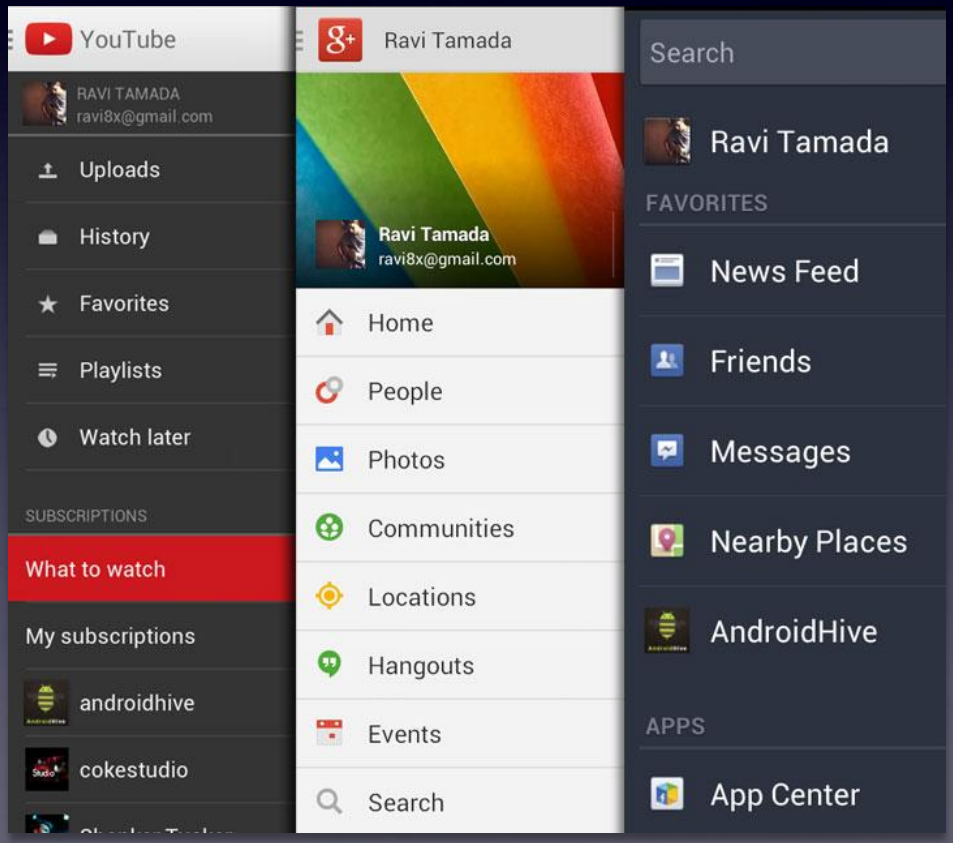
第十四章

NAVIGATION DRAWER

簡介

簡介

- 2013年開始陸續有許多APP推出了左方或右方可以拉開的menu選單



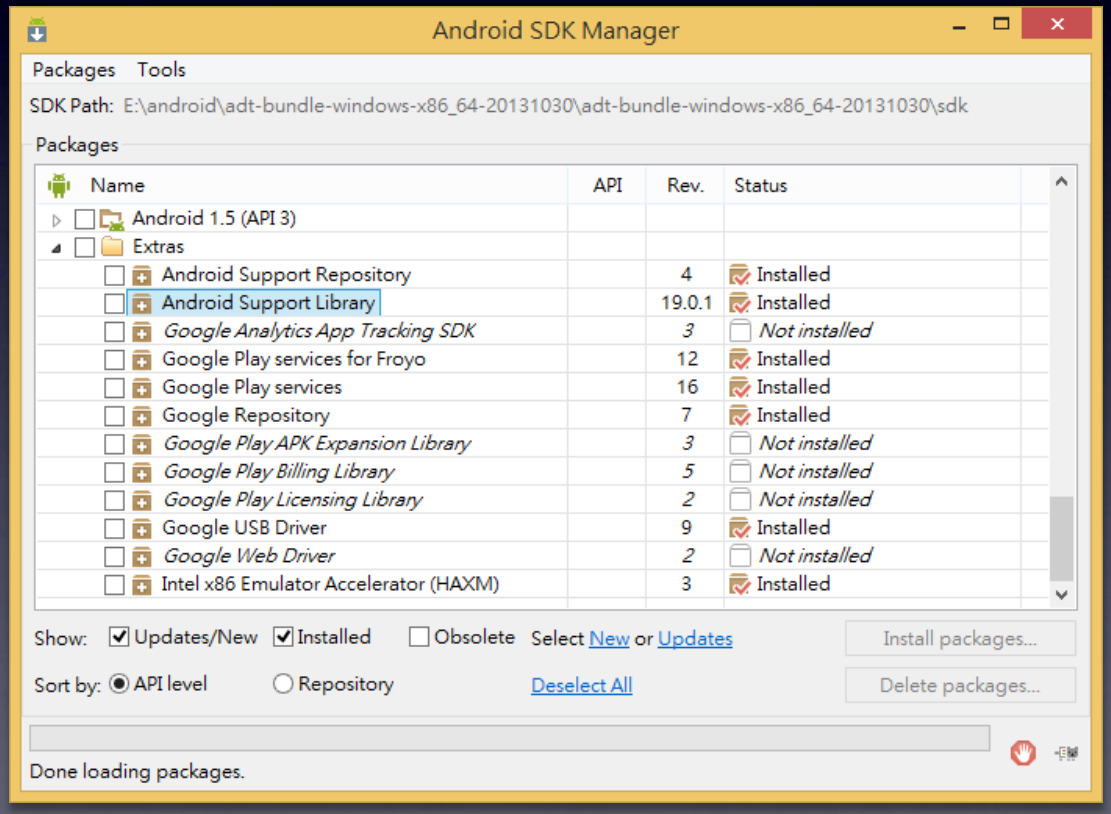
簡介

- Android今年終於由官方推出支援版本，稱為 Navigation Drawer
- 需要Android Support Library v7以上版本支援

前置作業

前置作業

- 打開Android SDK Manager
- 確認Android Support Library有下載



前置作業

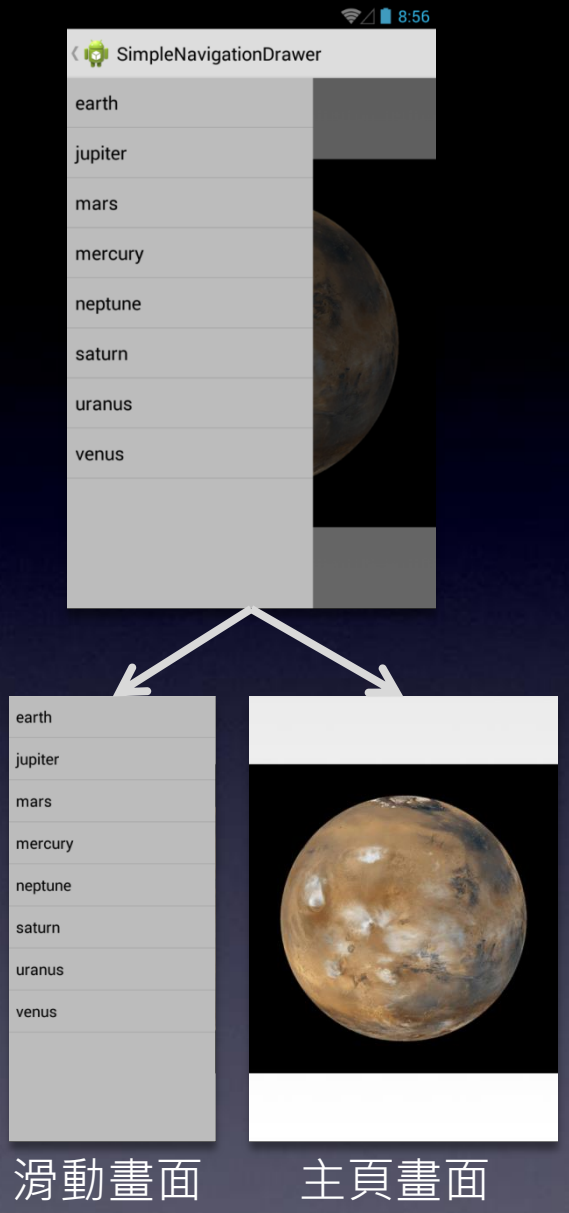
- 匯入Android Support Library v7版本的library project至eclipse
  - 路徑<sdk>\extras\android\support\v7\appcompat

SlideDrawer/SimpleNavigationDrawer

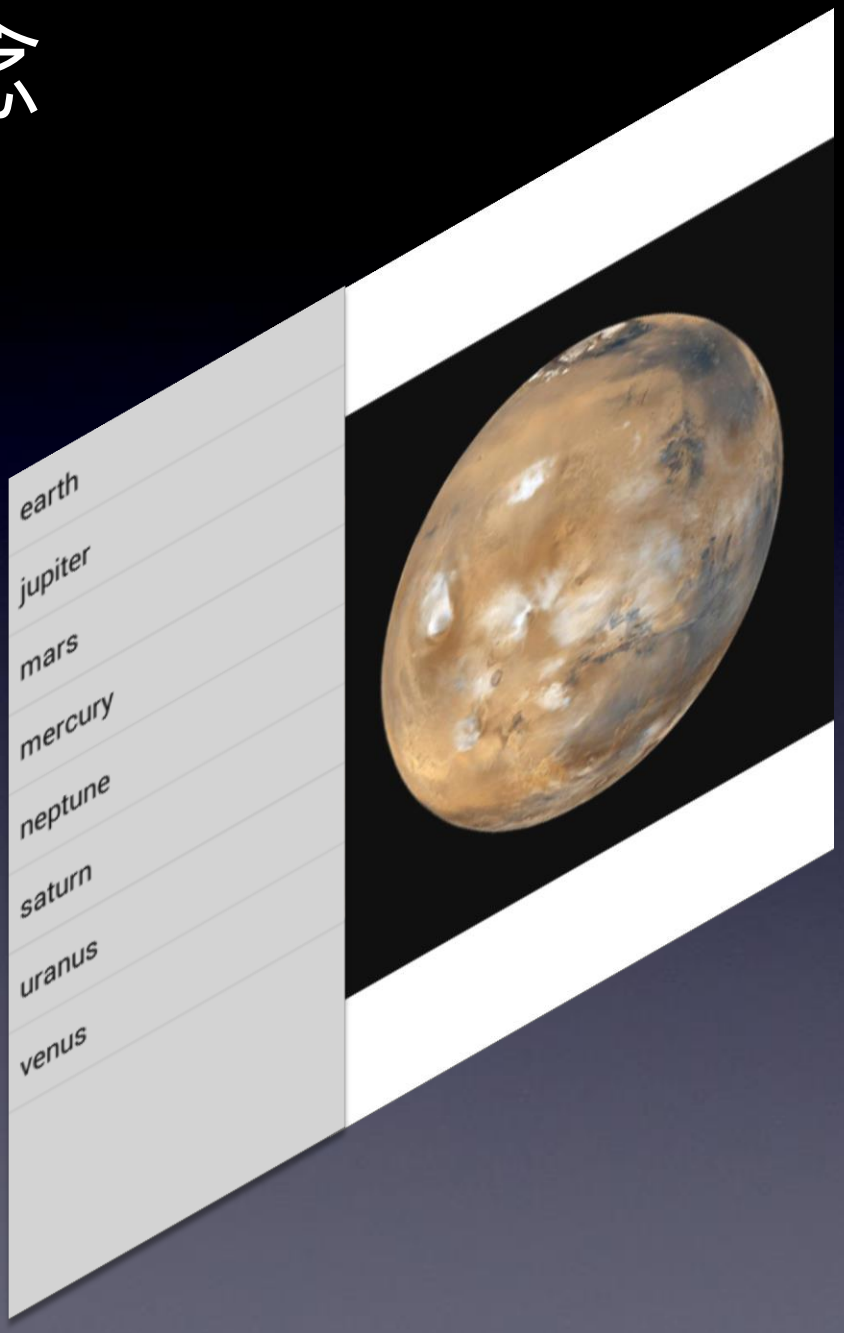
專案及介面設定

基本概念

- 如右圖所示
- 把滑動畫面當成一個 Fragment
- 而主頁畫面本身也使用 Fragment

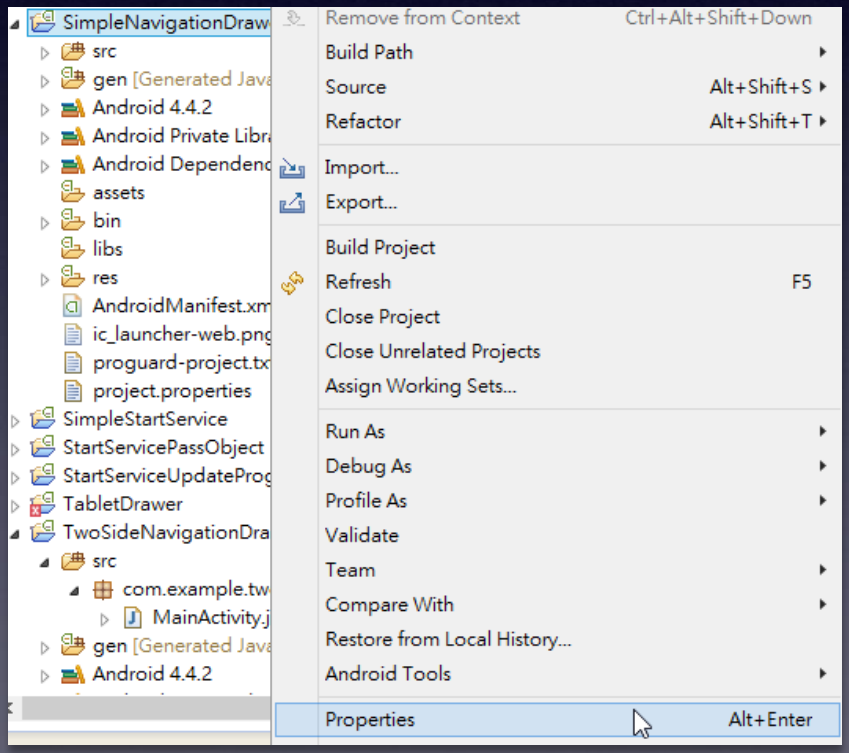


基本概念



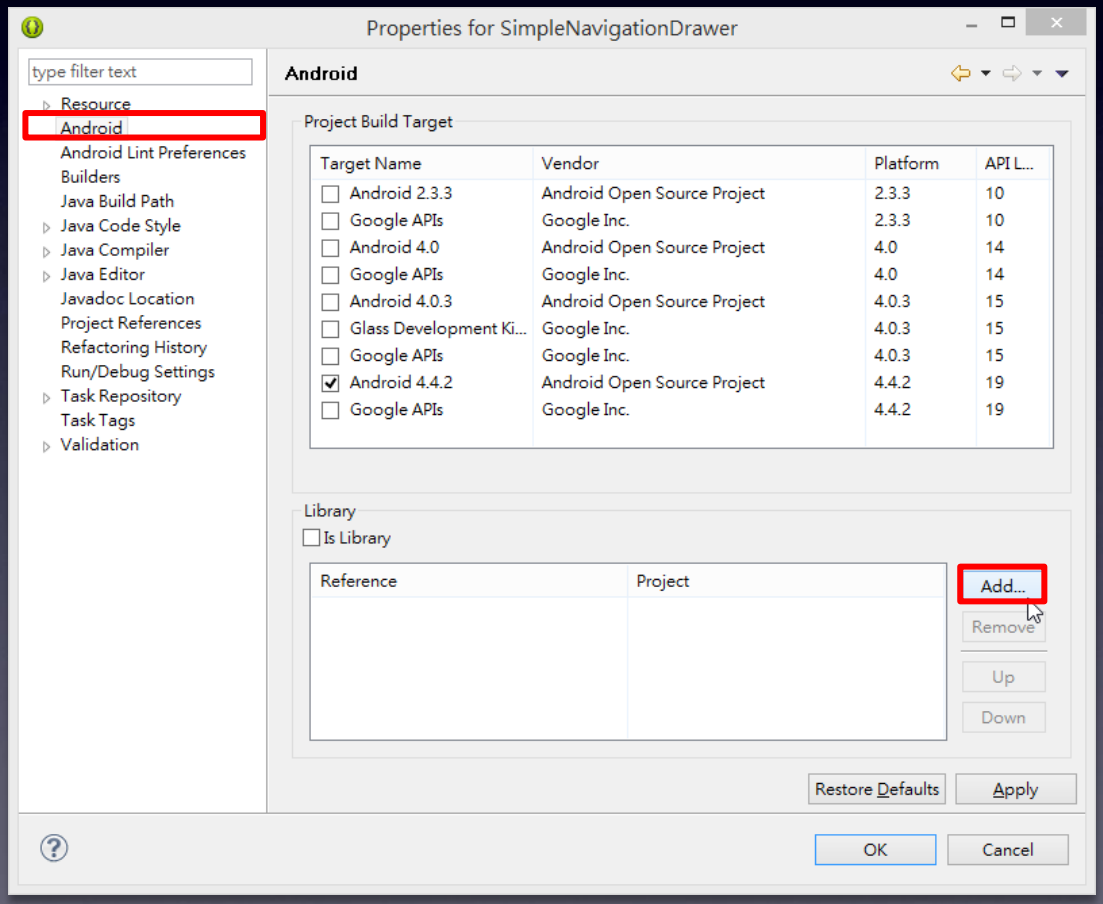
專案設定

- 使用NavigationDrawer的專案必須得加入Android Support Library v7
- 對專案按下滑鼠右鍵→Properties



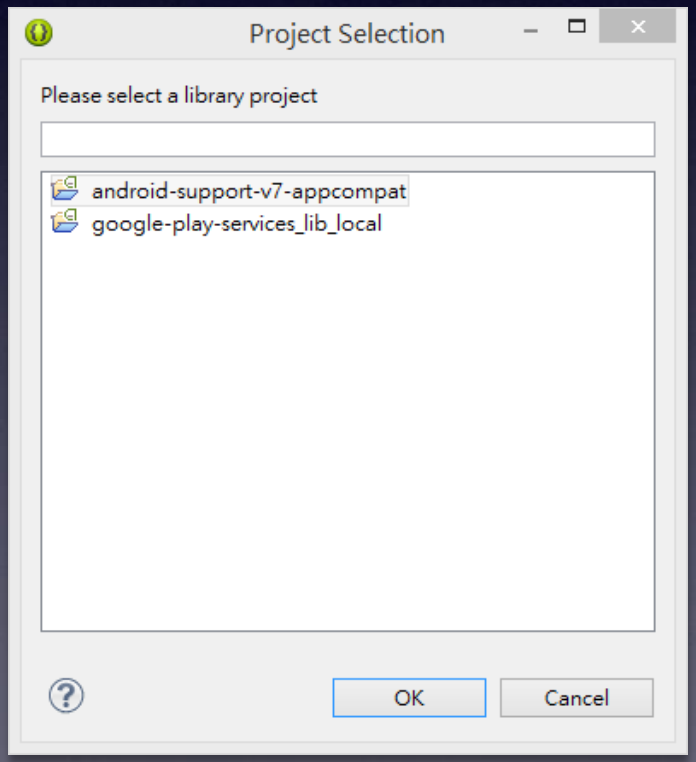
專案設定

- 對話視窗出現後，選擇左方的Android，並在右方畫面的下方欄選擇Add...



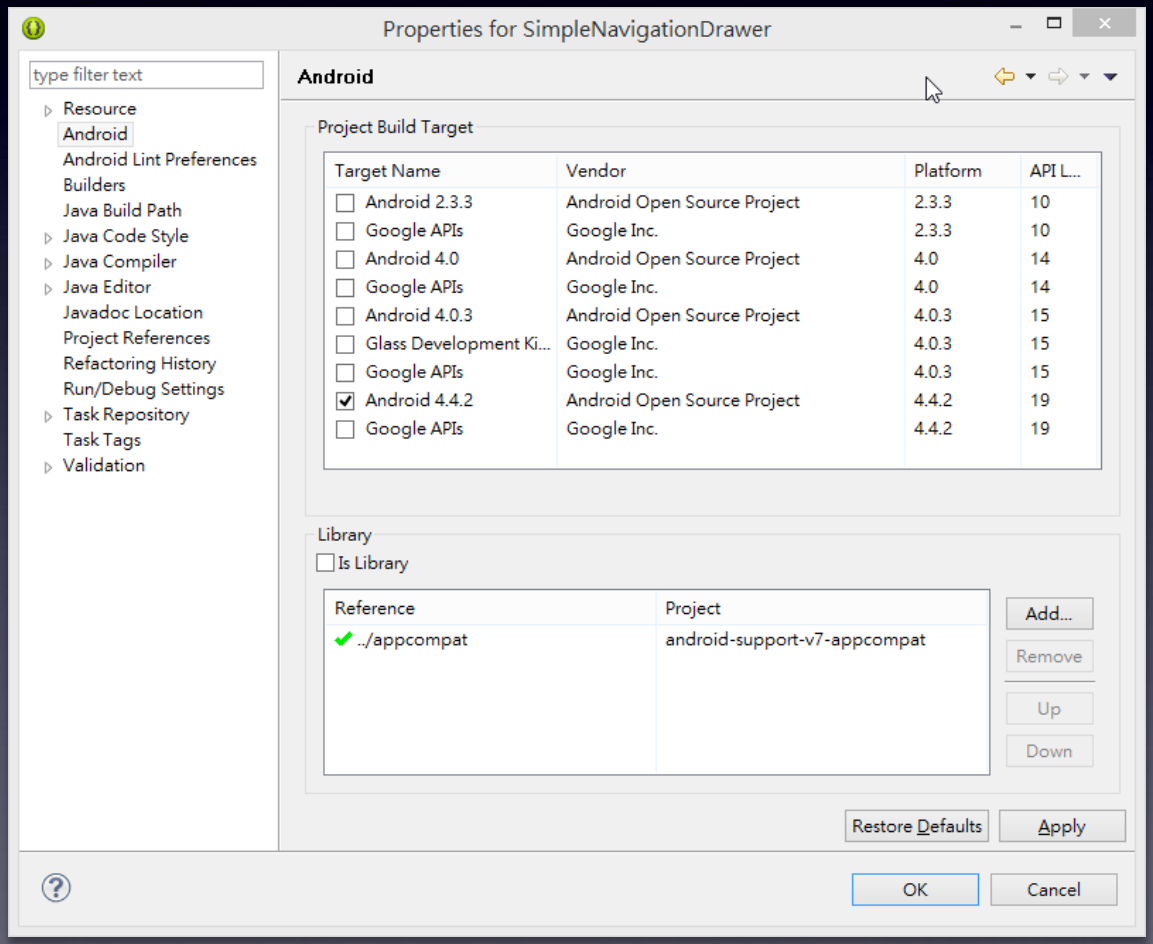
專案設定

- 出現的對話視窗會顯示目前workspace內可以匯入的Library類型專案有哪些
- 選擇android-support-v7-appcompat，按下OK



專案設定

- 設定完畢後的畫面如下



ANDROIDMANIFEST

- 打開專案的AndroidManifest.xml

- 使用Navigation Drawer，Activity的佈景主題必須是Android Support Library的Theme.AppCompat

ANDROIDMANIFEST

```
<application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/Theme.AppCompat" >
```

- 黃色文字就是使用指定的佈景主題
- 佈景主題可以調整
  - 淺色系：Theme.AppCompat.Light
  - 淺色系配上深色的標題列：Theme.AppCompat.Light.DarkActionBar

介面

- 如同本節一開始的基本概念介紹，Navigation Drawer是由兩個頁面重疊

- Android為了方便開發者製作這個效果而製作出了DrawerLayout

介面

```
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
        ……
</android.support.v4.widget.DrawerLayout>
```

- android.support.v4.widget.DrawerLayout 表示要使用NavigationDrawer
- 要組成的畫面的View就寫在之間

介面

```
<android.support.v4.widget.DrawerLayout …>
    <FrameLayout
        android:id="@+id/content"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
    <FrameLayout
        android:id="@+id/menu_root"
        android:layout_width="240dp"
        android:layout_height="match_parent"
        android:layout_gravity="left"
        android:background="#BCBCBC" />
</android.support.v4.widget.DrawerLayout>
```

- id/content 就是擺主頁畫面的View
- id/menu_root 是擺滑動畫面的View

介面

```
<android.support.v4.widget.DrawerLayout …>
    <FrameLayout
        android:id="@+id/content"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
    <FrameLayout
        android:id="@+id/menu_root"
        android:layout_width="240dp"
        android:layout_height="match_parent"
        android:layout_gravity="left"
        android:background="#BCBCBC" />
</android.support.v4.widget.DrawerLayout>
```

- id/menu_root這個view中的屬性layout_width可以決定滑動畫面的寬度為多少

介面

```
<android.support.v4.widget.DrawerLayout …>
    <FrameLayout
        android:id="@+id/content"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
    <FrameLayout
        android:id="@+id/menu_root"
        android:layout_width="240dp"
        android:layout_height="match_parent"
        android:layout_gravity="left"
        android:background="#BCBCBC" />
</android.support.v4.widget.DrawerLayout>
```

- layout_gravity是非常重要的屬性
- 針對滑動畫面，layout_gravity一定要是left或是right

介面

```
<android.support.v4.widget.DrawerLayout …>
    <FrameLayout
        android:id="@+id/content"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
    <FrameLayout
        android:id="@+id/menu_root"
        android:layout_width="240dp"
        android:layout_height="match_parent"
        android:layout_gravity="left"
        android:background="#BCBCBC" />
</android.support.v4.widget.DrawerLayout>
```

- DrawerLayout內寫的View有順序性的，滑動畫面一定要寫在主頁畫面之後

SlideDrawer/SimpleNavigationDrawer

程式撰寫

程式撰寫

```
public class MainActivity extends
ActionBarActivity {
```

- 首先，Activity一定要繼承ActionBarActivity
- 這是包在Android Support Library v7之中

程式撰寫

```
DrawerLayout mDrawerLayout = (DrawerLayout)
findViewById(R.id.drawer_layout);
```

- 藉由findViewById()找出DrawerLayout
- 開啟或關閉滑動畫面都是藉由DrawerLayout來達成

程式撰寫

```
mDrawerLayout.openDrawer(Gravity.LEFT);
```

- 打開滑動畫面
- 參數可以指定是左邊(Gravity.LEFT)或是右邊(Gravity.RIGHT)的滑動畫面
- 這必須得搭配layout設定中，還記得滑動畫面的屬性layout_gravity嗎？

程式撰寫

```
mDrawerLayout.closeDrawer(Gravity.LEFT);
mDrawerLayout.isDrawerOpen(Gravity.LEFT);
```

- closeDrawer() 關閉指定的滑動畫面
- isDrawerOpen() 檢查滑動畫面是否開啟

程式撰寫

```
mDrawerLayout.setDrawerLockMode(int);
```

- 限制滑動畫面開啟的方式
- 參數可以是
  - DrawerLayout.LOCK_MODE_UNLOCKED 不做任何限制，這也是預設模式
  - DrawerLayout.LOCK_MODE_LOCKED 不讓使用者直接用手指滑動開啟，而必須得由程式決定
  - DrawerLayout.LOCK_MODE_OPEN 保持滑動畫面開啟

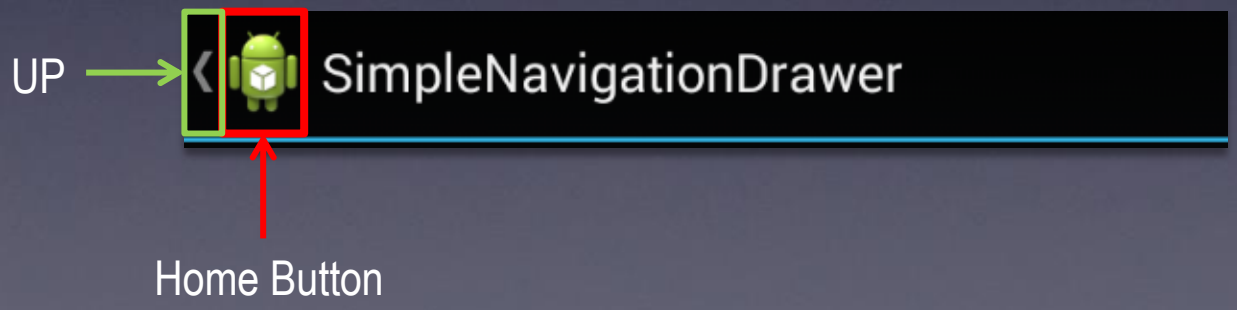
SlideDrawer/SimpleNavigationDrawer

與ACTIONBAR整合

與ACTIONBAR整合

- 一般滑動畫面的開啟，都會搭配app左上方的圖示點下後開啟

- 左上方圖示所在的區域稱為ActionBar
- 左上方的圖示稱為Home Button
- 左上方圖示旁的箭頭圖標稱為UP



與ACTIONBAR整合

```
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
getSupportActionBar().setHomeButtonEnabled(true);
```

- 在ActionBarActivity中，可以呼叫getSupportActionBar()取得ActionBar

與ACTIONBAR整合

```
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
getSupportActionBar().setHomeButtonEnabled(true);
```

- setDisplayHomeAsUpEnabled()
  - 決定是否要出現UP的箭頭圖標
- setHomeButtonEnabled()
  - 決定ActionBar上方的Home按鈕是否可以按

與ACTIONBAR整合

```
public boolean onOptionsItemSelected(MenuItem item) {
      if (item.getItemId() == android.R.id.home) {
      }
      return super.onOptionsItemSelected(item);
}
```

- 如何得知ActionBar上的Home Button按下了？
- 要在onOptionsItemSelected()中取得，可以看第五章、選項選單回憶一下

與ACTIONBAR整合

```
public boolean onOptionsItemSelected(MenuItem item) {
      if (item.getItemId() == android.R.id.home) {
      }
      return super.onOptionsItemSelected(item);
}
```

- ActionBar的Home Button，系統預設有給一個id android.R.id.home
- 藉由一般處理menu的方式，就可以處理ActionBar上按鈕的問題了

SlideDrawer/TwoSideNavigationDrawer

右方滑動畫面與DRAWER事件接收

介面

```
<android.support.v4.widget.DrawerLayout …>
    <FrameLayout
        android:id=“@+id/content” …/>
    <FrameLayout
        android:id="@+id/left_menu_root"
        android:layout_gravity="left"
        android:background=“#BCBCBC” />
    <FrameLayout
        android:id="@+id/right_menu_root"
        android:layout_gravity="right"
        android:background="#BCBCBC“/>
</android.support.v4.widget.DrawerLayout>
```

- 加入一個右邊的滑動畫面，記得layout_gravity的數值一定要left和right

程式碼

```
mDrawerLayout = (DrawerLayout)
findViewById(R.id.drawer_layout);
mDrawerLayout.openDrawer(Gravity.RIGHT);
mDrawerLayout.closeDrawer(Gravity.RIGHT);
mDrawerLayout.isDrawerOpen(Gravity.RIGHT);
```

- 開啟或關閉以及檢查右方的滑動畫面，作法與左方完全相同，差別是參數要帶Gravity.RIGHT

收到DRAWER的事件

- Navigation Drawer有提供事件接收的功能，稱為ActionBarDrawerToggle，接收的事件包括
  - 滑動畫面開啟
  - 滑動畫面關閉
  - 滑動畫面正在被滑動

收到DRAWER的事件

```
ActionBarDrawerToggle drawerToggle = new ActionBarDrawerToggle(this,
mDrawerLayout, R.drawable.ic_drawer, 0, 0) {
        @Override
        public void onDrawerClosed(View drawerView) {
        }
        @Override
        public void onDrawerOpened(View drawerView) {
        }
};
```

- 建立Drawer事件接收器的方法就是new一個ActionBarDrawerToggle
  - 參數1: Activity
  - 參數2: DrawerLayout
  - 參數3: 左上方UP的圖示

收到DRAWER的事件

```
ActionBarDrawerToggle drawerToggle = new ActionBarDrawerToggle(this,
mDrawerLayout, R.drawable.ic_drawer, 0, 0) {
        @Override
        public void onDrawerClosed(View drawerView) {
        }
        @Override
        public void onDrawerOpened(View drawerView) {
        }
};
```

- 開發者可以藉由Override來決定要處理哪些事件
- 本例子中Override了onDrawerClosed(), onDrawerOpened()

收到DRAWER的事件

```
mDrawerLayout.setDrawerListener(drawerToggle);
drawerToggle.syncState();
```

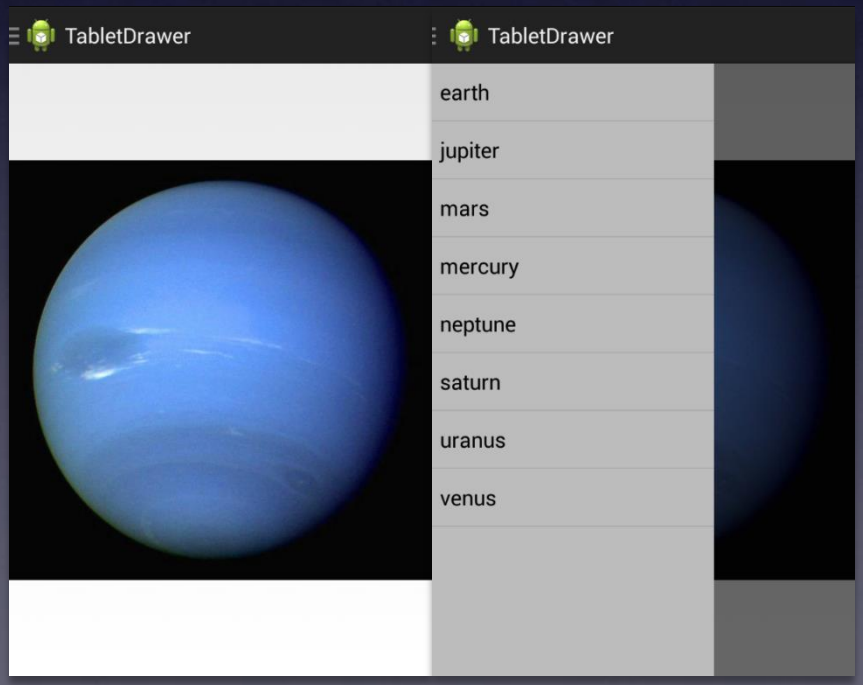
- 建立完ActionDrawerToggle，要使用setDrawerListener()設定給DrawerLayout

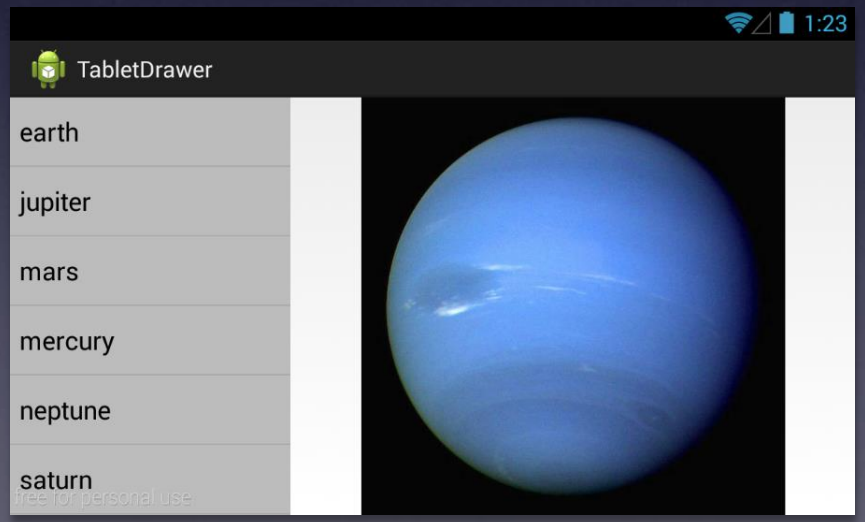
- 最後呼叫ActionDrawerToggle的syncState()將左上方UP的圖示更換

練習

練習

- 試著做出以下效果，可以讓你有能力建立跨手機和平板的介面
  - 當裝置垂直時，可以用Navigation Drawer顯示出選單
  - 當裝置水平時，讓選單固定在左邊





更多資料

更多資料

- https://developer.android.com/reference/android/support/v4/widget/DrawerLayout.html