5-2 AutoCompleteTextView

AutoCompleteTextView 非常類似 EditText，屬於文字輸入方塊；不過 AutoCompleteTextView 會在使用者輸入幾個字時就會顯示提示文字，方便使用者選取而無需輸入所有文字，是一種體貼使用者輸入的設計。

AutoCompleteTextView 的提示列表與 Spinner 的選項列表建立方式相同，需要建立字串陣列來儲存欲提示的文字。

範例 AutoCompleteTextViewDemo

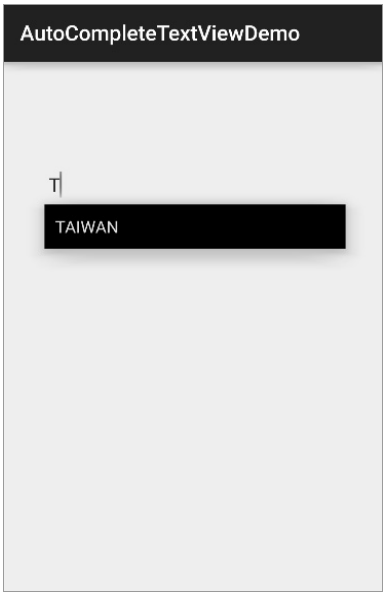

圖 5-2

範例說明：

- 輸入「T」，應用程式會作比對，並自動將符合的提示文字以列表方式呈現，方便使用者以選取方式輸入。

建立步驟：

1 STEP 使用 layout 檔案建立 AutoCompleteTextView，completionThreshold 屬性設定輸入多少個字元才會顯示提示文字，如果未設定則預設為 2 個字元。

AutoCompleteTextViewDemo > res > layout > main_activity.xml

```
<AutoCompleteTextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/actvCountry"
    android:hint="@string/text_actvCountry"
    android:completionThreshold="1"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="84dp" />
```

2 STEP 呼叫 findViewById()找到 layout 檔案上的 AutoCompleteTextView。建立 ArrayAdapter 並以字串陣列儲存提示列表上的文字；AutoCompleteTextView 再套用此 ArrayAdapter。最後 AutoCompleteTextView 註冊 OnItemClickListener 監聽器，當使用者選擇提示列表上的文字時會呼叫 onItemClick()。

AutoCompleteTextViewDemo > java > MainActivity.java

```
public class MainActivity extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_activity);

        final String[] countries = {
                "CANADA", "CHINA", "FRANCE", "GERMANY",
                "ITALY", "JAPAN", "KOREA", "TAIWAN", "UK", "US"
        };
        AutoCompleteTextView actvCountry =
                (AutoCompleteTextView) findViewById(R.id.actvCountry);
```

R.layout.list_item 是自行建立的 layout 檔案，當作選項內容的樣式

```
        ArrayAdapter<String> arrayAdapter =
                new ArrayAdapter<>(this, R.layout.list_item, countries);
        actvCountry.setAdapter(arrayAdapter);
```

AutoCompleteTextView 註冊 OnItemClickListener 監聽器，當使用者選擇提示列表上的文字時會呼叫 onItemClick()，此時呼叫 getItemAtPosition()取得使用者選取的文字並以 Toast 訊息方塊呈現

```
        actvCountry.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(
                    AdapterView<?> parent, View view, int position, long id) {
                String item = parent.getItemAtPosition(position).toString();
                Toast.makeText(
                        MainActivity.this,
                        item,
                        Toast.LENGTH_SHORT)
                        .show();
            }
        });
    }
}
```

5-3 ListView

ListView 元件屬於 AdapterView，以列表方式呈現內容，如果內容過長，使用者可以捲動畫面來瀏覽，此元件非常適合用來呈現大量資料。ListView 的每一列資料都是一個選項，而這些選項內容是由 Adapter 動態載入 layout 檔案，再將資料來源（List 或陣列）的資料取出後配置在 layout 檔案的各個 UI 元件上；換句話說，Adapter 負責管理 ListView 選項列的內容（包含值與樣式），這也是所有 AdapterView 元件的特色。當 ListView 資料內容有變時，開發者可以呼叫 BaseAdapter.notifyDataSetChanged()來刷新畫面。

範例 ListViewDemo

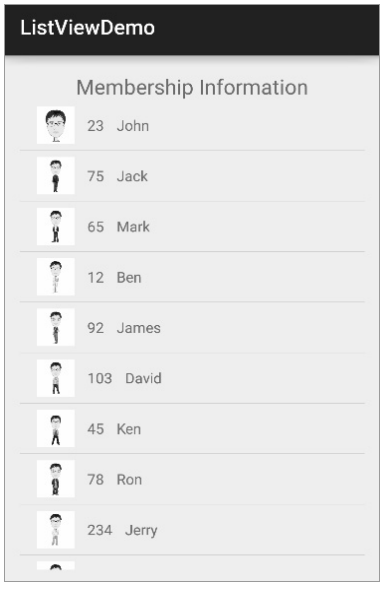

圖 5-3

範例說明：

- 主頁面有 2 個 UI 元件：TextView 顯示會員標題文字與 ListView 顯示各個會員資料
- ListView 每一筆選項列，需要另外載入 layout 檔案來配置圖片與文字等內容。此例所載入的 layout 檔案內有 1 個 ImageView 用來顯示會員照片；2 個 TextView 分別呈現會員 ID 與會員姓名。
- 點擊選項列後會以 Toast 顯示對應文字。

建立步驟：

1 STEP 建立主頁面的 layout 檔案，並在其內建立 ListView。

ListViewDemo > res > layout > main_activity.xml

```
<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"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/tvTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:textSize="20sp"
        android:text="@string/tvTitle" />

    <ListView
        android:id="@+id/lvMember"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

2 STEP 建立 ListView 各選項列所需的 layout 檔案。因為每一筆選項列的版面配置都一樣，所以只要建立一個 layout 檔案即可重複套用。在此例中，載入的 layout 檔案－listview_item.xml 其父元件為 LinearLayout，所以其實載入的是 LinearLayout；而 3 個子元件：1 個 ImageView 用來顯示會員照片；2 個 TextView 分別呈現會員 ID 與會員姓名。

ListViewDemo > res > layout > listview_item.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <ImageView
        android:id="@+id/ivImage"
        android:layout_width="48dp"
```

```
        android:layout_height="48dp"
        android:layout_marginLeft="10dp"
        android:padding="6dp" />

    <TextView
        android:id="@+id/tvId"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:padding="6dp" />

    <TextView
        android:id="@+id/tvName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:padding="6dp" />
</LinearLayout>
```

3 STEP 建立 BaseAdapter 子類別（例如 MemberAdapter），並改寫下列 4 個方法，以提供選項列內容：

- public int getCount()：提供選項列總數。
- public Object getItem(int position)：依據索引位置（position）提供該選項列對應的物件，這裡提供 Member 物件（會員物件）。
- public long getItemId(int position)：依據索引位置提供該選項列對應的 ID，這裡提供 Member ID（會員代號）。
- public view getView(int position, View convertView, parent)：依據索引位置提供該選項列對應的 View 給使用者觀看。

ListViewDemo > java > MainActivity.java

```
... // 尚有其他程式

private class MemberAdapter extends BaseAdapter {
    private LayoutInflater layoutInflater;
    private List<Member> memberList;
```

```
    public MemberAdapter(Context context) {
```

取得 LayoutInflater 物件以便之後動態載入 layout 檔案供選項列使用

```
        layoutInflater = LayoutInflater.from(context);
```

memberList 是此 ListView 的資料來源，而 Member 類別定義著會員資料如會員 ID、照片、姓名

```
        memberList = new ArrayList<>();
        memberList.add(new Member(23, R.drawable.p01, "John"));
        memberList.add(new Member(75, R.drawable.p02, "Jack"));
        memberList.add(new Member(65, R.drawable.p03, "Mark"));
        memberList.add(new Member(12, R.drawable.p04, "Ben"));
        memberList.add(new Member(92, R.drawable.p05, "James"));
        memberList.add(new Member(103, R.drawable.p06, "David"));
        memberList.add(new Member(45, R.drawable.p07, "Ken"));
        memberList.add(new Member(78, R.drawable.p08, "Ron"));
        memberList.add(new Member(234, R.drawable.p09, "Jerry"));
        memberList.add(new Member(35, R.drawable.p10, "Maggie"));
        memberList.add(new Member(57, R.drawable.p11, "Sue"));
        memberList.add(new Member(61, R.drawable.p12, "Cathy"));
    }
```

提供選項列總數，系統會依照回傳值來決定呼叫下面 getView()的次數

```
    @Override
    public int getCount() {
        return memberList.size();
    }
```

依據 position 位置提供該選項列對應物件，在此回傳代表會員的 Member 物件

```
    @Override
    public Object getItem(int position) {
        return memberList.get(position);
    }
```

依據 position 位置提供該選項列對應的 ID，在此回傳會員 ID

```
    @Override
    public long getItemId(int position) {
        return memberList.get(position).getId();
    }
```

getView()是依據 position 位置提供該選項列對應的 View。

一開始畫面尚未呈現時，converView 為 null，呼叫 inflate()載入 R.layout.listview_item 檔案其實就是載入 LinearLayout 這個 View。

畫面呈現時，使用者可以看到 ListView 畫面，當使用者向下滑動一列，原本第一列會被滑出畫面，被滑出選項列的 View 會自動傳遞給 convertView，所以不會為 null，可以重複利用該 View，只要將值替換成滑入選項列的值即可

```
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        if (convertView == null) {
            convertView = layoutInflater.inflate(R.layout.listview_item, parent,
false);
        }

依照 position 取得 memberList 內的 member 物件
        Member member = memberList.get(position);
找到 convertView 子元件 imageView，並指定欲顯示的圖片
        ImageView ivImage = (ImageView) convertView
                .findViewById(R.id.ivImage);
        ivImage.setImageResource(member.getImage());
找到 convertView 子元件 textView，並顯示會員 ID 與姓名
        TextView tvId = (TextView) convertView
                .findViewById(R.id.tvId);
        tvId.setText(String.valueOf(member.getId()));

        TextView tvName = (TextView) convertView
                .findViewById(R.id.tvName);
        tvName.setText(member.getName());
在此範例，回傳 convertView 其實就是回傳 LinearLayout（參看 listview_item.xml）
        return convertView;
    }
}
```

4 STEP ListView 呼叫 setAdapter() 套用 BaseAdapter 物件。註冊 OnItemClickListener 監聽器，當使用者點擊選項列時會呼叫 onItemClick()。

ListViewDemo > java > MainActivity.java

```
public class MainActivity extends ActionBarActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_activity);
```

ListView 呼叫 setAdapter()並套用建立好的 MemberAdapter

```
        ListView lvMember = (ListView) findViewById(R.id.lvMember);
        lvMember.setAdapter(new MemberAdapter(this));
```

ListView 註冊 OnItemClickListener 監聽器，當使用者點擊選項列時會呼叫 onItemClick()。

parent－被點擊的 ListView

view－被點擊的選項列所載入的 layout 內容，在此為 listview_item.xml 內的 LinearLayout 元件

position－被點擊的索引位置

id－實作 BaseAdapter.getItemId()所回傳的 ID

```
        lvMember.setOnItemClickListener(new AdapterView.OnItemClickListener() {
           @Override
           public void onItemClick(AdapterView<?> parent, View view,
                                   int position, long id) {
```

呼叫 getItemAtPosition()會取得 BaseAdapter.getItem()所回傳的物件，在此為會員物件，之後以 Toast 呈現此會員相關資訊

```
                Member member = (Member) parent.getItemAtPosition(position);
                String text = "ID = " + member.getId() +
                        ", name = " + member.getName();
                Toast.makeText(MainActivity.this, text, Toast.LENGTH_SHORT).show();
            }
        });
    }

... // 尚有其他程式
}
```

5-4 GridView

GridView 以格子（grid）方式呈現資料，與 ListView 以列方式呈現有所不同，但除此之外，無論使用 BaseAdapter 載入選項內容方式，或是點擊選項後的事件處理方式以及刷新畫面的方式可說是完全相同，所以請直接參看 5-3 ListView 說明，不再贅述！

範例 GridViewDemo

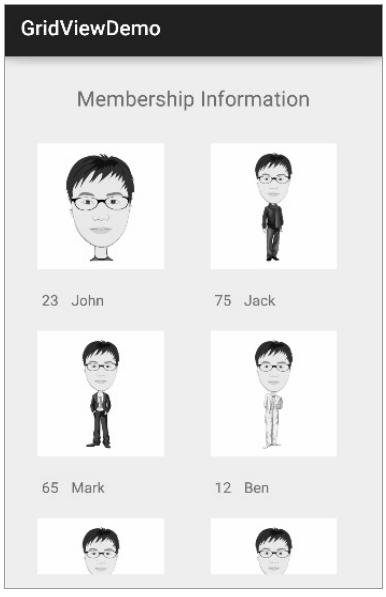

圖 5-4

範例說明：

- 主頁面有 2 個 UI 元件：TextView 顯示會員標題文字與 GridView 顯示各個會員資料
- GridView 每一個選項格，需要另外載入 layout 檔案來配置圖片與文字等內容。此例所載入的 layout 檔案內有 1 個 ImageView 用來顯示會員照片；2 個 TextView 分別呈現會員 ID 與會員姓名。
- 點擊選項格後會以 Toast 顯示對應圖片。

建立步驟：

完全與前述 ListView 相同，不再贅述。唯一不同的地方是點擊選項格時，此例會以 Toast 顯示圖片，而之前僅以 Toast 顯示文字，說明如下。

GridViewDemo > java > MainActivity.java

```
public class MainActivity extends ActionBarActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_activity);
        GridView gvMember = (GridView) findViewById(R.id.gvMember);
        gvMember.setAdapter(new MemberAdapter(this));
        gvMember.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view,
                                    int position, long id) {
                Member member = (Member) parent.getItemAtPosition(position);
建立 ImageView 並放上會員照片;再建立 Toast 並呼叫 setView()套用該 ImageView 即可顯示照片
                ImageView imageView = new ImageView(MainActivity.this);
                imageView.setImageResource(member.getImage());
                Toast toast = new Toast(MainActivity.this);
                toast.setView(imageView);
                toast.setDuration(Toast.LENGTH_SHORT);
                toast.show();
            }
        });
    }
  ...
}
```

5-5 CardView 與 RecyclerView

Android 5.0 時發表了 2 個新 UI 元件：CardView 與 RecyclerView，它們都屬於 support 函式庫的成員，所以可以向前相容，換句話說，舊版的 Android 裝置也可以呈現這 2 種 UI 元件。CardView 是 FrameLayout 的子類別，特色是可以設定圓角與陰影程度；而 RecyclerView 則非常類似 ListView/GridView，以有限的視窗大小呈現大量資料。可以將 CardView 置入 RecyclerView 內做更豐富的呈現，如圖 5-5。

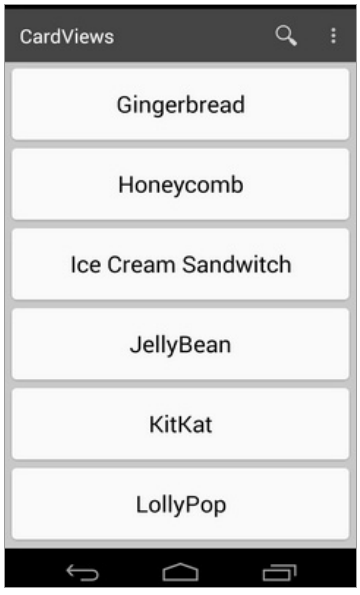

圖 5-5

CardView

CardView 屬於 FrameLayout，但比原來的 FrameLayout 多了圓角與陰影這 2 種設定。雖然 CardView 是最近才發表的 UI 元件，但由其完整名稱 android.**support**.v7.widget.CardView 可知它屬於 support 函式庫，所以可以向前相容。關於 CardView 的 2 個重要設定說明如下：

- 圓角設定：可以設定 FrameLayout 邊角的圓弧程度，在 layout 檔案使用 cardCornerRadius 屬性；在程式碼則呼叫 setRadius ()來設定。
- 陰影設定：可以設定 FrameLayout 周圍的陰影程度，在 layout 檔案使用 cardElevation 屬性；在程式碼則呼叫 setMaxCardElevation()。

RecyclerView

RecyclerView 就如其名會自動重複利用選項的 View 來呈現新的而且同樣式的選項。例如畫面上只可顯示 10 個選項，當使用者滑到第 11 個選項時，比較好的做法是將第 1 個選項的 View 放在暫存區供滑進來的第 11 個選項使用，

因為它們的樣式一樣，只不過值不同而已。這樣可以比較有效利用記憶體已存放的資料，而且可以提升執行效能。

RecyclerView 最大特色就是將 layout 設定抽離出來，可以直接呼叫 setLayoutManager()設定 layout 樣式，如果搭配 LinearLayoutManager，呈現的樣子就會幾乎跟 ListView 一樣；如果搭配 GridLayoutManager，就會如同 GridView 一般。最有趣的是 StaggeredGridLayoutManager，樣子像 GridView，但是可以水平滑動。

RecyclerView 無法像 ListView/GridView 一樣註冊 OnItemClickListener；如果仍舊想要監聽選項是否被點擊，可以將選項的 View 註冊我們熟悉的 OnClickListener 並透過 getAdapterPosition()來取得被點擊項目的位置。

當資料內容有變時，開發者可以呼叫 RecyclerView.Adapter.notifyDataSetChanged()來刷新畫面。

範例 RecyclerCardViewDemo

圖 5-6

範例說明：

- 主頁面有 2 個 UI 元件：TextView 顯示會員標題文字與 RecyclerView 顯示各個會員資料
- RecyclerView 每一個選項格，需要另外載入 layout 檔案來配置圖片與文字等內容。此例所載入的 layout 檔案內有 1 個 ImageView 用來顯示會員照片；2 個 TextView 分別呈現會員 ID 與會員姓名。
- 使用者可以左右滑動 RecyclerView，點擊選項格後會以 Toast 顯示對應圖片。

建立步驟：

1 STEP build.gradle 檔案內新增 cardview, recyclerview 套件：Android Studio 主選單 File > Project Structure > app > Dependencies > 點擊新增按鈕，新增 com.android.support:cardview 與 com.android.support:recyclerview，如圖 5-7。

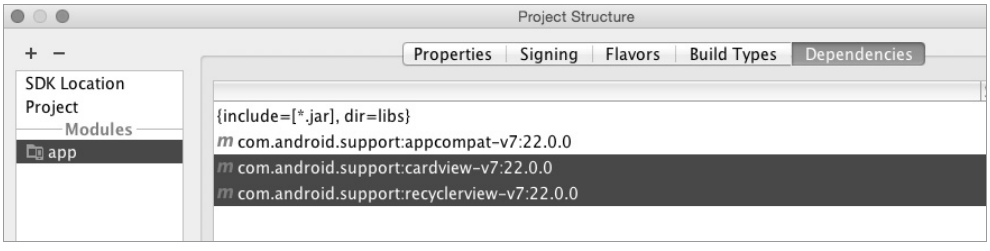

圖 5-7

2 STEP 建立主頁面的 layout 檔案，並在其內建立 RecyclerView。因為 RecyclerView 不屬於 android.widget 套件，所以必須輸入完整名稱 android.support.v7.widget.RecyclerView。

RecyclerCardViewDemo > res > layout > main_activity.xml

```
<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"
```

```
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/tvTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:textSize="20sp"
        android:text="@string/tvTitle" />

    <android.support.v7.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

3 STEP 建立 RecyclerView 選項所需的 layout 檔案。因為每一個選項樣式都相同，所以只要建立一個 layout 檔案即可重複套用。在此例中為了要搭配 CardView，所以建立 CardView 元件；因為也不屬於 android.widget 套件，所以也必須使用完整名稱 android.support.v7.widget.CardView。因為 CardView 使用到不同的名稱空間"http://schemas.android.com/apk/res-auto"，所以必須加入，並以 card_view 名稱代表。CardView 重要屬性有 cardBackgroundColor 用來設定背景色，cardCornerRadius 設定圓角弧度，cardElevation 設定陰影。

RecyclerCardViewDemo > res > layout > recyclerview_cardview_item.xml

```
<android.support.v7.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:id="@+id/cardview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    android:padding="6dp"
    card_view:cardBackgroundColor="#ffdddddd"
    card_view:cardCornerRadius="28dp"
    card_view:cardElevation="6dp"
    android:layout_margin="6dp">

    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <ImageView
            android:id="@+id/ivImage"
            android:layout_width="120dp"
            android:layout_height="160dp"
            android:layout_marginLeft="16dp" />

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal">

            <TextView
                android:id="@+id/tvId"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginLeft="20dp"
                android:layout_marginBottom="12dp" />

            <TextView
                android:id="@+id/tvName"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginLeft="24dp"
                android:layout_marginBottom="12dp" />
        </LinearLayout>

    </LinearLayout>
</android.support.v7.widget.CardView>
```

4 STEP RecyclerView 呼叫 setLayoutManager() 套用 RecyclerView.LayoutManager 提供的版面配置樣式。RecyclerView 呼叫 setAdapter() 套用 RecyclerView.Adapter 提供的選項內容設定。建立 RecyclerView.Adapter 子類別（例如 MemberAdapter），並在其內建立 RecyclerView.ViewHolder 子類別（例如 MemberAdapter.ViewHolder），ViewHolder 目的在於暫存 RecyclerView 選項的 View，以方便之後相同樣式的選項重複利用。另外 RecyclerView.Adapter 子類別還需要改寫下列 3 個方法，以提供選項內容：

- public int getItemCount()：提供 RecyclerView 選項總數。
- public ViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType)：當 RecyclerView 需要一個 View 來顯示特定選項內容時會呼叫此方法，此時要提供一個 View 給 ViewHolder 保存著，然後回傳這個 ViewHolder 讓 RecyclerView 使用。
- public void onBindViewHolder(ViewHolder viewHolder, int position)：要顯示 RecyclerView 特定位置（position）的選項內容時會呼叫此方法，此時要將 ViewHolder 內保存的各個 View 設定好要呈現的資料。

RecyclerCardViewDemo > java > MainActivity.java

```
...

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_activity);
        RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recyclerView);
呼叫 setLayoutManager()設定 layout 樣式，StaggeredGridLayoutManager 的樣式像 GridView，StaggeredGridLayoutManager 建構式第 1 個參數設定為 2 代表 2 列（或 2 欄），第 2 個參數為 HORIZONTAL 代表為水平走向
        recyclerView.setLayoutManager(
                new StaggeredGridLayoutManager(
                        2, StaggeredGridLayoutManager.HORIZONTAL));
memberList 是資料來源，而 Member 類別定義著會員資料如會員 ID、照片、姓名
        List<Member> memberList = new ArrayList<>();
```

```
        memberList.add(new Member(92, R.drawable.p05, "James"));
        memberList.add(new Member(103, R.drawable.p06, "David"));
        memberList.add(new Member(234, R.drawable.p09, "Jerry"));
        memberList.add(new Member(35, R.drawable.p10, "Maggie"));
        memberList.add(new Member(23, R.drawable.p01, "John"));
        memberList.add(new Member(75, R.drawable.p02, "Jack"));
        memberList.add(new Member(65, R.drawable.p03, "Mark"));
        memberList.add(new Member(12, R.drawable.p04, "Ben"));
        memberList.add(new Member(45, R.drawable.p07, "Ken"));
        memberList.add(new Member(78, R.drawable.p08, "Ron"));
        memberList.add(new Member(57, R.drawable.p11, "Sue"));
        memberList.add(new Member(61, R.drawable.p12, "Cathy"));
```

呼叫 MemberAdapter 建構式並傳入 memberList 會員清單資料以建立 MemberAdapter 物件後供 RecyclerView 套用

```
        recyclerView.setAdapter(new MemberAdapter(this, memberList));
    }
```

RecyclerView 要透過 RecyclerView.Adapter 來處理欲顯示的選項內容，必須建立 RecyclerView.Adapter 子類別並改寫對應的方法

```
    private class MemberAdapter extends
            RecyclerView.Adapter<MemberAdapter.ViewHolder> {
        private Context context;
        private LayoutInflater layoutInflater;
        private List<Member> memberList;

        public MemberAdapter(Context context, List<Member> memberList) {
            this.context = context;
```

初始化 layoutInflater 以載入 layout 檔案；而 memberList 儲存著欲顯示的資料

```
            layoutInflater = LayoutInflater.from(context);
            this.memberList = memberList;
        }
```

建立 RecyclerView.ViewHolder 的子類別以設定欲參照到的 View，便於之後使用

```
        public class ViewHolder extends RecyclerView.ViewHolder {
            ImageView ivImage;
            TextView tvId, tvName;
            View itemView;
```

呼叫 ViewHolder 建構式必須提供 RecyclerView 其中一個選項的 View

```
            public ViewHolder(View itemView) {
                super(itemView);
```

```
            this.itemView = itemView;
            ivImage = (ImageView) itemView.findViewById(R.id.ivImage);
            tvId = (TextView) itemView.findViewById(R.id.tvId);
            tvName = (TextView) itemView.findViewById(R.id.tvName);
        }
    }
```

提供 RecyclerView 選項總數

```
    @Override
    public int getItemCount() {
        return memberList.size();
    }
```

提供一個選項所需的 View，可以透過 LayoutInflater 載入，再透過呼叫 ViewHolder 建構式將選項的 View 傳給 ViewHolder

```
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType) {
        View itemView = layoutInflater.inflate(
                R.layout.recyclerview_cardview_item, viewGroup, false);
        return new ViewHolder(itemView);
    }
```

要顯示 RecyclerView 指定位置（position）選項的資料時會呼叫此方法，開發者應該依照 position 提供 Member 物件，並將資料顯示在 ViewHolder 參照到的 View 上

```
    @Override
    public void onBindViewHolder(ViewHolder viewHolder, final int position) {
        Member member = memberList.get(position);
        viewHolder.ivImage.setImageResource(member.getImage());
        viewHolder.tvId.setText(String.valueOf(member.getId()));
        viewHolder.tvName.setText(member.getName());
```

RecyclerView 無法像 ListView/GridView 一樣註冊 OnItemClickListener；如果仍舊想要監聽選項是否被點擊，可以將選項的 View 註冊 OnClickListener

```
        viewHolder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
```

使用者點擊 RecyclerView 的一個選項時，會 Toast 該選項所代表的會員照片

```
            public void onClick(View v) {
                ImageView imageView = new ImageView(context);
                imageView.setImageResource(member.getImage());
                Toast toast = new Toast(context);
                toast.setView(imageView);
                toast.setDuration(Toast.LENGTH_SHORT);
```

```
                toast.show();
            }
        });
    }
}
```

5-6 自訂 View 元件與 2D 繪圖

當函式庫沒有提供開發者所需要的 UI 元件時，開發者可以自行定義，但自行定義的 UI 元件仍舊必須繼承 View 類別並改寫 onDraw()讓 Android 系統可以繪製此自訂元件。關於繪圖部分可以使用 Android API 提供的 2D 繪圖功能，套件名稱為 android.graphics，常用到的類別為 Paint（繪圖功能）與 Canvas（畫布功能）。

範例 Draw2dDemo

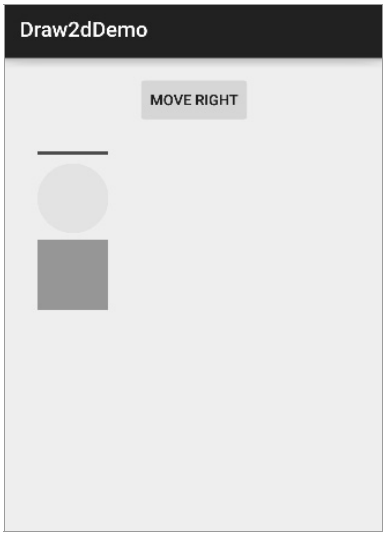

圖 5-8

範例說明：

- 按下「MOVE RIGHT」按鈕會讓下面的幾何圖形向右移動。

建立步驟：

1 STEP 繼承 View 類別並改寫 onDraw()：想繪圖必須有個可顯示的元件供繪製，可以自行定義類別（例如 GeometricView 類別）去繼承 View 類別，並且改寫 onDraw()，將想要繪製的圖形置入 onDraw()方法內。除此之外還需建立至少 2 個建構式方便開發者可以用程式碼或 layout 檔案來建立此 UI 元件。

Draw2dDemo > java > GeometricView.java

```
public class GeometricView extends View {
    private int offset = 0;
    private Paint paint = new Paint();

此建構式方便直接使用程式碼建立 GeometricView 元件
    public GeometricView(Context context) {
        super(context);
    }

透過 layout 檔案建立 GeometricView 元件會呼叫此建構式，在 layout 檔案使用到的屬性會傳遞給 attrs 參數
    public GeometricView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

呼叫此方法並傳遞偏移量給 offset 參數，會在 onDraw()繪圖時使用到
    public void setOffset(int offset) {
        this.offset = offset;
    }

    @Override
    protected void onDraw(Canvas canvas) {
paint 呼叫 setColor()設定顏色、setStrokeWidth()設定線的粗細
        paint.setColor(Color.RED);
        paint.setStrokeWidth(10);
paint 呼叫 drawLine()畫線，需提供起點與終點的 x, y 座標；
```

drawCircle()畫圓，需提供圓點的x, y座標與半徑長度；
drawRect()畫方形，需提供左、上、右、下四條邊線的座標

```
        canvas.drawLine(10 + offset, 10, 210 + offset, 10, paint);
        paint.setColor(Color.YELLOW);
        canvas.drawCircle(110 + offset, 140, 100, paint);
        paint.setColor(Color.GREEN);
        canvas.drawRect(10 + offset, 260, 210 + offset, 460, paint);
    }
}
```

2 STEP 以 layout 檔案建立自行定義的 GeometricView 元件會自動呼叫前述的 GeometricView(Context, AttributeSet) 建構式。因為不屬於 android.widget 套件，所以必須輸入完整名稱 idv.ron.draw2ddemo.GeometricView。

Draw2dDemo > res > layout > main_activity.xml

```
<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"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/btOffset"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="@string/text_btOffset"
        android:onClick="onOffsetClick" />

    <idv.ron.draw2ddemo.GeometricView
        android:id="@+id/geometricView"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp" />

</LinearLayout>
```

3 STEP 呼叫 View.invalidate()重繪元件：如果想要重新繪製 UI 元件，該元件呼叫 invalidate()，系統會先廢棄原來的畫布然後再次呼叫 onDraw()並提供新的畫布，以重新繪製此元件的內容。

Draw2dDemo > java > MainActivity.java

```
public class MainActivity extends ActionBarActivity {
    private GeometricView geometricView;
    private int offset = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_activity);
        geometricView = (GeometricView) findViewById(R.id.geometricView);
    }
```

使用者每按下一次 Move Right 按鈕時會將偏移量+10，也就是向右移動 10 像素，需要呼叫 invalidate()廢棄原來在 GeometricView 元件上的畫布；系統會自動呼叫 onDraw()並傳送新的畫布以便重新繪製

```
    public void onOffsetClick(View view) {
        if (geometricView != null) {
            offset += 10;
            geometricView.setOffset(offset);
            geometricView.invalidate();
        }
    }
}
```