

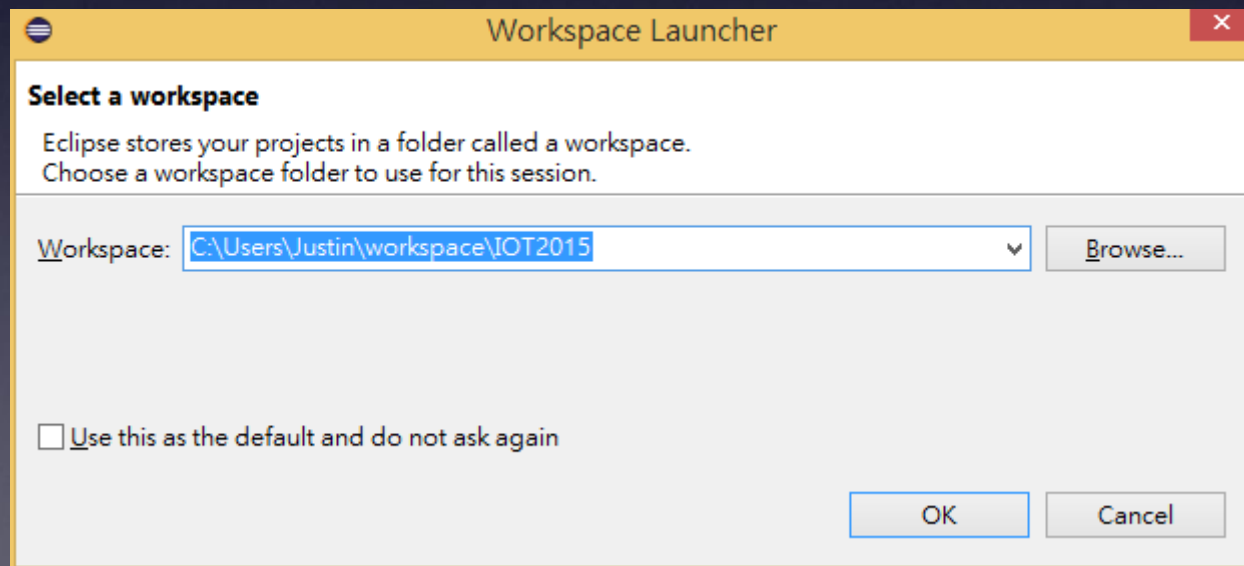
# 第一章

# ANDROID基本教學

建立WORKSPACE

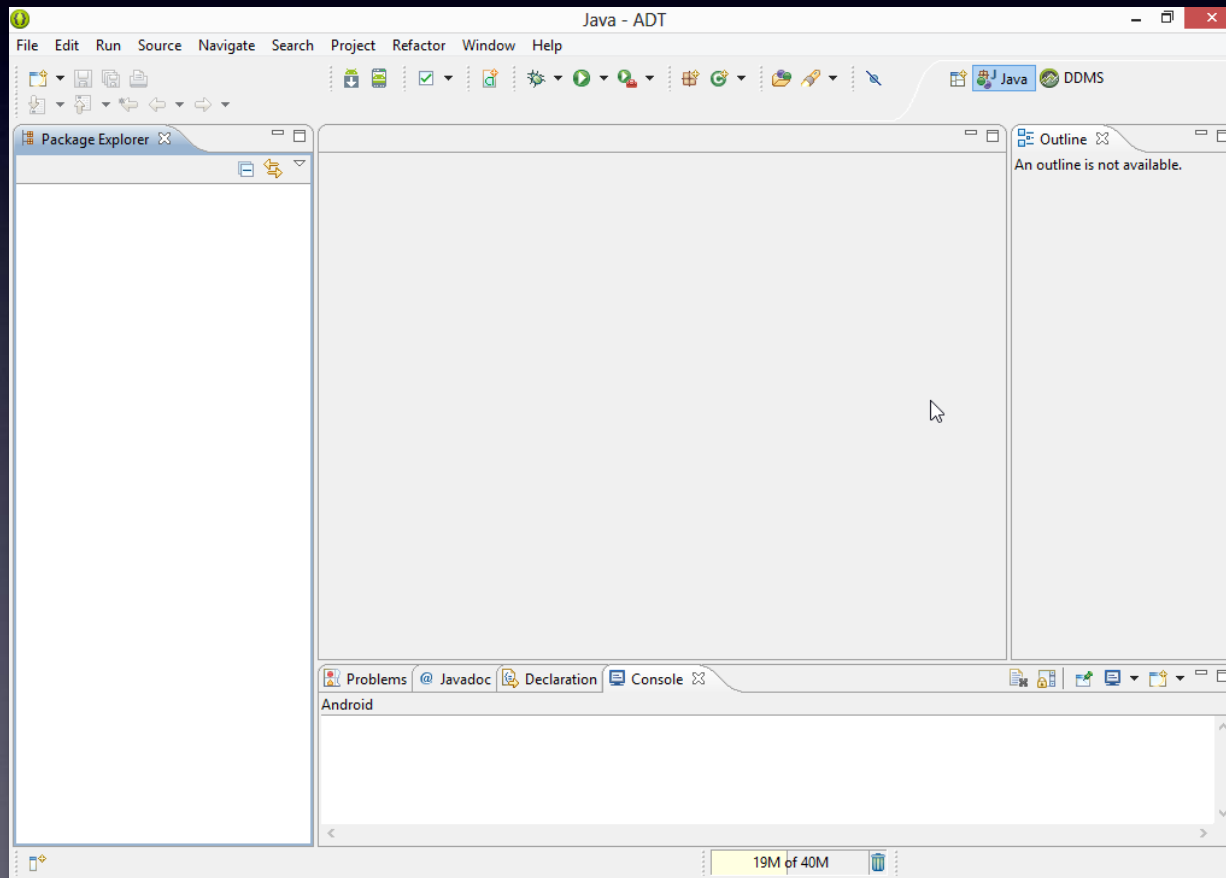
# 開啟第一個WORKSPACE

- Eclipse是以專案型式來管理程式碼
- workspace表示眾多專案集合的地方
- 可以針對不同的專案，或不同的時間，建立不同的workspace



# 開啟第一個WORKSPACE

- 恭喜你，開啟成功了



# ECLIPSE與SDK說明

Android ADT  
工具區

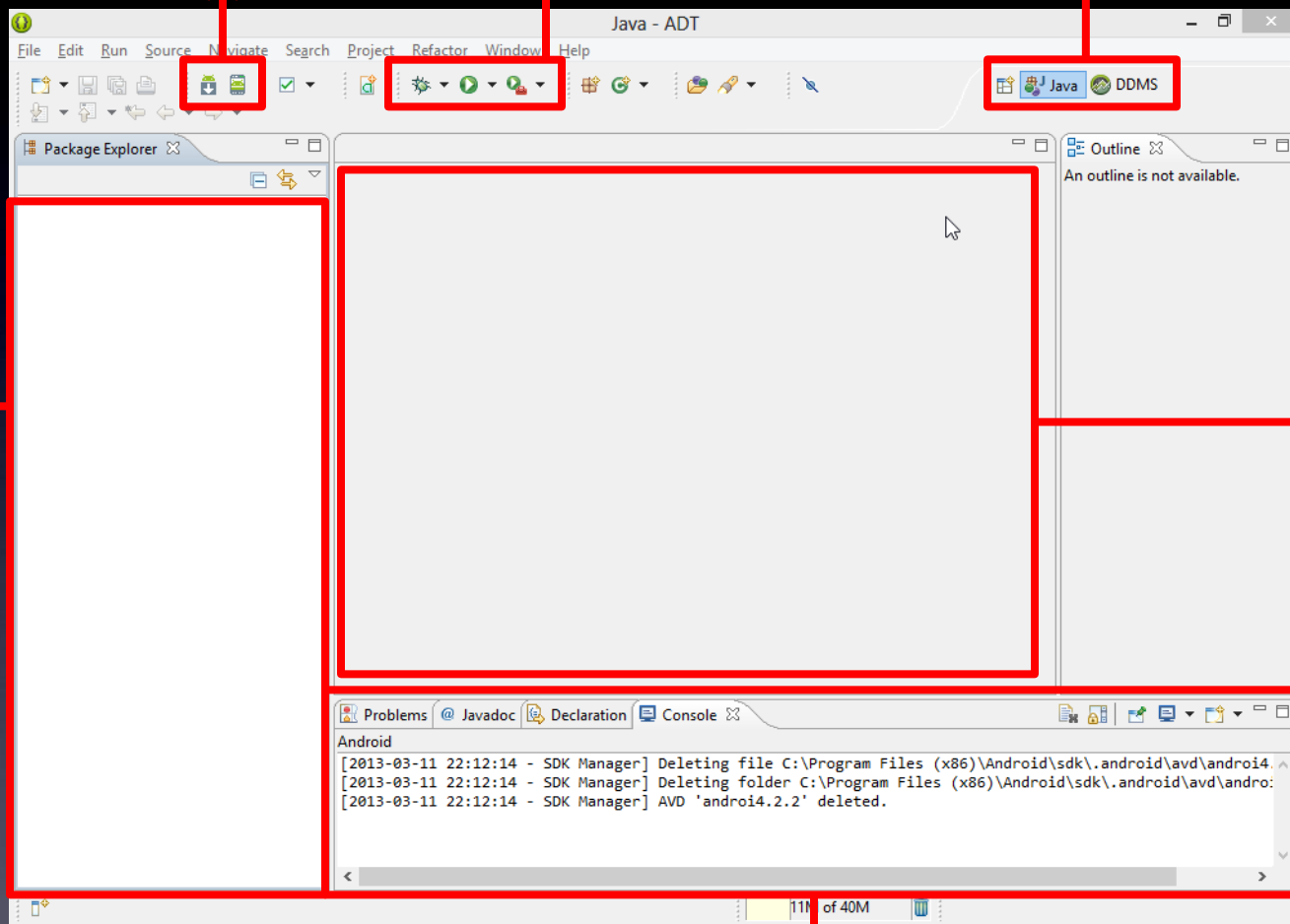
專案編譯  
除錯工具

切換模式

專案  
列表區

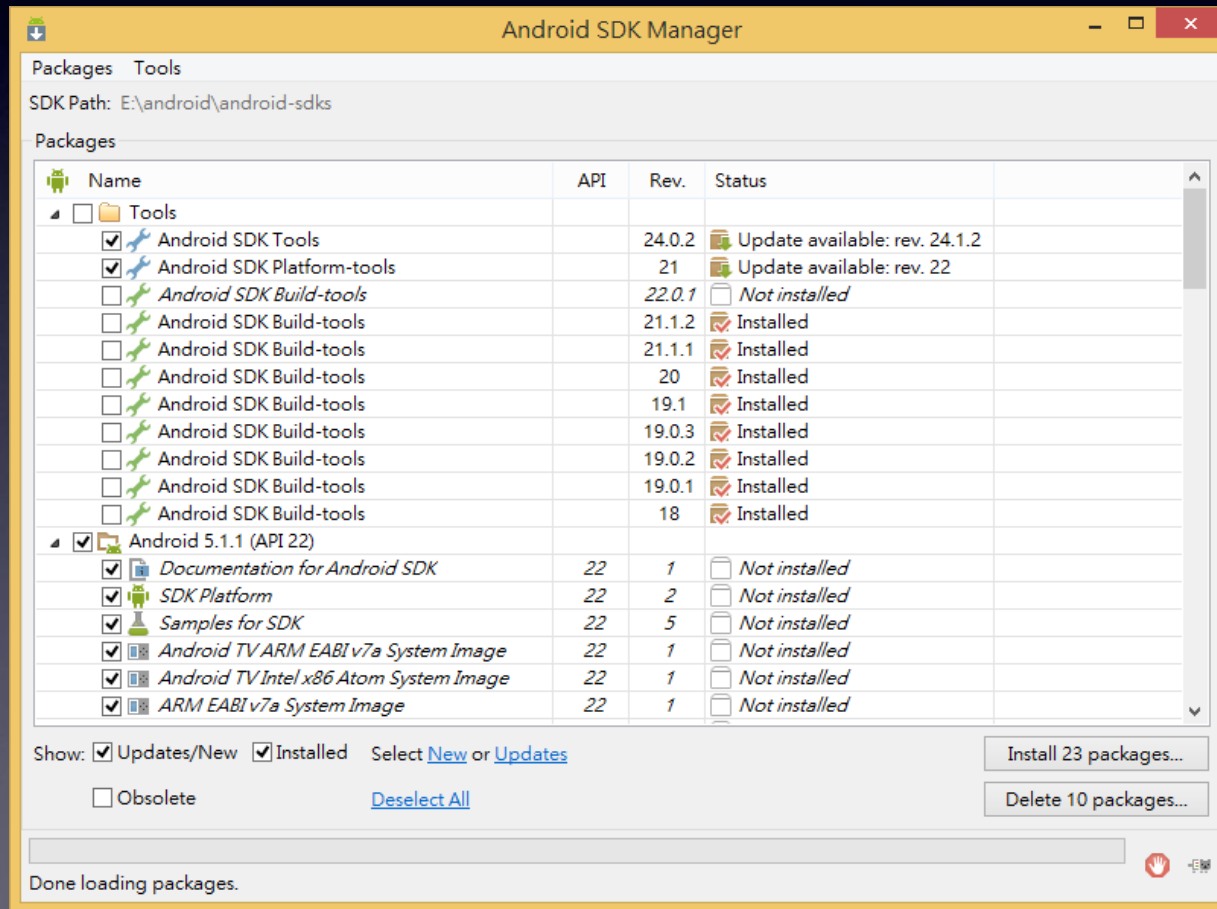
程式  
編輯區

資訊輸出區



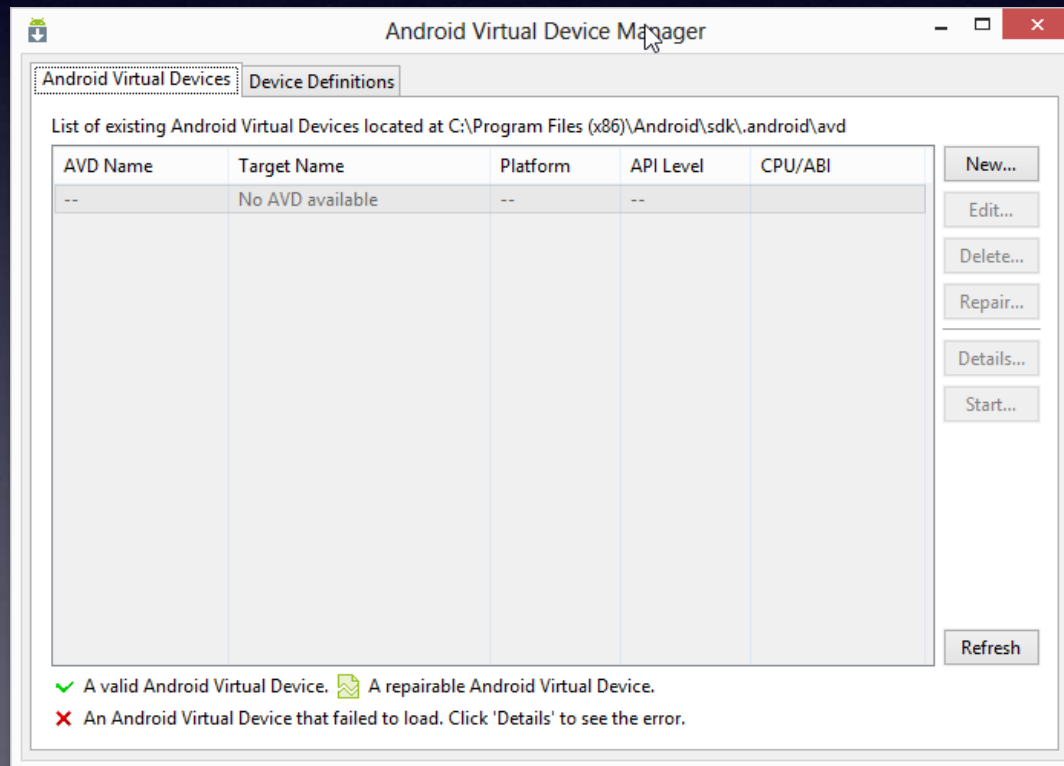
# 打開SDK MANAGER

- 點選Eclipse上方的ADT工具區圖示 
- 可以勾選要下載的SDK版本來進行開發



# 打開AVD MANAGER

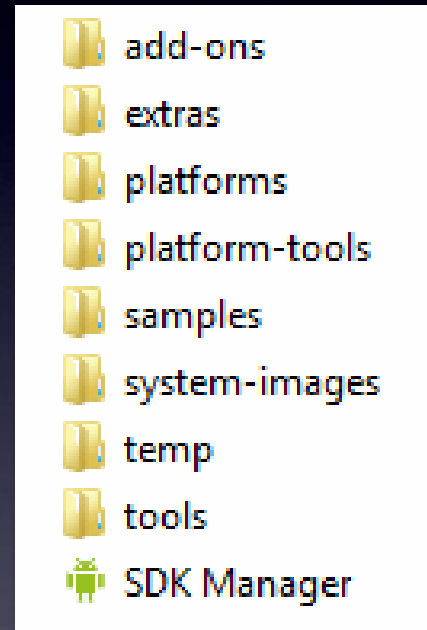
- AVD: Android Virtual Device
- Eclipse上方的ADT工具區圖示





# ANDROID SDK架構

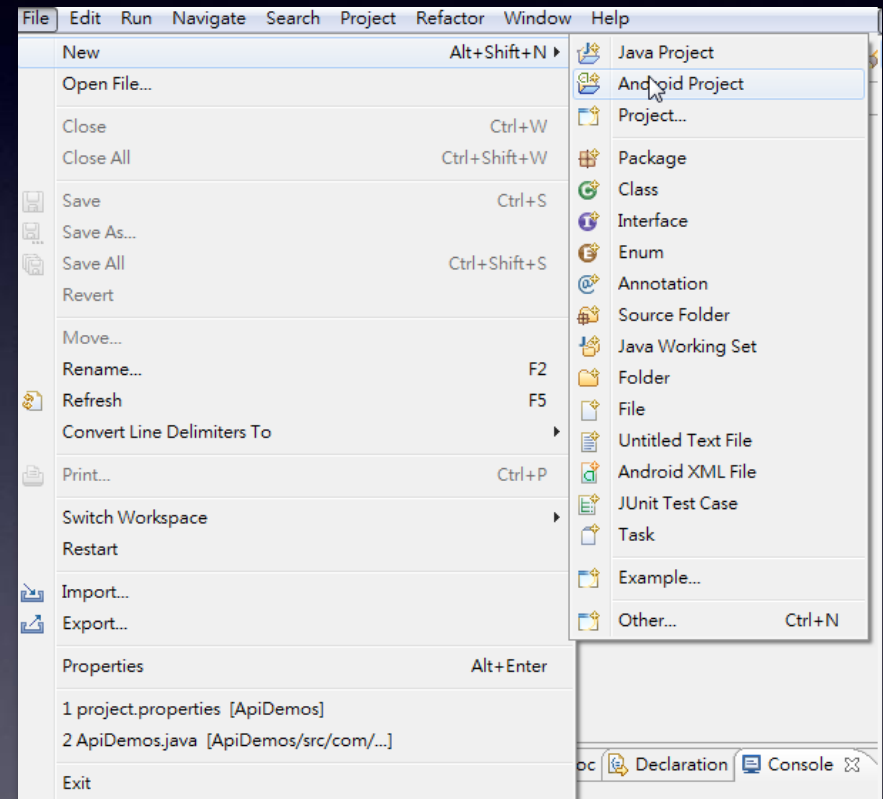
- SDK Manager.exe
  - Android SDK管理器
- tools資料夾
  - Android執行時所需執行檔和函式庫
- samples資料夾
  - 依照不同target而提供的範例程式碼
- platform-tools
  - 編譯android和除錯時所需的執行檔和函式庫
- platforms
  - 不同版本的android模擬器的資料及映像檔



建立新專案

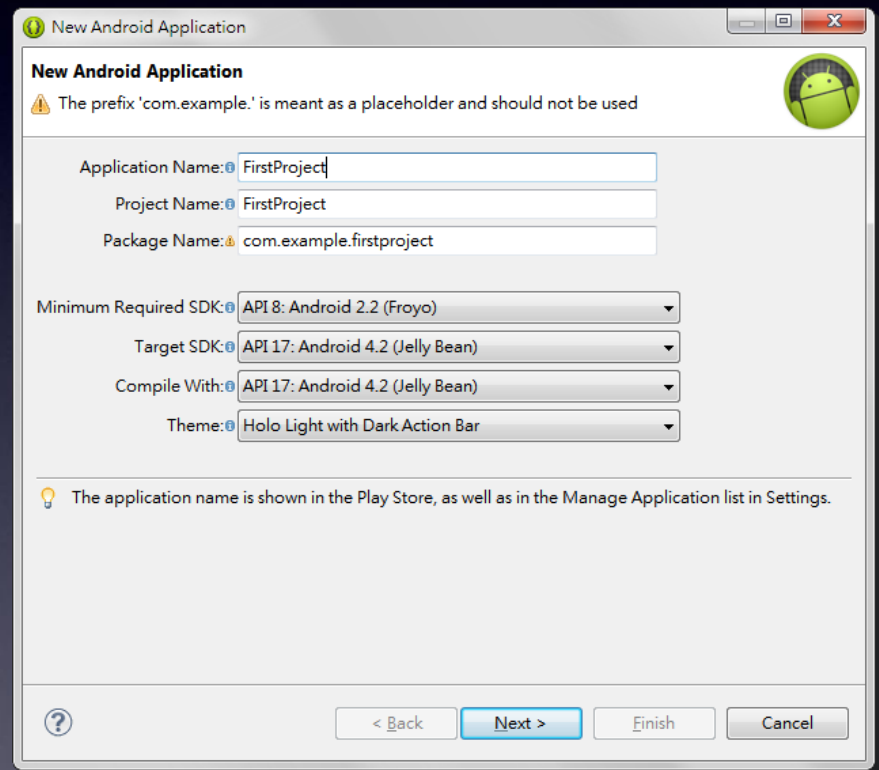
# 建立新專案

- 選擇上方選單  
File→New→Android  
Project



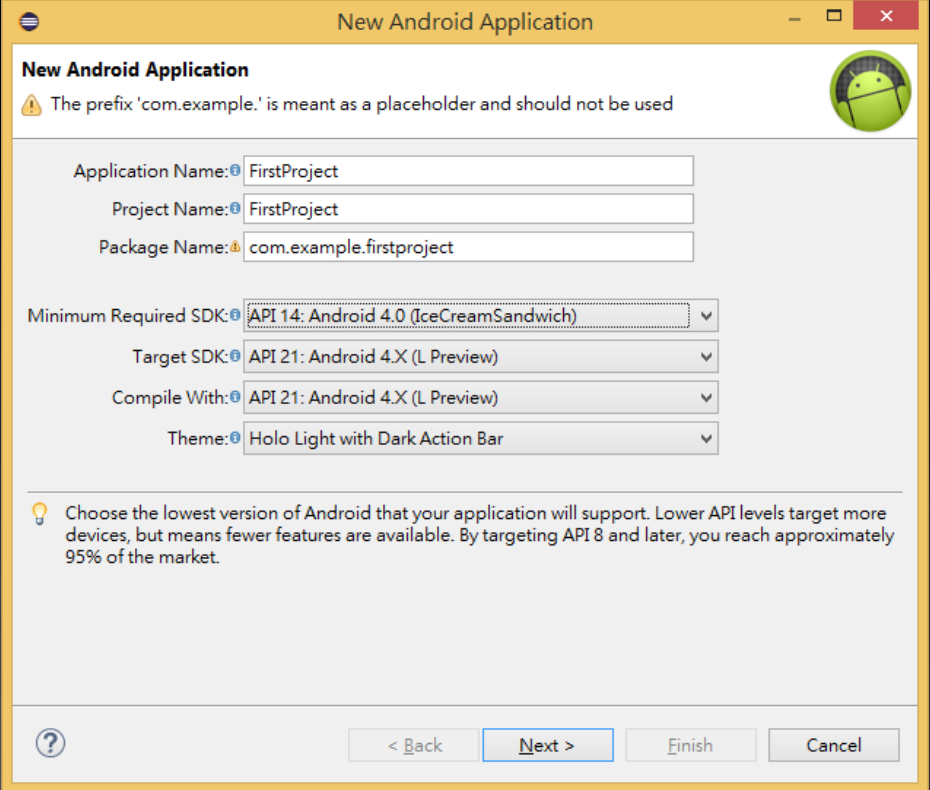
# 建立新專案

- Application Name
  - 安裝後呈現的名稱
- Project Name
  - 在workspace的名稱
- Package Name
  - 要獨立的名稱
- Minimum Require SDK
  - 可執行App的最低Android版本



# 建立新專案

- Target SDK
  - App最重點執行的Android版本
- Compile With
  - 使用哪個版本的SDK來編譯
- Theme
  - 應用程式的佈景主題



New Android Application

The prefix 'com.example.' is meant as a placeholder and should not be used

Application Name: FirstProject

Project Name: FirstProject

Package Name: com.example.firstproject

Minimum Required SDK: API 14: Android 4.0 (IceCreamSandwich)

Target SDK: API 21: Android 4.X (L Preview)

Compile With: API 21: Android 4.X (L Preview)

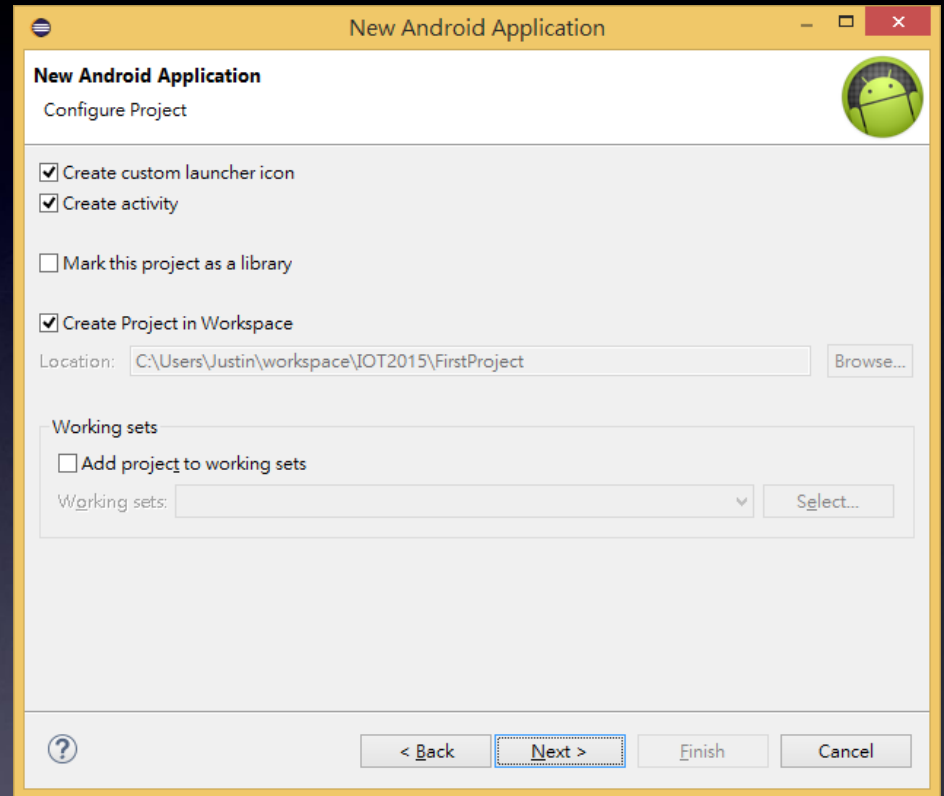
Theme: Holo Light with Dark Action Bar

Choose the lowest version of Android that your application will support. Lower API levels target more devices, but means fewer features are available. By targeting API 8 and later, you reach approximately 95% of the market.

< Back Next > Finish Cancel

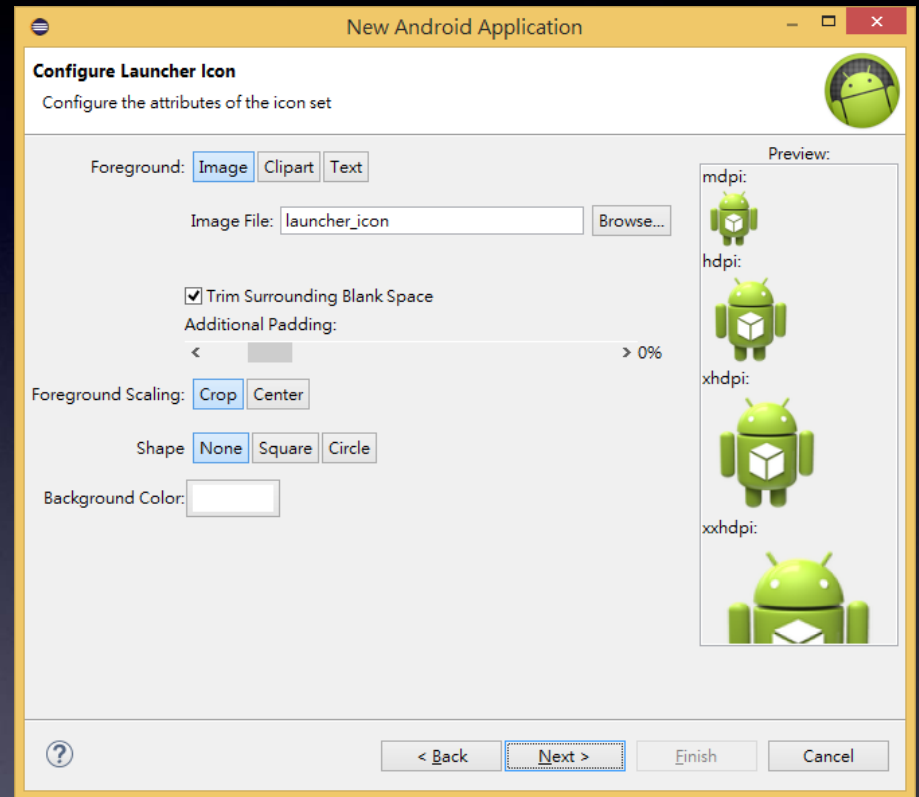
# 建立新專案

- Create custom launcher icon
  - 可以藉由工具建立app的圖示
- Create activity
  - 建立最基本的畫面
- Create Project in Workspace
  - 建立的專案是否擺放在workspace中



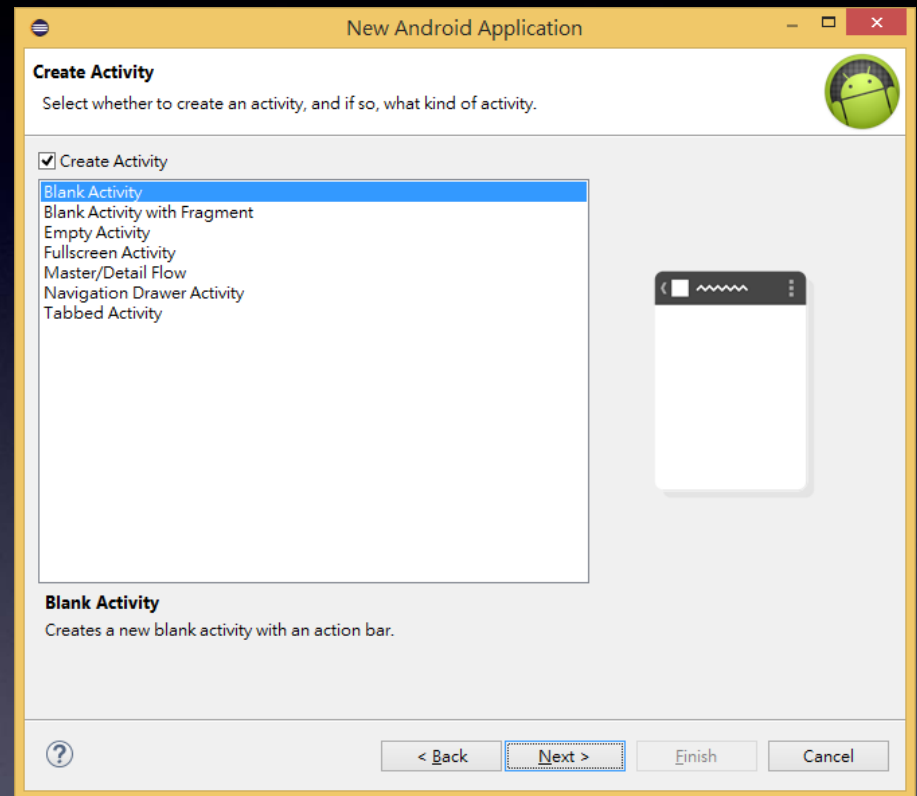
# 建立新專案

- Create Custom launcher icon
  - 建立客製化的app圖示



# 建立新專案

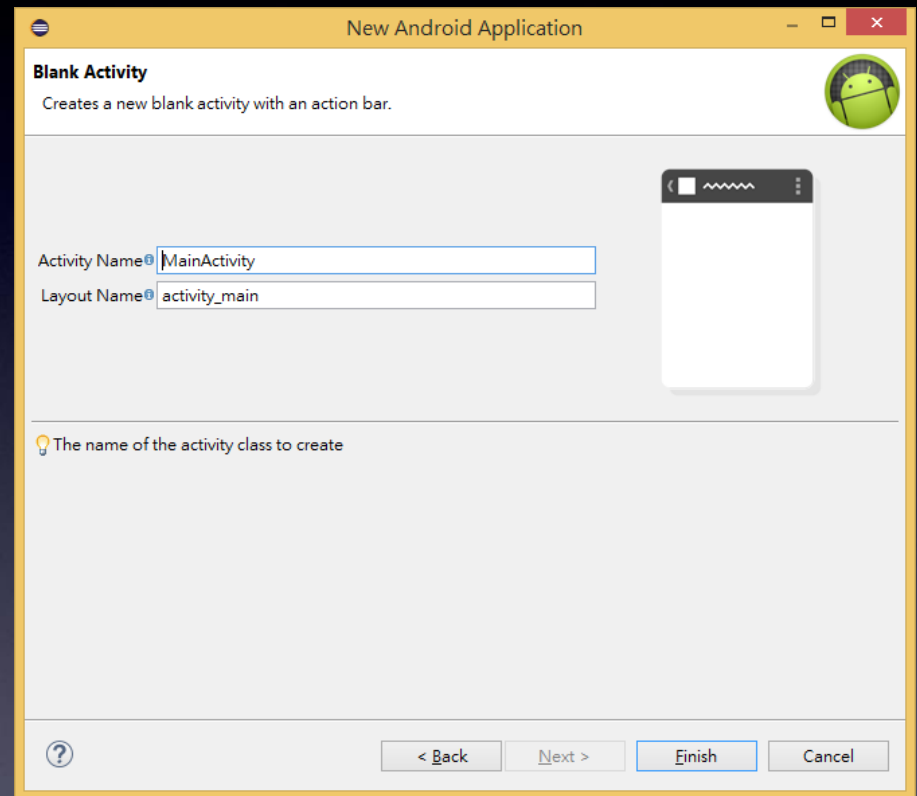
- Create Activity
  - 建立基本畫面的類型
  - 選擇Blank Activity即可



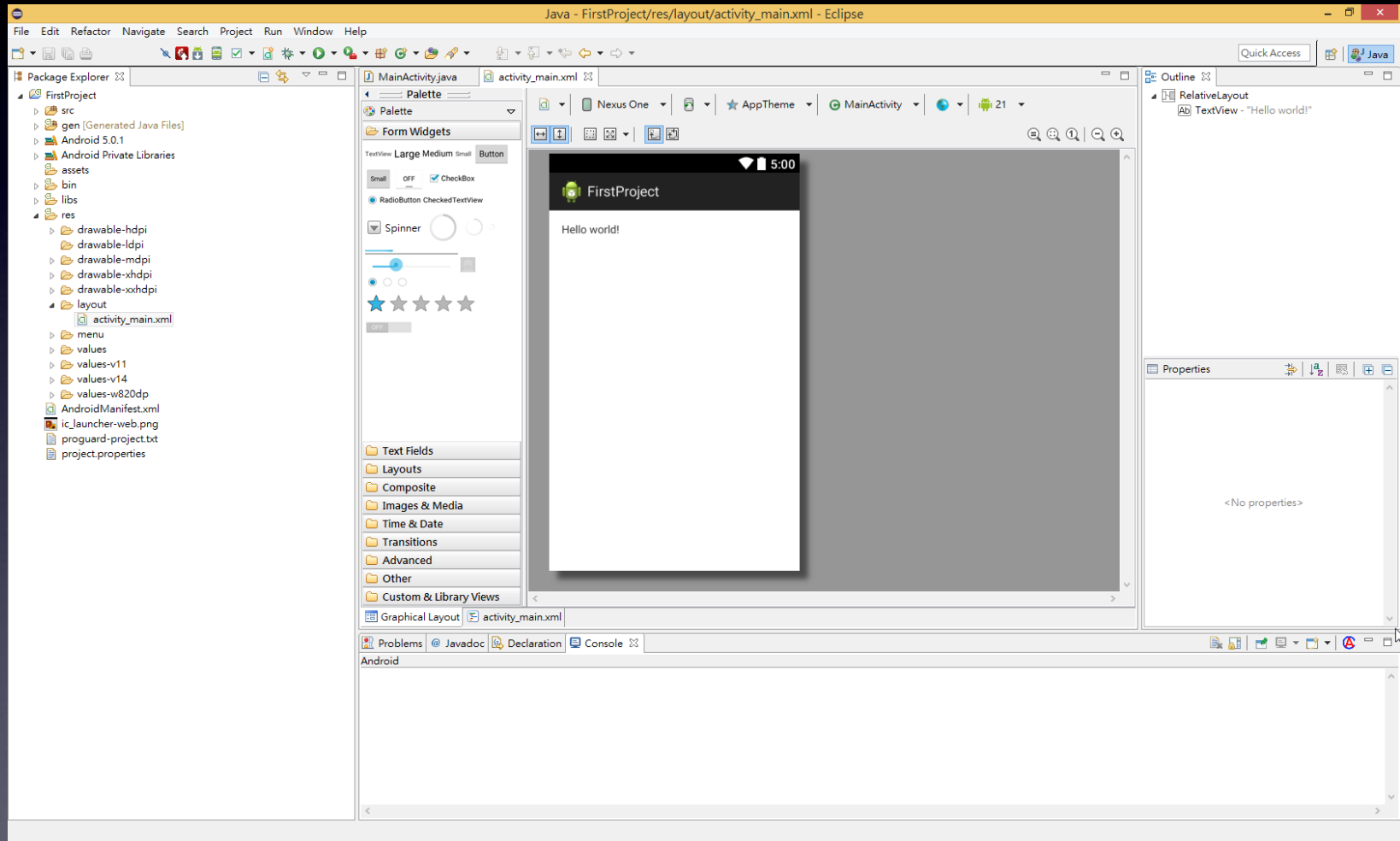


# 建立新專案

- Activity Name
  - 畫面會產生一個Java檔，檔案的名稱
- Layout Name
  - 介面定義檔的名稱
  - Java檔要呈現的畫面會以XML定義



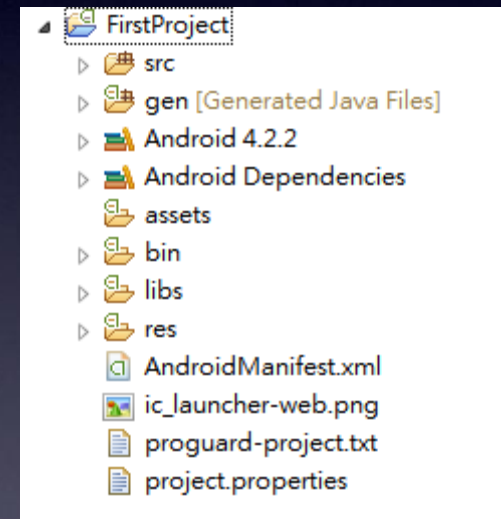
# 建立完成



# 專案架構與執行方式

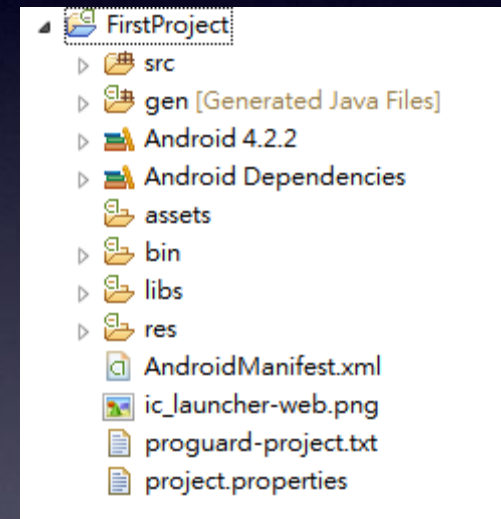
# 專案架構

- src
  - 專案原始碼的擺放位置
- gen
  - 編譯後自動產生  
Android需要用的程式  
碼的擺放處
- Android x.x.x
  - Android SDK函式庫



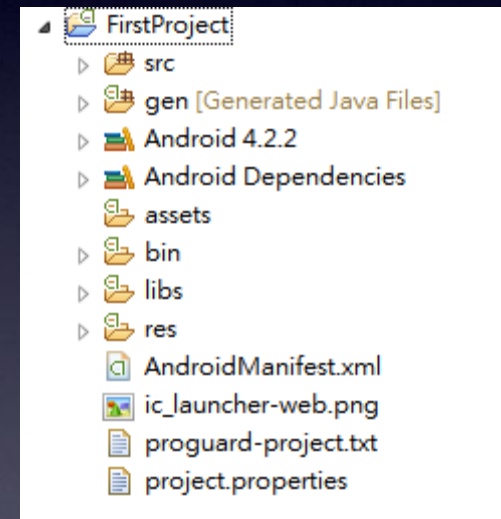
# 專案架構

- Android Dependencies
  - 編譯Android需要用的其他的官方library
- assets
  - 專案額外用的資源檔
- bin
  - 編譯後產生目的檔和APK的地方



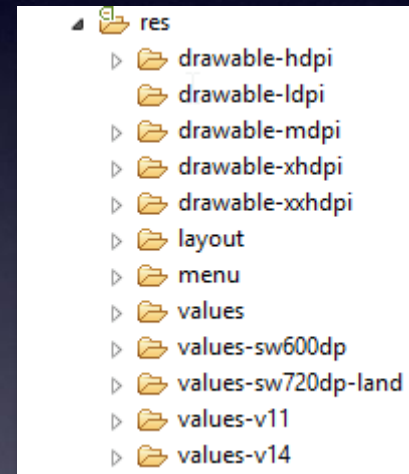
# 專案架構

- **libs**
  - 編譯專案自己客製的library
  - 之後會常常使用，要有印象
- **res**
  - 資源檔擺放處
- **AndroidManifest.xml**
  - Android app屬性定義檔



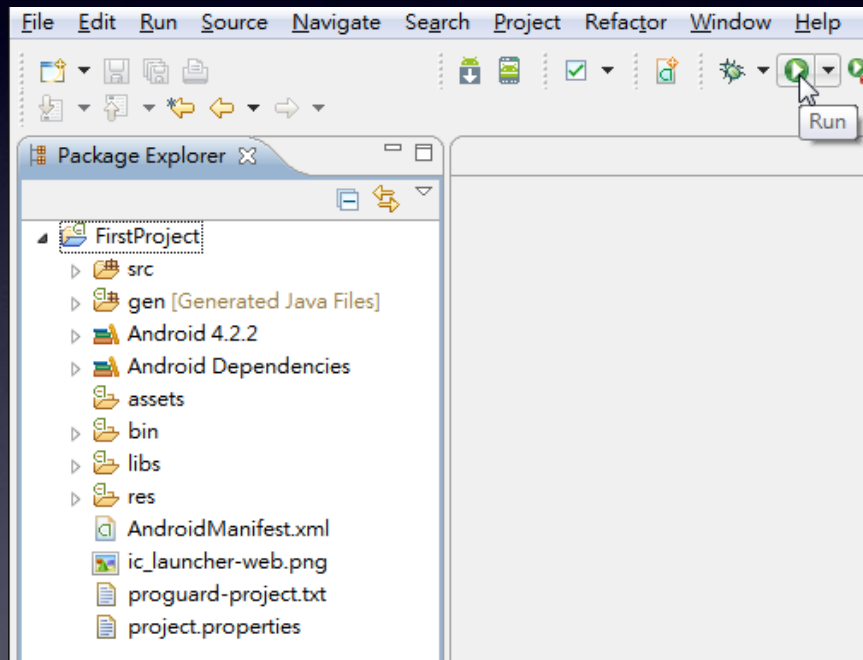
# 專案架構

- Res資料夾
  - drawable
    - 圖檔
  - layout
    - 介面定義檔
  - menu
    - 選單定義檔
  - values
    - 字串
    - 程式參數定義檔
    - 佈景主題



# 執行專案

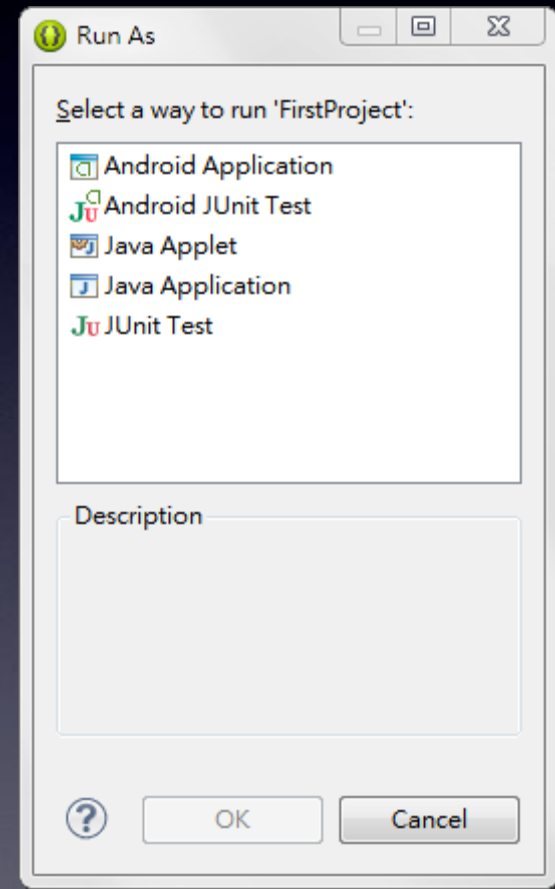
- 選擇要執行的專案
- 按下右圖中右上方的按鈕 





# 執行專案

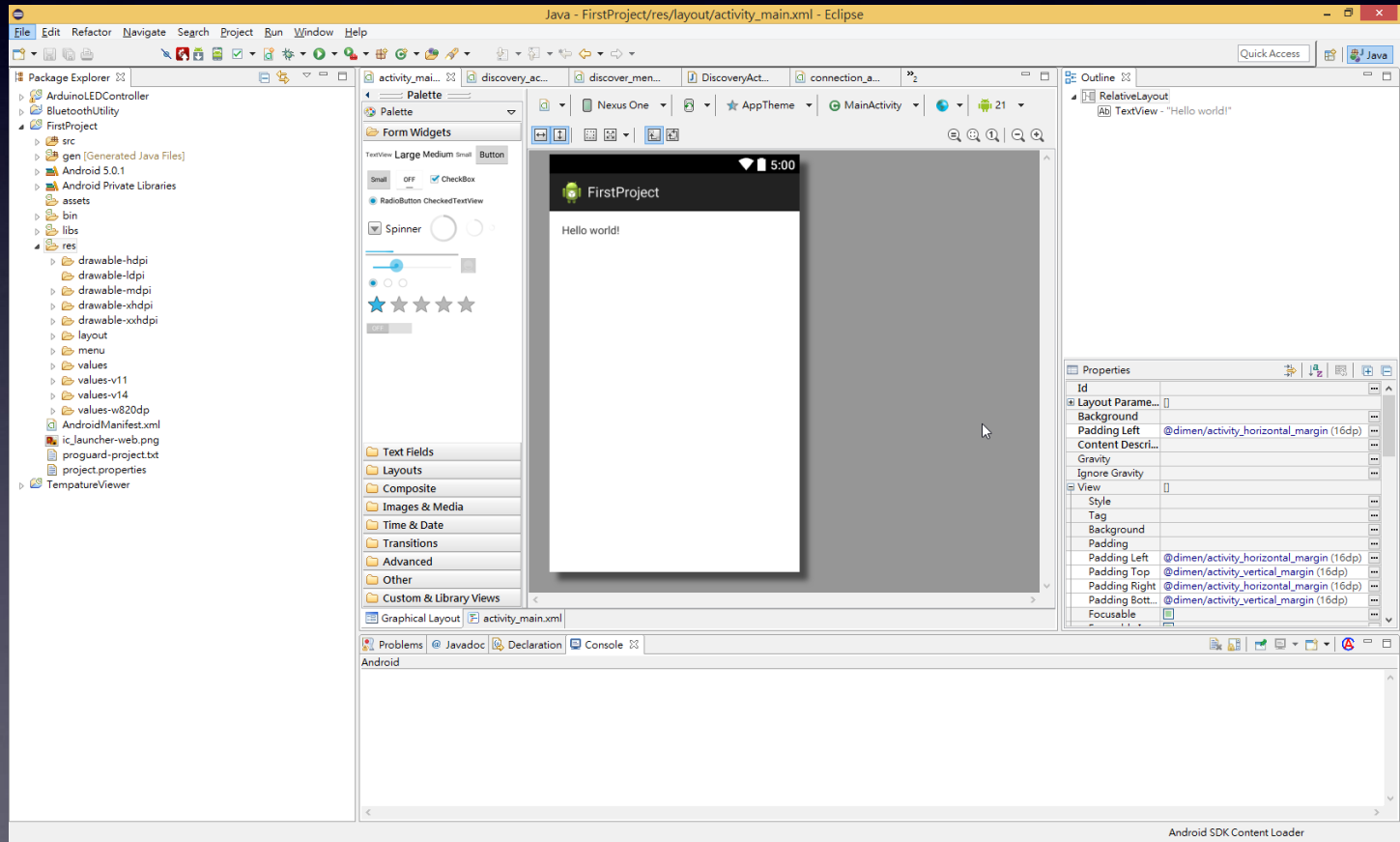
- 選擇Android Application
- OK



介面(LAYOUT)定義檔

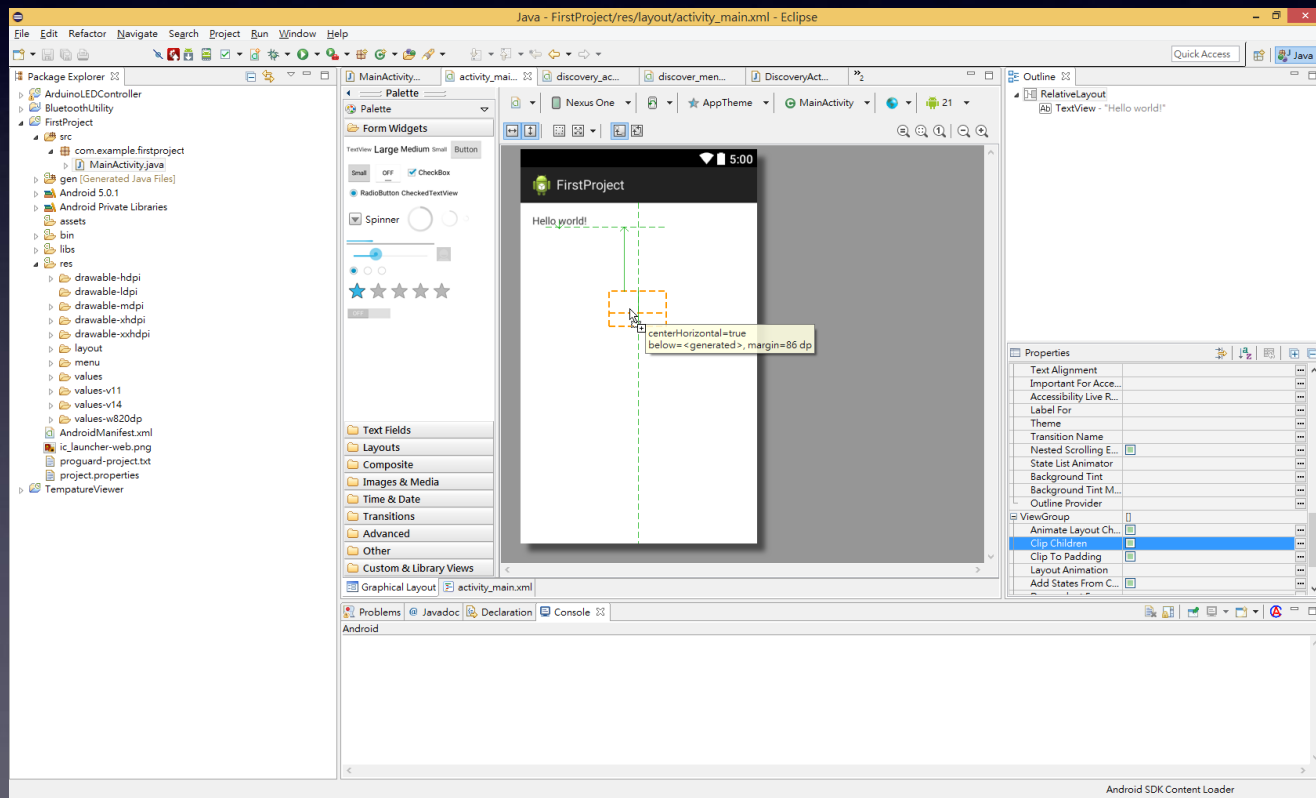
# 介面定義檔

- 打開專案資料夾res/layout中的activity\_main.xml
- 下圖為Android Layout編輯器



# 介面定義檔

- 使用滑鼠將按鈕從左方拖進畫面  
讓畫面增加一個按鈕



# 常用基本元件

- 常用基本元件屬性
  - **id**
    - 給每個建立的元件一個索引值，@+id/<索引值名稱>
  - **Layout Parameters**
    - **width, height**
      - 元件長寬
  - **View**
    - **Visibility**
      - 元件是否可視

呈現畫面

# 呈現畫面

頁面要繼承Activity

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

# 呈現畫面

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

onCreate是啟動Activity  
在這個方法中新增畫面



# 呈現畫面

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

先呼叫父類別的onCreate  
讓父類別作初始化

# 呈現畫面

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

設定本Activity  
呈現的畫面

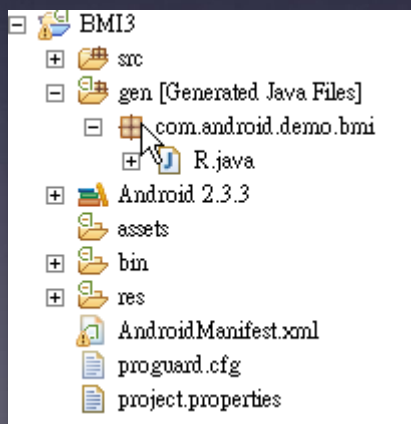
# 呈現畫面

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

設定layout給activity  
R.layout是什麼？

# 呈現畫面

- 在Android中，會自動將資源檔以它的類型轉為class，再以索引值轉變為特殊的變數，每個變數都可以讓我們存取到該資源
- 這個Android自動產生的檔案稱為R.java，它存在於/gen/<應用程式的package>/R.java



# 程式讓畫面顯示

- 我們在程式中要存取資源檔，就是使用 **R.<資源類型>.<索引值>**
- 例如
  - R.layout.main
  - R.string.hello
  - R.drawable.launcher
  - R.id.btn\_ok
- setContentView()會依照傳入的參數來將整個XML的介面元件轉為Java的物件

那我要切換應用程式畫面時  
使用setContentView()設定其他頁面就可以了嗎??

- **千萬不要!!**
- 一個Activity只呼叫一次setContentView()
- 要換頁面就使用Activity切換

# 點擊(CLICK)事件

# 點擊(CLICK)事件

- Click就是使用者用手指頭去點的動作
- 撰寫程式時必須得撰寫View接收到事件後的行為  
要達到這個目的，包含以下步驟
  1. 在程式中找出該按鈕(使用findViewById)
  2. 針對該按鈕，設定事件接受器OnClickListener
- 來看看程式怎麼寫



# 按鈕事件

```
public class MainActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Button button = (Button) findViewById(R.id.button1);  
        button.setOnClickListener(new OnClickListener() {  
  
            @Override  
            public void onClick(View v) {  
  
            }  
        });  
    }  
}
```

findViewById找出View

# 按鈕事件

```
public class MainActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Button button = (Button) findViewById(R.id.button1);  
        button.setOnClickListener(new OnClickListener() {  
  
            @Override  
            public void onClick(View v) {  
  
            }  
        });  
    }  
}
```

轉型成特定的View  
在此是轉型成Button

# 按鈕事件

```
public class MainActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Button button = (Button) findViewById(R.id.button1);  
        button.setOnClickListener(new OnClickListener() {  
  
            @Override  
            public void onClick(View v) {  
  
            }  
        });  
    }  
}
```

設定事件接收器  
OnClickListener給View

# 按鈕事件

```
public class MainActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Button button = (Button) findViewById(R.id.button1);  
        button.setOnClickListener(new OnClickListener() {  
  
            @Override  
            public void onClick(View v) {  
  
            }  
        });  
    }  
}
```

使用new建立一個  
OnClickListener的實體

# 按鈕事件

```
public class MainActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Button button = (Button) findViewById(R.id.button1);  
        button.setOnClickListener(new OnClickListener() {  
  
            @Override  
            public void onClick(View v) {  
  
            }  
        });  
    }  
}
```

按到按鈕後的程式寫在  
onClick的方法內

# 按鈕事件

```
public class MainActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Button button = (Button) findViewById(R.id.button1);  
        button.setOnClickListener(new OnClickListener() {  
  
            @Override  
            public void onClick(View v) {  
  
            }  
        });  
    }  
}
```

參數傳入的v就是Button  
R.id.button1

# 接收事件

- 其他事件
  - 長按事件

```
new View.OnLongClickListener() {  
    public void onLongClick(View v) {}  
}
```

- 指觸事件

```
new View.OnTouchListener() {  
    public void onTouch(View v, MotionEvent e) {}  
}
```

- 按鍵事件

```
new View.OnKeyListener() {  
    public void onKey(View v, int keyCode, KeyEvent e) {}  
}
```

# 結束ACTIVITY

- 如果我希望讓按鈕按下後目前的畫面(Activity)就關閉，該怎麼做？
  - `finish()`
    - 只要在Activity內呼叫此方法，Activity就會關閉



# 結束ACTIVITY

```
public class MainActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Button button = (Button) findViewById(R.id.button1);  
        button.setOnClickListener(new OnClickListener() {  
  
            @Override  
            public void onClick(View v) {  
                finish();  
            }  
        });  
    }  
}
```

呼叫finish結束Activity

畫面提示

# 畫面提示

- Toast是短暫性跳出通知使用者的一種方式
  - 只能顯示資訊，使用者無法互動
- 常用在通知APP狀態
  - 登入失敗、資料未填

# 畫面提示

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button button = (Button) findViewById(R.id.button1);
        button.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                Toast.makeText(MainActivity.this, "Click!!",
                    Toast.LENGTH_LONG).show();
            }
        });
    }
}
```

# 畫面提示

```
public class MainActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Button button = (Button) findViewById(R.id.button1);  
        button.setOnClickListener(new OnClickListener() {  
            @Override  
            public void onClick(View v) {  
                Toast.makeText(MainActivity.this, "Click!!",  
                    Toast.LENGTH_LONG).show();  
            }  
        });  
    }  
}
```

Toast.makeText來建立  
畫面提示

# 畫面提示

```
public class MainActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Button button = (Button) findViewById(R.id.button1);  
        button.setOnClickListener(new OnClickListener() {  
  
            @Override  
            public void onClick(View v) {  
                Toast.makeText(MainActivity.this, "Click!!",  
                    Toast.LENGTH_LONG).show();  
            }  
        });  
    }  
}
```

參數1傳入Activity

# 畫面提示

```
public class MainActivity extends Activity {
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout.activity_main);
```

```
    Button button = (Button) findViewById(R.id.button1);
```

```
    button.setOnClickListener(new OnClickListener() {
```

參數2傳入要顯示的文字

```
        @Override
```

```
        public void onClick(View v) {
```

```
            Toast.makeText(MainActivity.this, "Click!!",
```

```
                Toast.LENGTH_LONG).show();
```

```
        }
```

```
    });
```

```
}
```

# 畫面提示

```
public class MainActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Button button = (Button) findViewById(R.id.button1);  
        button.setOnClickListener(new OnClickListener() {  
  
            @Override  
            public void onClick(View v) {  
                Toast.makeText(MainActivity.this, "Click!!",  
                    Toast.LENGTH_LONG).show();  
            }  
        });  
    }  
}
```

參數3決定通知顯示的長短  
LENGTH\_LONG or LENGTH\_SHORT



# 畫面提示

```
public class MainActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Button button = (Button) findViewById(R.id.button1);  
        button.setOnClickListener(new OnClickListener() {  
  
            @Override  
            public void onClick(View v) {  
                Toast.makeText(MainActivity.this, "Click!!",  
                    Toast.LENGTH_LONG).show();  
            }  
        });  
    }  
}
```

最後一定要記得呼叫show  
才會顯示在畫面上

# 作業

- 寫一個登入畫面，如右圖  
有兩個EditText和兩個Button
- EditText要預設有提示字  
告訴使用者輸入帳號和輸入密碼
- 按下登入按鈕
  - 帳號欄和密碼欄都有輸入文字  
就顯示「登入成功」的Toast
- 按下取消按鈕
  - 關閉這個頁面



應用程式配置設定檔

# 應用程式配置設定檔

- 前面有說到每個App都可包含多個載體
- 而載體在App安裝到裝置時，必須得向Android Framework註冊
- 註冊的內容就是寫在應用程式配置設定檔 **AndroidManifest.xml** 中
- 打開FirstProject的AndroidMainfest.xml

# 應用程式配置設定檔

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firstproject"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="17"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

應用程式的package

# 應用程式配置設定檔

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firstproject"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="17" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

應用程式開發的版號  
versionCode是給開發者看的版本  
versionName是給使用者看的版號

# 應用程式配置設定檔

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firstapp"
    android:versionCode="1"
    android:versionName="1.0">
```

App可安裝的Android版本  
minSdkVersion 最低支援版本  
targetSdkVersion App最佳支援版本

```
<uses-sdk android:minSdkVersion="14" android:targetSdkVersion="17"/>
```

```
<application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name" >
    <activity
        android:name=".MainActivity"
        android:label="@string/app_name" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
</manifest>
```

# 應用程式配置設定檔

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firstproject"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="17" android:targetSdkVersion="17"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

表示開始定義應用程式的設定



# 應用程式配置設定檔

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firstproject"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="17"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

icon App的ICON  
label App呈現在裝置上的名稱

# 應用程式配置設定檔

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firstproject"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="17"/>

    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name"
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
                <intent-filter>
                    <action android:name="android.intent.action.MAIN" />

                    <category android:name="android.intent.category.LAUNCHER" />
                </intent-filter>
            </activity>
        </application>
    </manifest>
```

表示開始定義頁面的設定  
App有幾個Activity，就有幾個定義

# 應用程式配置設定檔

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firstproject"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="17"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
                <intent-filter>
                    <action android:name="android.intent.action.MAIN" />

                    <category android:name="android.intent.category.LAUNCHER" />
                </intent-filter>
            </activity>
        </application>
    </manifest>
```

activity的定義一定要寫在  
<application>內

# 應用程式配置設定檔

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firstproject"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="17"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

name 頁面所使用的class

# 應用程式配置設定檔

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firstproject"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="17"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Android會自動將此名稱與最上方宣告的package結合

# 應用程式配置設定檔

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firstproject"
    android:versionCode="1"
    android:versionName="1.0" >
```

就是這個!!

```
<uses-sdk android:minSdkVersion="14" android:targetSdkVersion="17"/>

<application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name" >
    <activity
        android:name=".MainActivity"
        android:label="@string/app_name" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
</manifest>
```

# 應用程式配置設定檔

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firstproject"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="17"/>

    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

所以這邊指的就是  
com.example.firstproject.MainActivity



# 應用程式配置設定檔

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firstproject"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="17"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

label  
會顯示在頁面最上方的標題



# 應用程式配置設定檔

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firstproject"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="17"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

安裝App時，每個頁面都要向  
Android的Framework註冊

# 應用程式配置設定檔

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firstproject"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="17"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

intent-filter就是註冊的條件

# 應用程式配置設定檔

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firstproject"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="17"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

因為每個頁面可以啟動的條件不一樣  
例如：有些頁面專門用來當作App的啟動頁

# 應用程式配置設定檔

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firstproject"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="17"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

一個Activity可以有多個註冊的條件

# 應用程式配置設定檔

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firstproject"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="17"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

以下的action和category的組合就是  
搭配出此頁為應用程式的啟動頁

# 應用程式配置設定檔

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.firstproject"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="17"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

一個「可以啟動」的App至少  
要有一個Activity是註冊成App的啟動頁

# ACTIVITY概觀

# ACTIVITY概觀

- 盡量保持每個Activity只負責一個畫面
- 要轉至其他畫面就使用Activity切換
- 依照是否需要與其它Activity交換資料來區分，Activity可分為兩種類型
  - 獨立的Activity
  - 相依的Activity



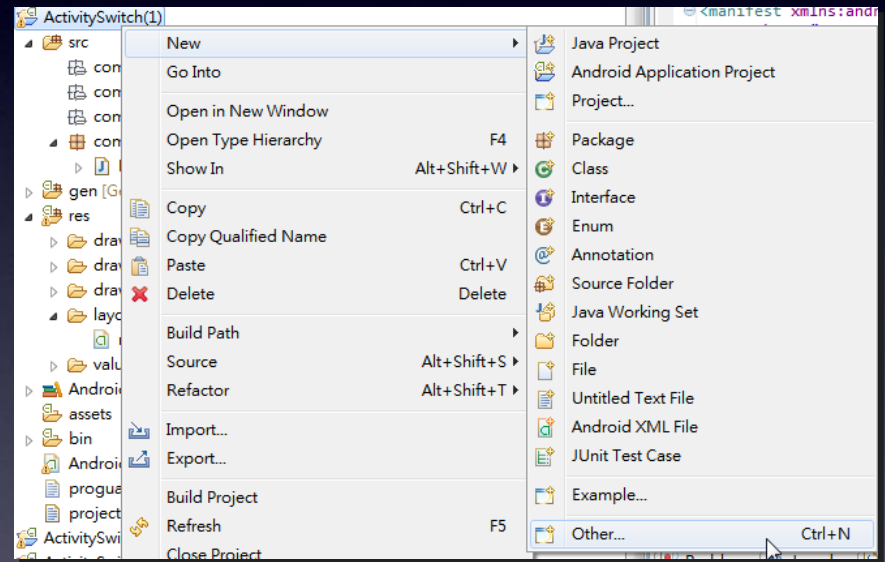
# ACTIVITY概觀

- 獨立的Activity
  - 單純從一個螢幕跳到另一個螢幕，不涉及資料交換
- 相依的Activity
  - 需要與其它的Activity交換資料
  - 分為單向交換與雙向交換
    - 單向：資料由一個螢幕攜帶至另一個螢幕
    - 雙向：螢幕上的資料除了攜帶至另一個螢幕外，還會因為另一個螢幕的操作而改變，進而影響到原本螢幕的資料呈現

新增一個ACTIVITY

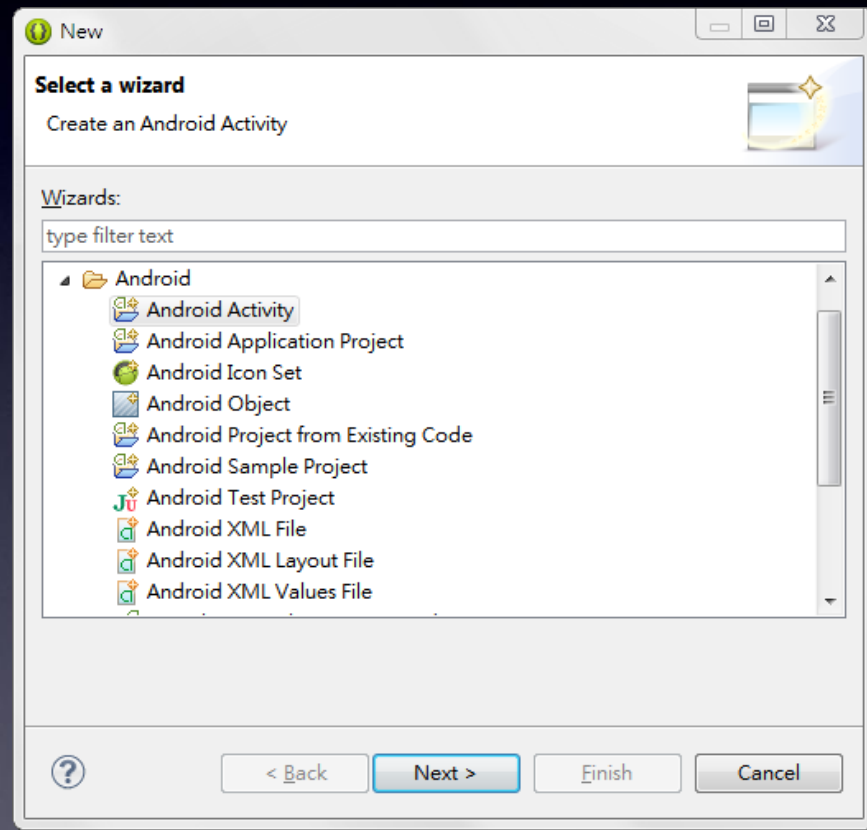
# 新增一個ACTIVITY

- 對專案按下滑鼠右鍵
- New
- Others



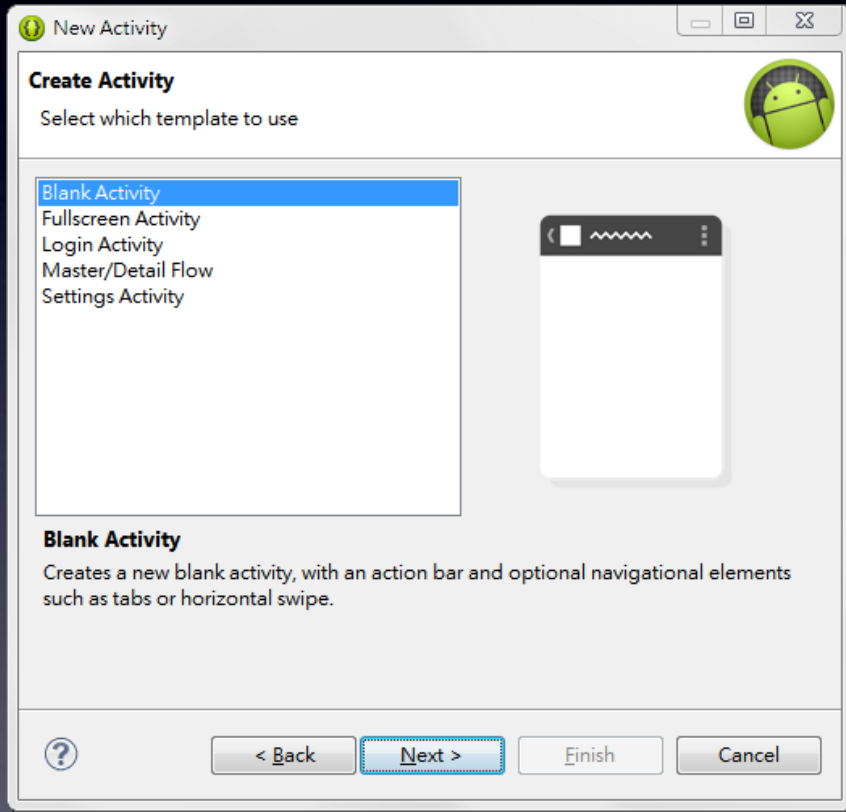
# 新增一個ACTIVITY

- 選擇Android的  
Android Activity
- Next



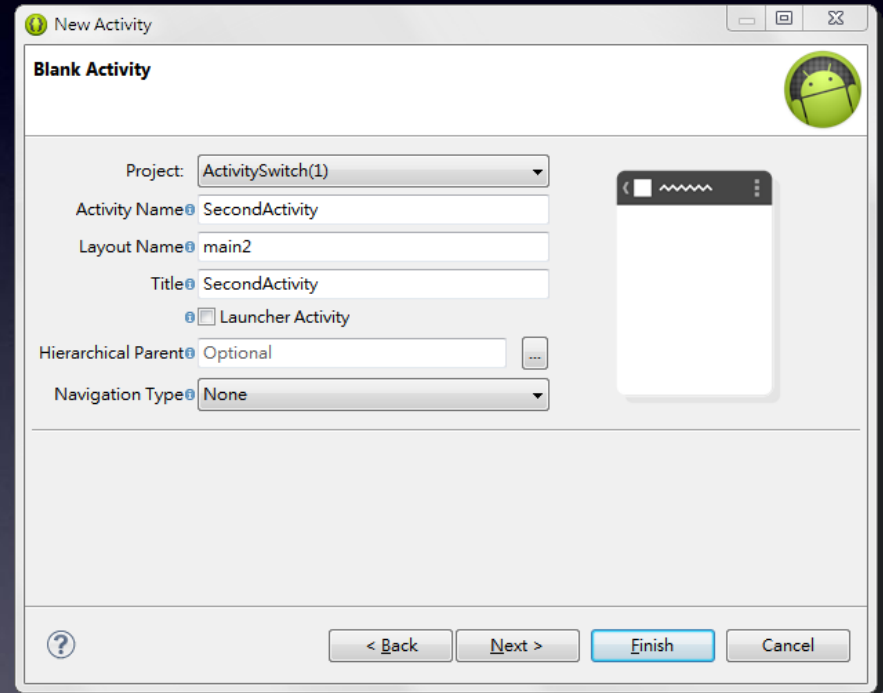
# 新增一個ACTIVITY

- 選擇Blank Activity
- Next



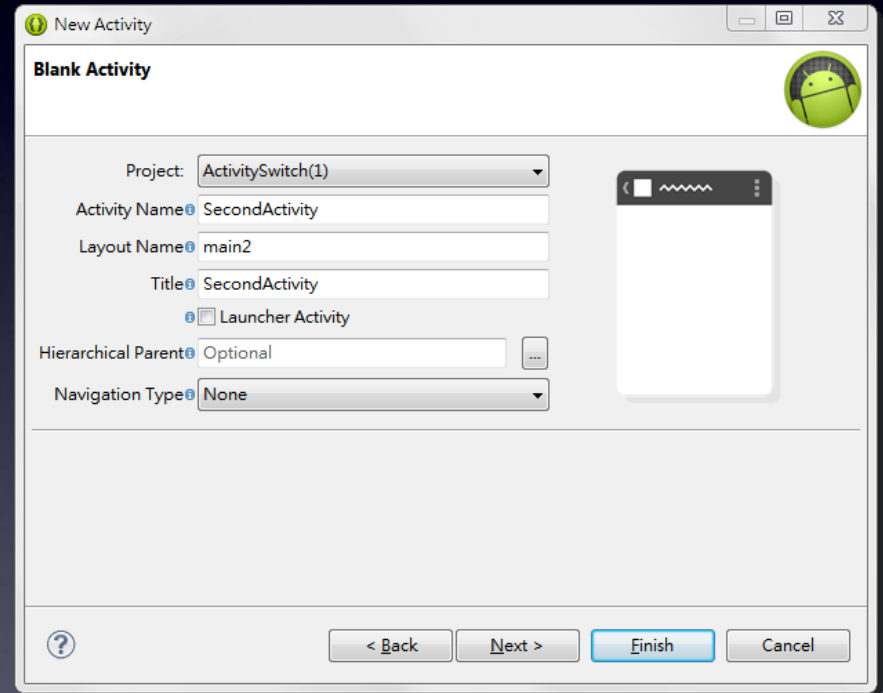
# 新增一個ACTIVITY

- Activity Name
  - 建立Java檔
- Layout Name
  - 介面設定檔的名稱
- Title
  - Activity的名稱
- Launcher Activity
  - 勾選表示這是程式進入的Activity



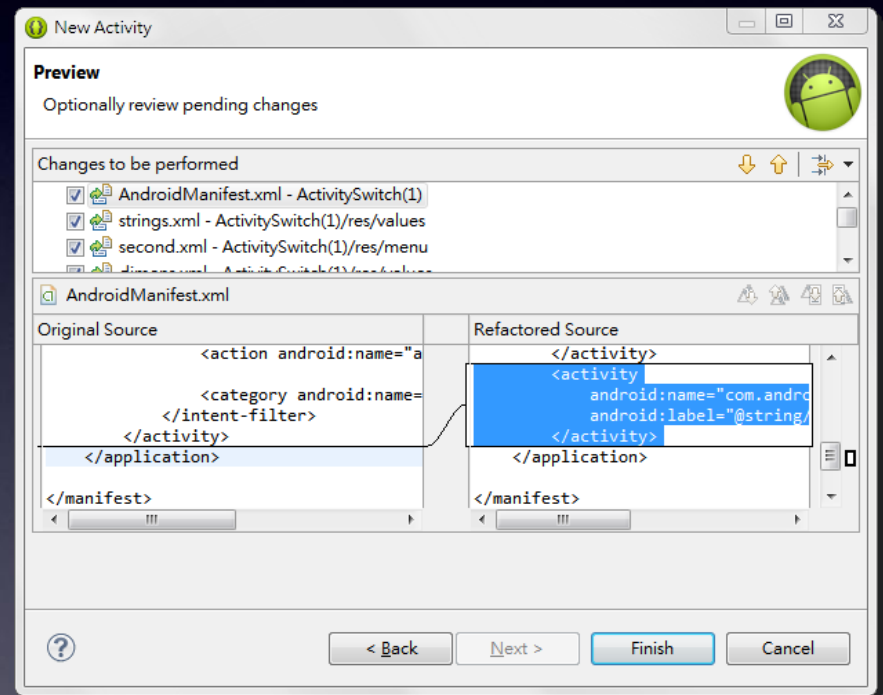
# 新增一個ACTIVITY

- Launcher Activity
  - 勾選表示這是程式進入的Activity
- Next



# 新增一個ACTIVITY

- 接下來ADT會自動將相關程式碼加入AndroidManifest.xml中
- Finish

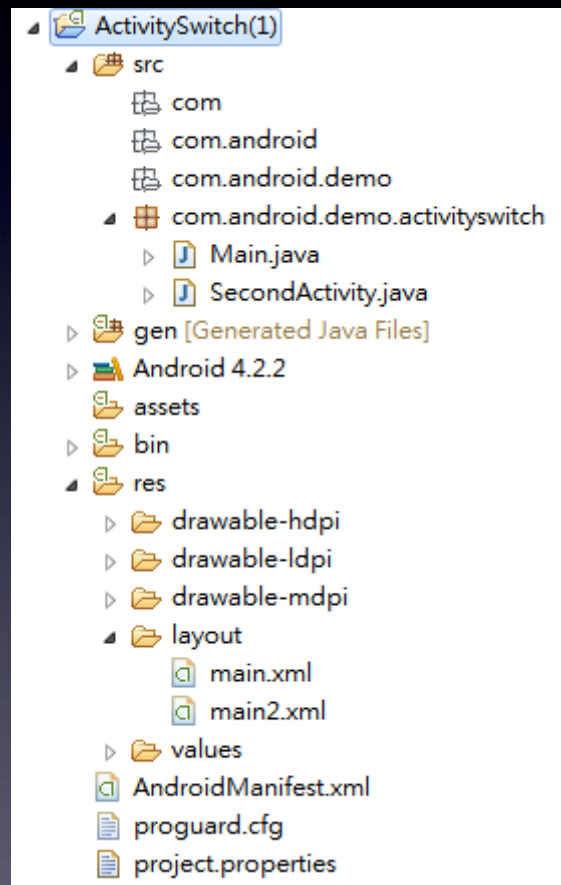




# 新增一個ACTIVITY

- 建立完成
- 確認

AndroidManifest.xml  
有新的Activity定義



androidbasic/ActivitySwitch

切換至另一個ACTIVITY

# 切換至另一個ACTIVITY

- Android之間的頁面切換使用Intent來進行
- 要傳遞的內容也是藉由Intent來傳遞
- Activity切換
  - 使用 `Intent.setClass`
    - `Intent.setClass(目前Activity.this, 目標Activity.class)`
  - 再呼叫`startActivity(intent)`

# 切換至另一個ACTIVITY

- 資料傳遞
  - 送出端
    - `Intent.putExtra(String key, 型態 val)`
      - 支援的傳遞類型
        - Int, boolean, byte, char, double, float, long, String, short
        - 上述型態的陣列
        - Serializable, Parcelable
  - 接收端
    - 要知道傳遞過來的型態為何、索引值為何並呼叫對應的方法取出傳遞的數值
    - `getIntent().getStringExtra(String key)`
    - `getIntent().getIntExtra(String key, int default)`

# 傳送端

```
public class Main extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button = (Button) findViewById(R.id.button1);
        button.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(Main.this, SecondActivity.class);
                intent.putExtra("string_value", "This is Main Activity");
                intent.putExtra("text_size", 25);
                startActivity(intent);
            }
        });
    }
}
```

# 傳送端

```
public class Main extends Activity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
        Button button = (Button) findViewById(R.id.button1);  
        button.setOnClickListener(new OnClickListener() {  
            @Override  
            public void onClick(View v) {  
                Intent intent = new Intent(Main.this, SecondActivity.class);  
                intent.putExtra("string_value", "This is Main Activity");  
                intent.putExtra("text_size", 25);  
                startActivity(intent);  
            }  
        });  
    }  
}
```

建立Intent  
參數為(目前Activity, 目標Activity)

# 傳送端

```
public class Main extends Activity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
        Button button = (Button) findViewById(R.id.button1);  
        button.setOnClickListener(new OnClickListener() {  
            @Override  
            public void onClick(View v) {  
                Intent intent = new Intent(Main.this, SecondActivity.class);  
                intent.putExtra("string_value", "This is Main Activity");  
                intent.putExtra("text_size", 25);  
                startActivity(intent);  
            }  
        });  
    }  
}
```

放置要傳遞的數值，以(索引值,內容)來設定

# 傳送端

```
public class Main extends Activity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
        Button button = (Button) findViewById(R.id.button1);  
        button.setOnClickListener(new OnClickListener() {  
            @Override  
            public void onClick(View v) {  
                Intent intent = new Intent(Main.this, SecondActivity.class);  
                intent.putExtra("string_value", "This is Main Activity");  
                intent.putExtra("text_size", 25);  
                startActivity(intent);  
            }  
        });  
    }  
}
```

使用startActivity，讓Android Framework開始進行畫面轉換的流程



# 接收端

```
public class SecondActivity extends Activity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main2);  
        Intent intent = getIntent();  
        String value = intent.getStringExtra("string_value");  
        int textSize = intent.getIntExtra("text_size", 15);  
        TextView text = (TextView) findViewById(R.id.textView1);  
        text.setText(value);  
        text.setTextSize(textSize);  
    }  
}
```

# 接收端

```
public class SecondActivity extends Activity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main2);  
        Intent intent = getIntent();  
        String value = intent.getStringExtra("string_value");  
        int textSize = intent.getIntExtra("text_size", 15);  
        TextView text = (TextView) findViewById(R.id.textView1);  
        text.setText(value);  
        text.setTextSize(textSize);  
    }  
}
```

取得傳遞到這個Activity的Intent

# 接收端

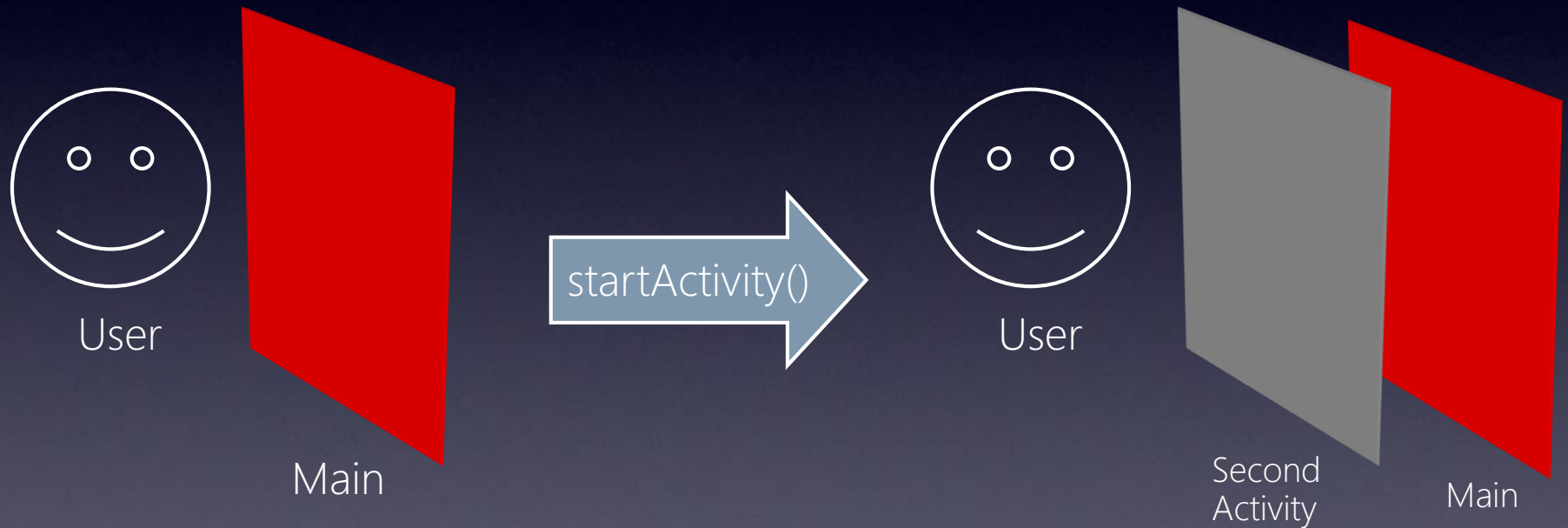
```
public class SecondActivity extends Activity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main2);  
        Intent intent = getIntent();  
        String value = intent.getStringExtra("string_value");  
        int textSize = intent.getIntExtra("text_size", 15);  
        TextView text = (TextView) findViewById(R.id.textView1);  
        text.setText(value);  
        text.setTextSize(textSize);  
    }  
}
```

取得傳遞內容，使用相對應型態的  
**get型態Extra(索引值)**

# ACTIVITY切换的原理

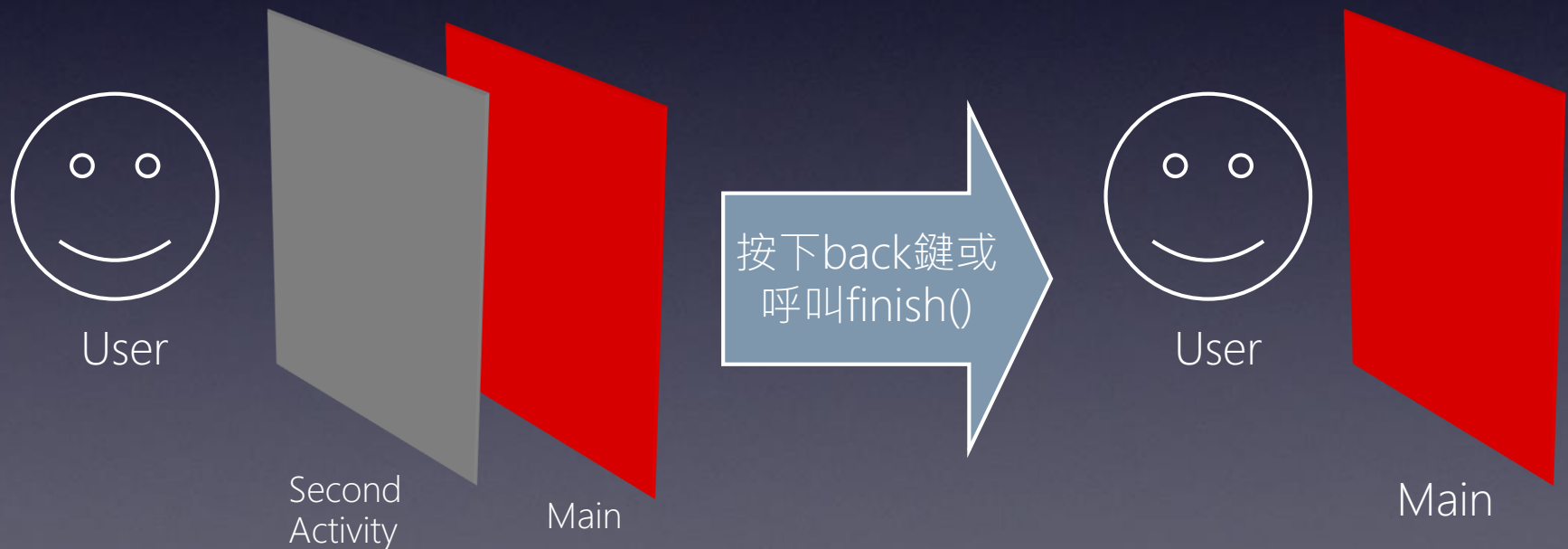
# ACTIVITY切換的原理

- 轉換Activity時，其實就是把新的畫面推到使用者眼前

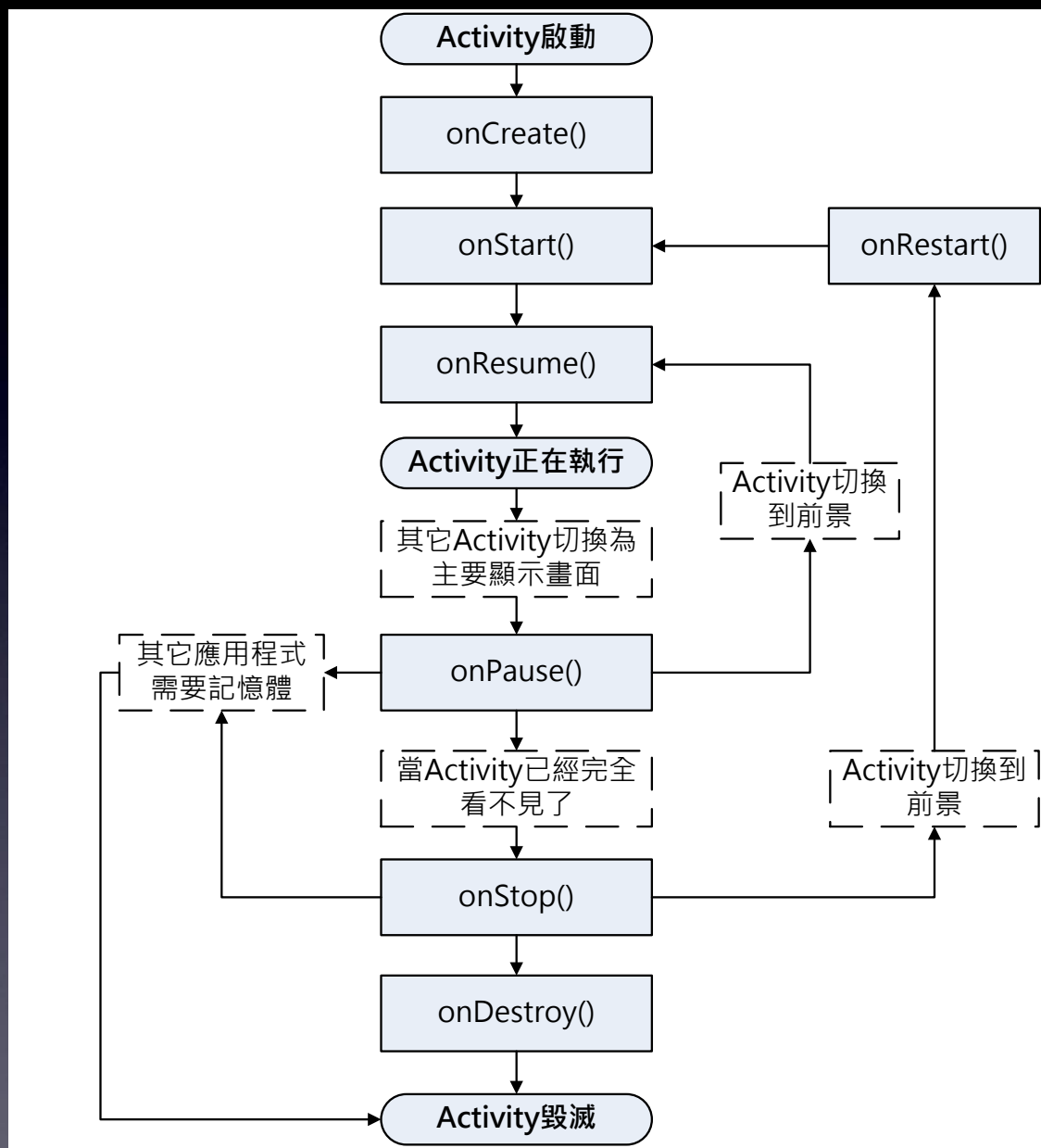


# ACTIVITY切換的原理

- 所以要回到上一個畫面時，千萬不要用startActivity()來啟動，而是要移除現在最前方的Activity
- 當你按下back鍵，或是在第二個Activity中呼叫finish()都可以使Activity結束



# ACTIVITY生命週期



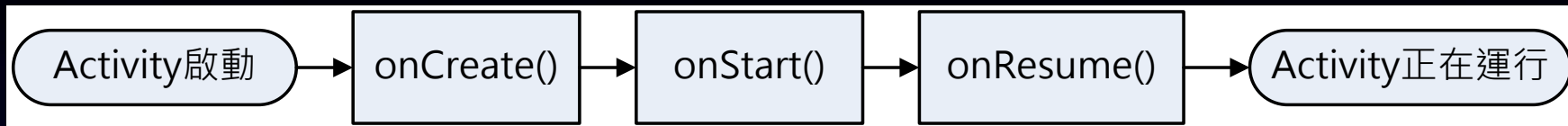


# ACTIVITY生命週期

- Android的Activity有各種不同的狀態需要處理
- **onDestroy()** 表示Activity要結束了
- **onCreate()** 表示Activity要建立了
- **onStop()**表示Activity要完全在畫面上看不見了
- **onStart()**表示Activity要在開始運作了
- **onPause()**表示Activity要離開前景了
- **onResume()**表示Activity表示畫面要顯示在前景了

# 生命週期

- 正常啟動Activity

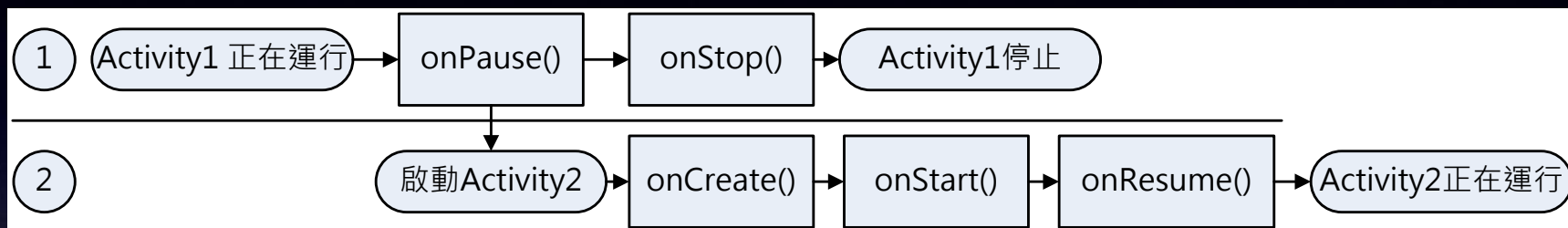


- 正常中止Activity

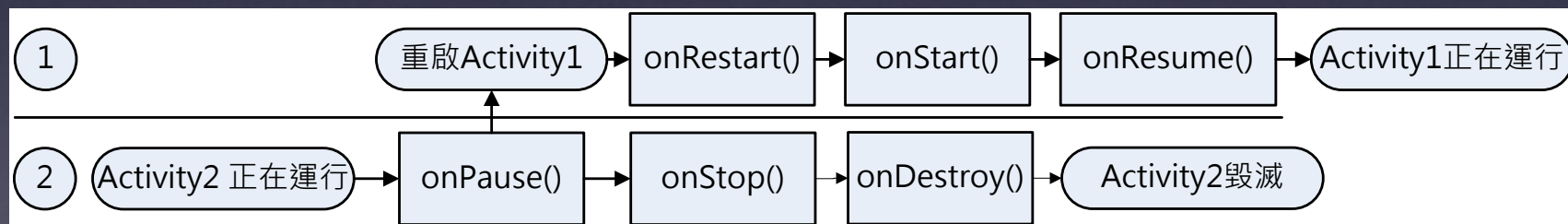


# 生命週期

- 呼叫另一個Activity (由1到2)



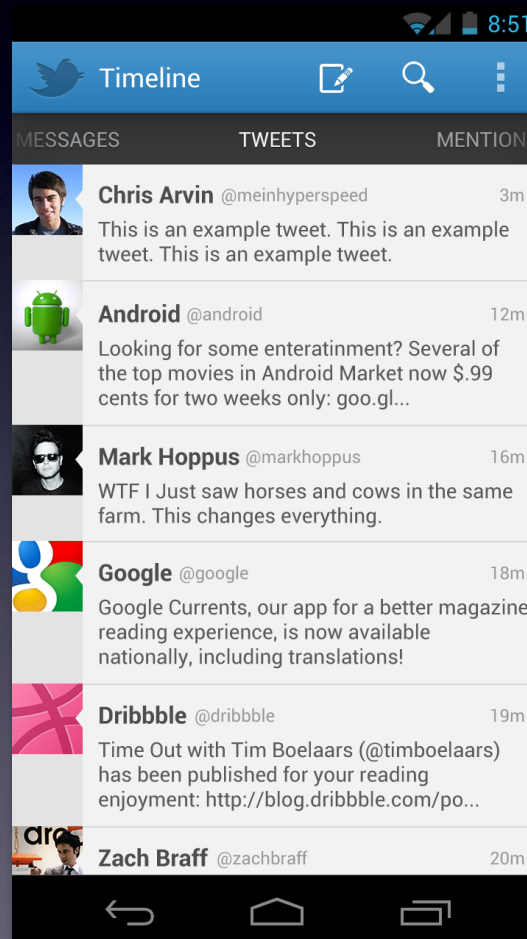
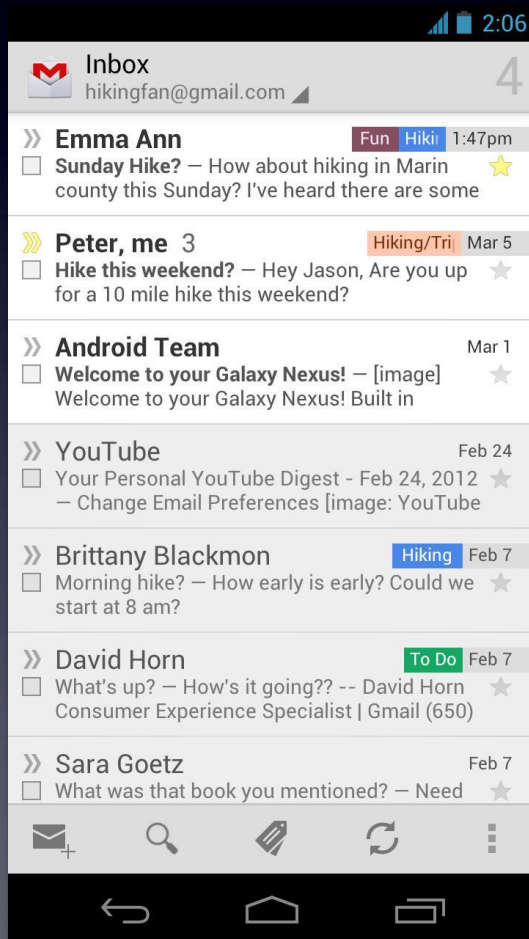
- 按下Back鍵返回原Activity (由2到1)



# LISTVIEW與ADAPTER

# LISTVIEW

- ListView可以說是資料在手機上呈現最好的方法



# LISTVIEW

- 建立ListView的流程
  - 準備一連串資料
  - 準備呈現資料的Layout
  - 準備整合資料和Layout的Adapter (適配器)
  - 將Adapter設定給ListView

# 什麼是ADAPTER

- Adapter就是你的資料、要呈現的Layout與ListView之間的溝通橋樑
- Adapter提供給ListView呈現的畫面與資料的資訊
- Adapter整合資料到Layout上
- Adapter是資料與畫面的結合處
- Adapter是一種Design Pattern
  - [http://en.wikipedia.org/wiki/Adapter\\_pattern](http://en.wikipedia.org/wiki/Adapter_pattern)

# 什麼是ADAPTER

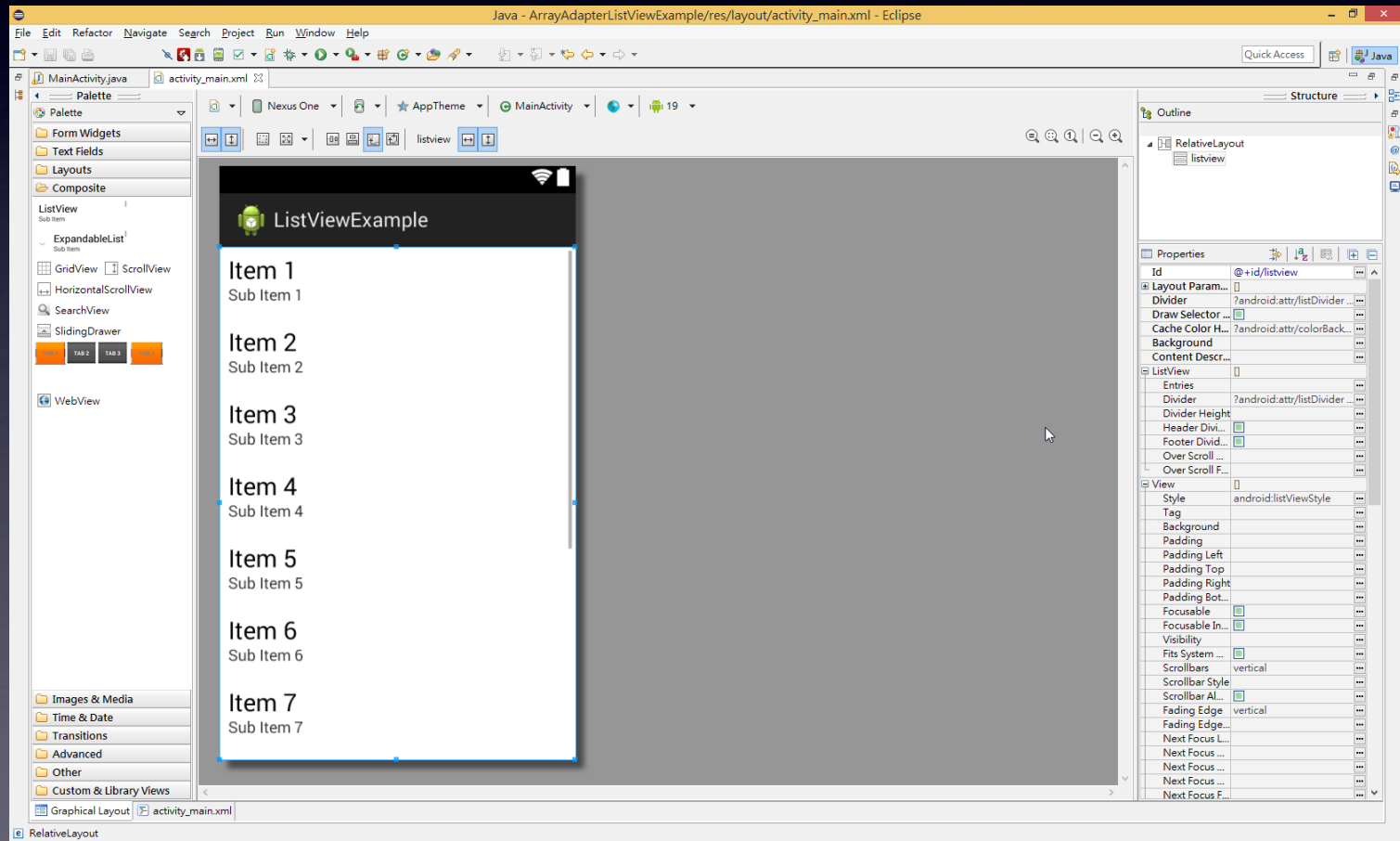
- 為了方便大家製作，Android提供了簡便的Adapter
  - ArrayAdapter
  - SimpleAdapter



# ARRAY ADAPTER

# LAYOUT

- 畫面中擺放Composite→ListView
- id設定為listview




# 程式碼

```
public class MainActivity extends Activity {  
    private static final String[] NAMES = {  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark"};  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        ListView listView = (ListView) findViewById(R.id.listview);  
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
            android.R.layout.simple_list_item_1, NAMES);  
        listView.setAdapter(adapter);  
    }  
}
```

# 程式碼

```
public class MainActivity extends Activity {  
    private static final String[] NAMES = {  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark"};  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        ListView listView = (ListView) findViewById(R.id.listview);  
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
            android.R.layout.simple_list_item_1, NAMES);  
        listView.setAdapter(adapter);  
    }  
}
```



# 程式碼

```
public class MainActivity extends Activity {  
    private static final String[] NAMES = {  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark"};  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        ListView listView = (ListView) findViewById(R.id.listview);  
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
            android.R.layout.simple_list_item_1, NAMES);  
        listView.setAdapter(adapter);  
    }  
}
```

設定Activity介面

# 程式碼

```
public class MainActivity extends Activity {  
    private static final String[] NAMES = {  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark"};  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        ListView listView = (ListView) findViewById(R.id.listview);  
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
            android.R.layout.simple_list_item_1, NAMES);  
        listView.setAdapter(adapter);  
    }  
}
```

findViewById找出id為listview  
並轉型成ListView

# 程式碼

```
public class MainActivity extends Activity {  
    private static final String[] NAMES = {  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark"};  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        ListView listView = (ListView) findViewById(R.id.listview);  
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
            android.R.layout.simple_list_item_1, NAMES);  
        listView.setAdapter(adapter);  
    }  
}
```

建立一個ArrayAdapter  
列表的資料型態是字串

# 程式碼

```
public class MainActivity extends Activity {  
    private static final String[] NAMES = {  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark"};  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        ListView listView = (ListView) findViewById(R.id.listView);  
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
            android.R.layout.simple_list_item_1, NAMES);  
        listView.setAdapter(adapter);  
    }  
}
```

參數1是目前的Activity



# 程式碼

```
public class MainActivity extends Activity {  
    private static final String[] NAMES = {  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark"};  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        ListView listView = (ListView) findViewById(R.id.listview);  
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
            android.R.layout.simple_list_item_1, NAMES);  
        listView.setAdapter(adapter);  
    }  
}
```

參數2是ListView的每一個條列呈現時  
要使用的layout

# 程式碼

```
public class MainActivity extends Activity {  
    private static final String[] NAMES = {  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark"};  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        ListView listView = (ListView) findViewById(R.id.listview);  
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
            android.R.layout.simple_list_item_1, NAMES);  
        listView.setAdapter(adapter);  
    }  
}
```

本處使用Android Framework提供的  
android.R.layout.simple\_list\_item1

# 程式碼

```
public class MainActivity extends Activity {  
    private static final String[] NAMES = {  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark"};  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        ListView listView = (ListView) findViewById(R.id.listview);  
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
            android.R.layout.simple_list_item_1, NAMES);  
        listView.setAdapter(adapter);  
    }  
}
```

這個layout的細節，可以看  
<SDK資料夾>/platforms/<Android版本>  
/data/res/layout/simple\_list\_item\_1.xml

# 程式碼

```
public class MainActivity extends Activity {  
    private static final String[] NAMES = {  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark"};  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        ListView listView = (ListView) findViewById(R.id.listview);  
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
            android.R.layout.simple_list_item_1, NAMES);  
        listView.setAdapter(adapter);  
    }  
}
```

參數3是ListView要呈現的資料陣列  
型態必須得與建立Adapter時一致

# 程式碼

```
public class MainActivity extends Activity {  
    private static final String[] NAMES = {  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark"};  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        ListView listView = (ListView) findViewById(R.id.listView);  
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
            android.R.layout.simple_list_item_1, NAMES);  
        listView.setAdapter(adapter);  
    }  
}
```

型態必須得與建立Adapter時一致  
這就是建立時的型態

# 程式碼

```
public class MainActivity extends Activity {  
    private static final String[] NAMES = {  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark"};  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        ListView listView = findViewById(R.id.listview);  
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
            android.R.layout.simple_list_item_1, NAMES);  
        listView.setAdapter(adapter);  
    }  
}
```

最後記得要呼叫setAdapter  
將建立好的Adapter設定給ListView

androidbasic/ListViewEventExample

# LISTVIEW事件

# 點擊事件

- 在ListView中可以使用OnItemClickListener接收到點擊事件



# 程式碼

```
public class MainActivity extends Activity {
    private static final String[] NAMES = {
        "John", "Luke", "Matthew", "Peter", "Mark",
        "John", "Luke", "Matthew", "Peter", "Mark",
        "John", "Luke", "Matthew", "Peter", "Mark",
        "John", "Luke", "Matthew", "Peter", "Mark"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ListView listView = (ListView) findViewById(R.id.listview);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1, NAMES);
        listView.setAdapter(adapter);
        listView.setOnItemClickListener(mOnItemClickListener);
    }
    private OnItemClickListener mOnItemClickListener = ...
}
```

# 程式碼

```
public class MainActivity extends Activity {  
    private static final String[] NAMES = {  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark"};  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        ListView listView = (ListView) findViewById(R.id.listview);  
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
            android.R.layout.simple_list_item_1, NAMES);  
        listView.setAdapter(adapter);  
        listView.setOnItemClickListener(mOnItemClickListener);  
    }  
    private OnItemClickListener mOnItemClickListener = ...  
}
```

使用setOnItemClickListener設定  
ListView每個條列的事件接收器

# 程式碼

```
public class MainActivity extends Activity {  
    private static final String[] NAMES = {  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark",  
        "John", "Luke", "Matthew", "Peter", "Mark"};  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        ListView listView = (ListView) findViewById(R.id.listview);  
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
            android.R.layout.simple_list_item_1, NAMES);  
        listView.setAdapter(adapter);  
        listView.setOnItemClickListener(mItemClickListener);  
    }  
    private OnItemClickListener mOnItemClickListener = ...  
}
```

事件接收器在此宣告成「成員變數」

# 程式碼

事件接收器在此宣告成「成員變數」

```
private OnItemClickListener mOnItemClickListener = new
OnItemClickListener() {

    @Override
    public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
        Toast.makeText(MainActivity.this, NAMES[position],
        Toast.LENGTH_SHORT).show();
    }
};
```

# 程式碼

```
private OnItemClickListener mOnItemClickListener = new  
OnItemClickListener() {  
  
    @Override  
    public void onItemClick(AdapterView<?> parent, View view, int  
        position, long id) {  
        Toast.makeText(MainActivity.this, NAMES[position],  
            Toast.LENGTH_SHORT).show();  
    }  
};
```

使用new建立OnItemClickListener

# 程式碼

```
private OnItemClickListener mOnItemClickListener = new  
OnItemClickListener() {
```

當有點擊到ListView的Item的時候  
onItemClick就會被呼叫

```
    @Override  
    public void onItemClick(AdapterView<?> parent, View view, int  
        position, long id) {  
        Toast.makeText(MainActivity.this, NAMES[position],  
            Toast.LENGTH_SHORT).show();  
    }  
};
```

# 程式碼

```
private OnItemClickListener mOnItemClickListener = new  
OnItemClickListener()
```

參數1其實就是ListView，  
AdapterView為ListView的父類別

```
@Override  
public void onItemClick(AdapterView<?> parent, View view, int  
position, long id) {  
    Toast.makeText(MainActivity.this, NAMES[position],  
        Toast.LENGTH_SHORT).show();  
}  
};
```

# 程式碼

```
private OnItemClickListener mOnItemClickListener = new  
OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> parent, View view, int  
        position, long id) {  
        Toast.makeText(MainActivity.this, NAMES[position],  
            Toast.LENGTH_SHORT).show();  
    }  
};
```

參數2就是點擊到ListView的那一系列的View



# 程式碼

```
private OnItemClickListener mOnItemClickListener = new  
OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> parent, View view, int  
        position, long id) {  
        Toast.makeText(MainActivity.this, NAMES[position],  
            Toast.LENGTH_SHORT).show();  
    }  
};
```

參數3就是點擊到的ListView的索引值  
就等同於陣列的索引值

# Q & A