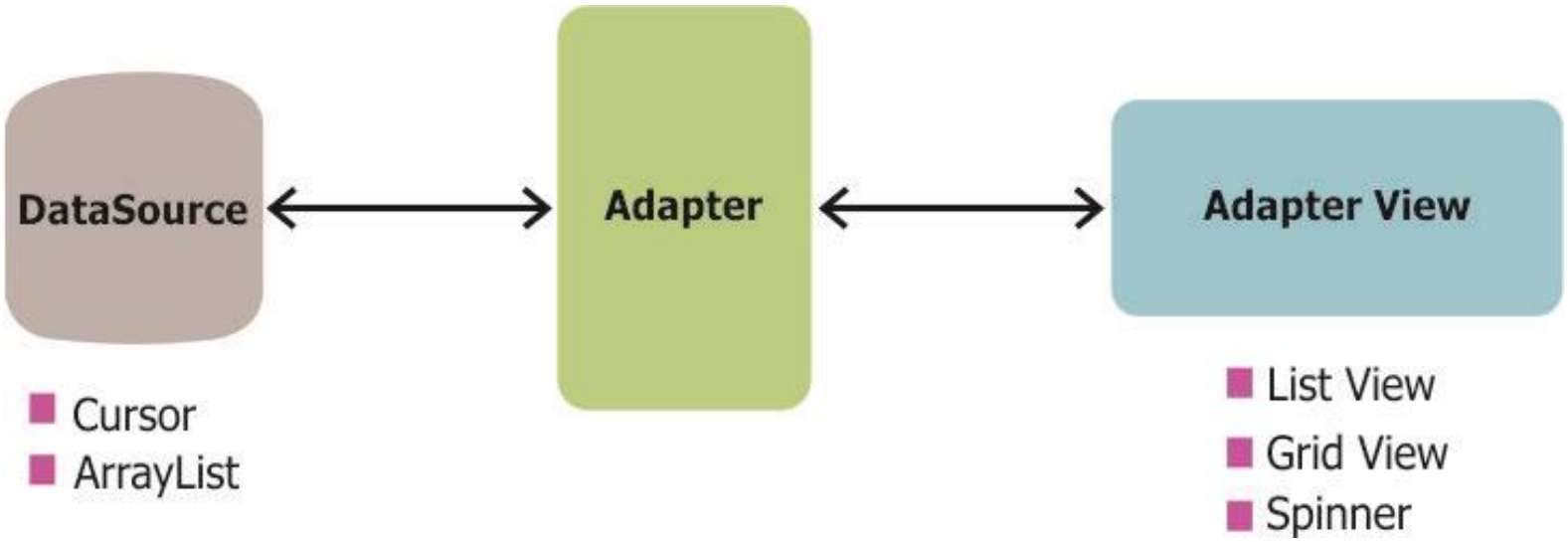
- 註：Gallery元件從Android 4.1以後不再建議使用(Deprecated)

Android AdapterView (2/2)

- What is adapter

Android AdapterView (2/2)

- Adapter in android



Spinner

- Spinner跟下拉式選單非常類似，可節省顯示空間

- 建立Spinner元件與處理事件相關：
  - 在string.xml檔建立字串陣列做為選項
  - layout檔案內使用<Spinner>建立元件
  - android:entries屬性設定選項
  - 呼叫setOnItemSelectedListener()方法註冊，並實做onItemSelected()方法以回應選項改變

- 選項來源可能是**已知固定**或是**動態產生**

範例：SpinnerEx

AutoCompleteTextView

- AutoCompleteTextView會自動列出符合的提示文字列表，方便使用者能快速完成輸入
- 提示文字列表與Spinner相同，可以在string.xml裡建立字串陣列來儲存要做為提示的文字
- android:completionThreshold指定至少要輸入多少字元才會顯示提示文字

範例：AutoCompleteTextViewEx

ListView (1/6)

- ListView可以顯示多個元件(如TextView, ImageView)，需要自訂類別並利用其屬性分別參照到對應的元件即可，建立步驟：
  - 在主要layout檔建立ListView元件
  - 建立一個layout檔並定義要在ListView顯示的元件
  - 建立BaseAdapter子類別並改寫getCount(), getItem(), getItemId(), getView()方法，系統會自動呼叫這些方法
  - 呼叫setAdapter(adapter)套用BaseAdapter設定
  - 實作OnItemClickListener.onItemClick()處理使用者點擊ListView圖片的事件

範例：ListViewEx

ListView (2/6)

- 實做BaseAdapter的四個方法：
  - public int getCount()
    回傳資料總數
  - public Object getItem(int position)
    回傳該列物件
  - public long getItemId(int position)
    較無實際應用
  - public View getView(int position, View convertView, ViewGroup parent)
    回傳每一個List要顯示的畫面內容是什麼，也是BaseAdapter實做的重點方法！

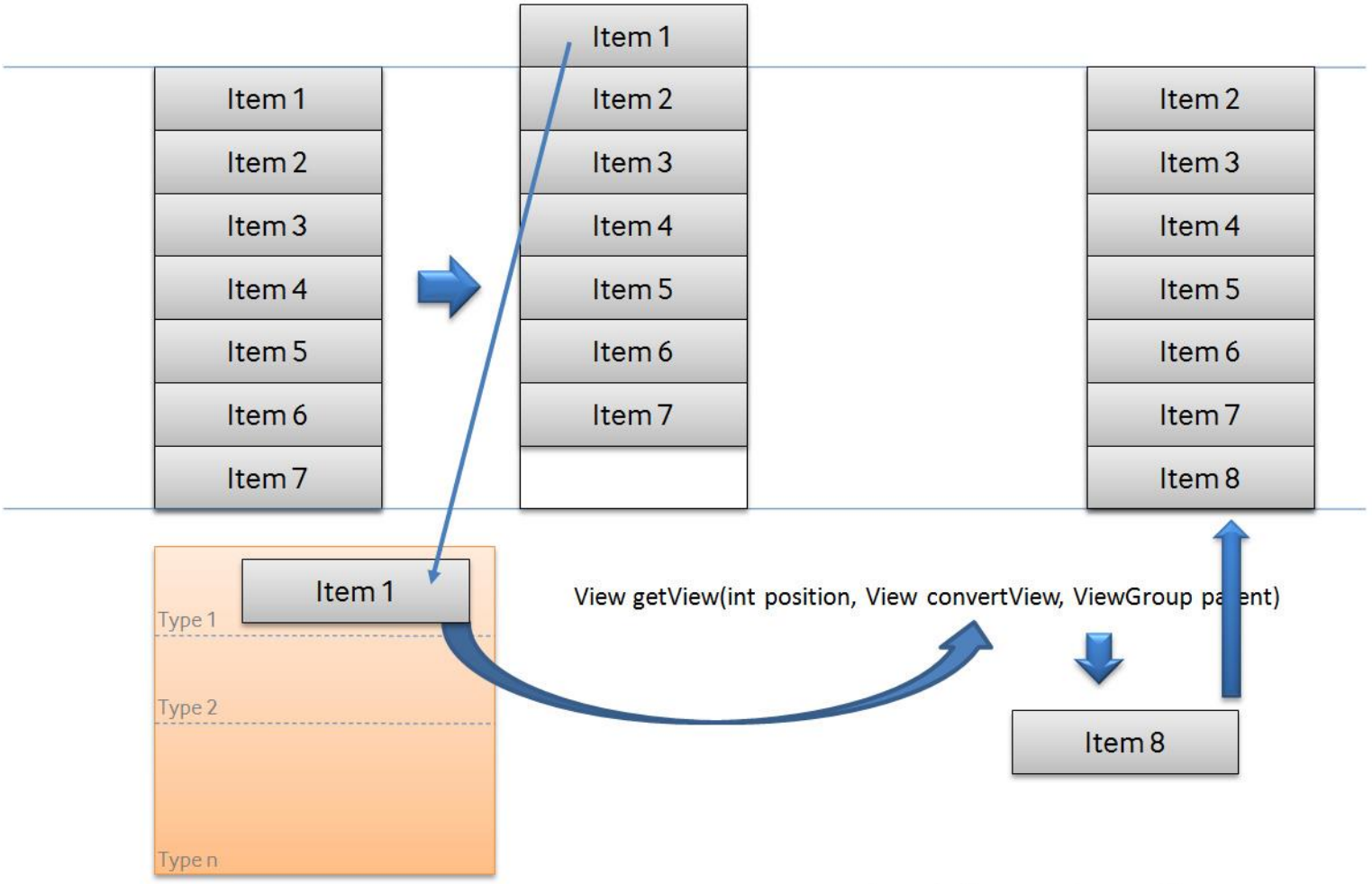
ListView (3/6)

- In getView method (NG)

```
@Override
// 依照position回傳該列資料所需呈現的UI畫面(View)
public View getView(int position, View convertView, ViewGroup parent) {
    View view = layoutInflater.inflate(R.layout.list_item, null);
    Team team = teamList.get(position);
    ImageView ivLogo = (ImageView)view.findViewById(R.id.ivLogo);
    ivLogo.setImageResource(team.getLogo());
    TextView tvId = (TextView)view.findViewById(R.id.tvId);
    tvId.setText(Integer.toString(team.getId()));
    TextView tvName = (TextView)view.findViewById(R.id.tvName);
    tvName.setText(team.getName());

    return view;
}
```

ListView (4/6)


Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7
Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7
Item 8
Item 1
Type 1
Type 2
Type n
View getView(int position, View convertView, ViewGroup parent)
Item 8
Recycler

ListView (5/6)

- In getView method (Much better…)

```
@Override
// 依照position回傳該列資料所需呈現的UI畫面(View)
public View getView(int position, View convertView, ViewGroup parent) {
    if(convertView == null) {
        convertView = layoutInflater.inflate(R.layout.list_item, null);
    }

    Team team = teamList.get(position);
    ImageView ivLogo = (ImageView)convertView.findViewById(R.id.ivLogo);
    ivLogo.setImageResource(team.getLogo());
    TextView tvId = (TextView)convertView.findViewById(R.id.tvId);
    tvId.setText(Integer.toString(team.getId()));
    TextView tvName = (TextView)convertView.findViewById(R.id.tvName);
    tvName.setText(team.getName());

    return convertView;

}
```

ListView (6/6)

- In getView method (The best yet!)

```
@Override
// 依照position回傳該列資料所需呈現的UI畫面(View)
public View getView(int position, View convertView, ViewGroup parent) {
    ViewHolder holder;
    if(convertView == null) {
        holder = new ViewHolder();
        convertView = layoutInflater.inflate(R.layout.list_item, null);
        holder.ivLogo = (ImageView)convertView.findViewById(R.id.ivLogo);
        holder.tvId = (TextView)convertView.findViewById(R.id.tvId);
        holder.tvName = (TextView)convertView.findViewById(R.id.tvName);
        convertView.setTag(holder);
    } else {
        holder = (ViewHolder)convertView.getTag();
    }

    Team team = teamList.get(position);
    holder.ivLogo.setImageResource(team.getLogo());
    holder.tvId.setText(Integer.toString(team.getId()));
    holder.tvName.setText(team.getName());

    return convertView;
}

private class ViewHolder{
    ImageView ivLogo;
    TextView tvName, tvId;
}
```

範例：ListViewEx

GridView

- 使用GridView呈現縮圖方便使用者瀏覽，是以二維方式顯示內容物
  - 準備圖像放到專案的res/drawable目錄內
  - layout檔內新增GridView元件
  - 繼承BaseAdapter類別並改寫getCount(), getItem(), getItemId(), getView()方法，系統會自動呼叫這些方法
  - 呼叫setAdapter(adapter)套用BaseAdapter設定
  - 實作OnItemClickListener.onItemClick()處理使用者點擊GridView圖片的事件

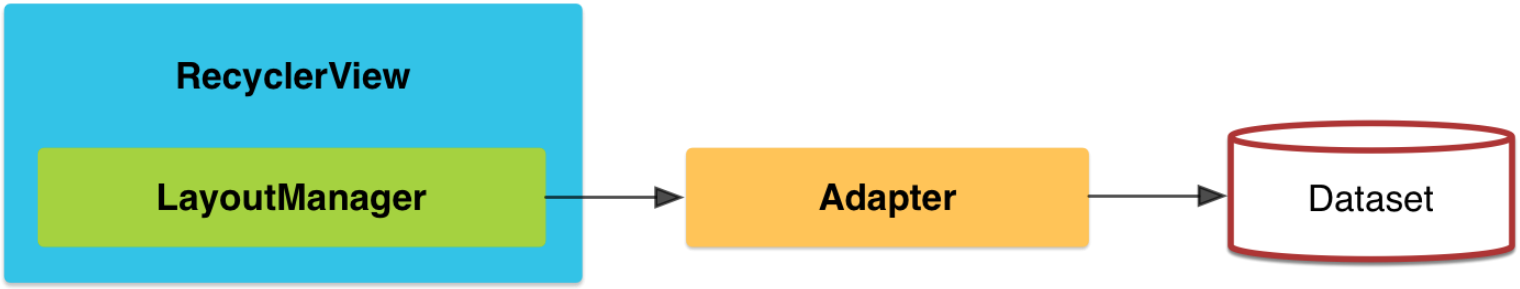
- 常用xml屬性設定
  - columnWidth：指定每一欄的寬度 (單位為dp)
  - numColumns：指定欄數 (auto_fit為自動指定)
  - verticalSpacing：指定二列垂直間距 (單位為dp)
  - horizontalSpacing：指定二欄水平間距 (單位為dp)
  - gravity：指定儲存格對齊方法 (center代表置中對齊)

範例：GridViewEx

Recycler View (1/4)

- RecyclerView
  - 比ListView更進階也具彈性
  - ViewHolder變成強制性必須實作的類別
  - 回收的速度比以往更有效率
  - 以前你只建立ListView和Adapter，現在你還多需要建立一個LayoutManager
  - LayoutManager能幫你避免過多次呼叫findViewById的所造成的資源浪費

Recycler View (2/4)



- LayoutManager屬於RecyclerView的內部元件之一，其目的用來決定RecyclerView當一個view不再顯示給使用者，要怎麼重新使用這些view資源。

- 無論是要重覆使用(reuse)或資源回收(recycle)一個view，LayoutManager都會從數據集(Dataset)去讀取出需要的資料，並且取代原view來顯示給使用者。然用這種方式去更換view能避免創建一些不需要的view、也能增進使用findViewById找資源的效能。

Recycler View (3/4)

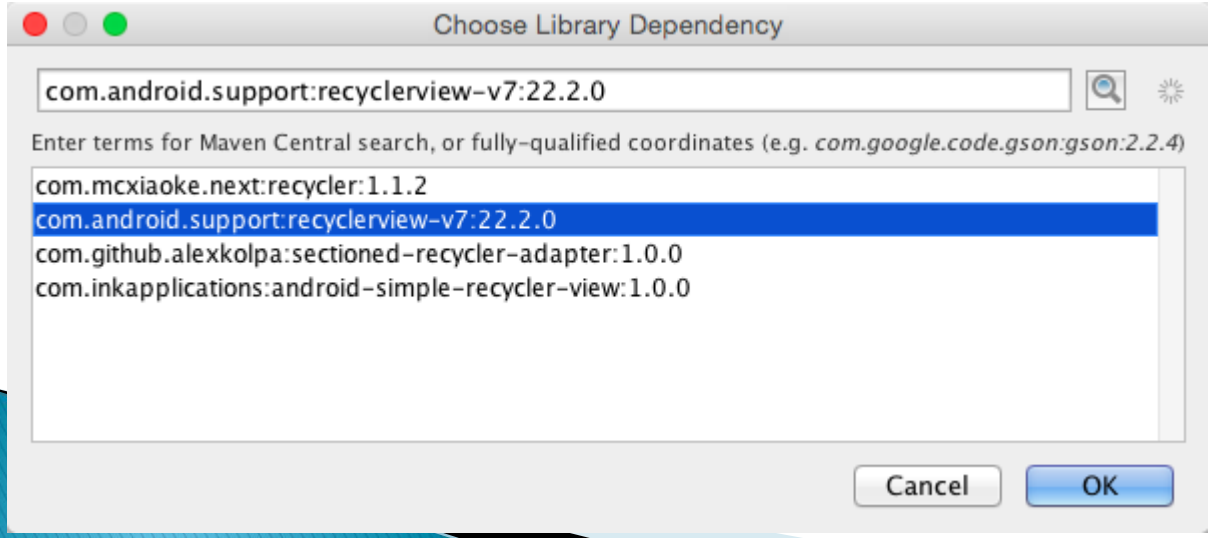
- 欲使用Recycler View，需先對自己專案加入Recycler View library
  - 方法一：

    在build.gradle檔案的dependencies加入此行

    – compile 'com.android.support:recyclerview-v7:+'
  - 方法二：

    File → Project Structure → Module → app → Dependencies →

    +/- button 新增或移除 library dependency



Recycler View (4/4)

- 建立子類別繼承Recycler.Adapter類別，並實作以下三個方法：
  - public int getItemCount()
    回傳資料總數
  - public ViewHolder onCreateViewHolder(int position)
    在此載入要呈現的layout檔，以便ViewHolder綁定元件實體
  - public void onBindViewHolder(ViewHolder holder, final int position)
    回傳每一個項目要顯示的畫面內容
- 另在此子類別裡，再建立一個子類別繼承Recycler.ViewHolder類別，做findViewById動作，綁定元件實體用

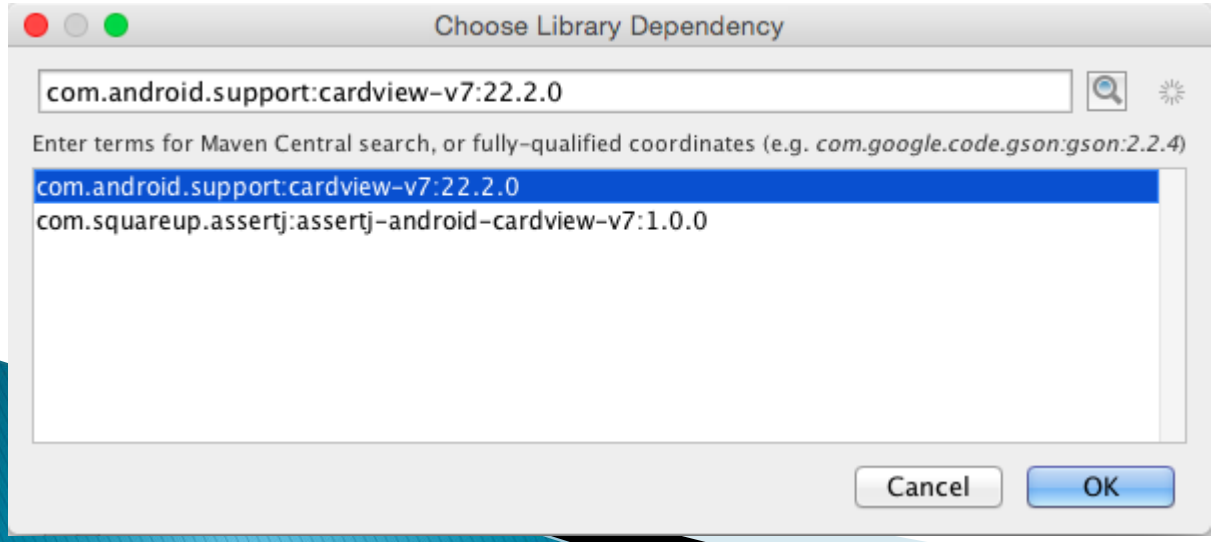
CardView (1/2)

- 它是屬於FrameLayout的子類別，讓想要顯示的資訊以類似一張一張卡片的形式呈現出來，通常搭配List容器一起使用(如RecyclerView)

- Layout檔加入<android.support.v7.widget.CardView>元件，相關xml屬性：
  - cardCornerRadius：卡片邊角的圓弧度 (單位為dp)
  - cardBackgroundColor：卡片背景顏色

CardView (2/2)

- 欲使用Card View，需先對自己專案加入Card View library
  - 方法一：
    在build.gradle檔案的dependencies加入此行
    - compile 'com.android.support:cardview-v7:+'
  - 方法二：
    File → Project Structure → Module → app → Dependencies → +/- button 新增或移除 library dependency



範例：RecyclerCardViewEx