

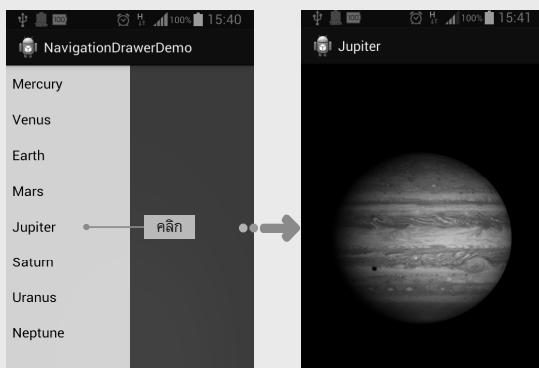
CHAPTER

# 13

## Fragment และ UI Navigation

### เนื้อหาในบทนี้

- ◆ การสร้าง UI Module โดยใช้ Fragment
- ◆ แฟร์กเมนต์ และการรองรับปุ่ม Back
- ◆ การสร้าง Navigation Drawer

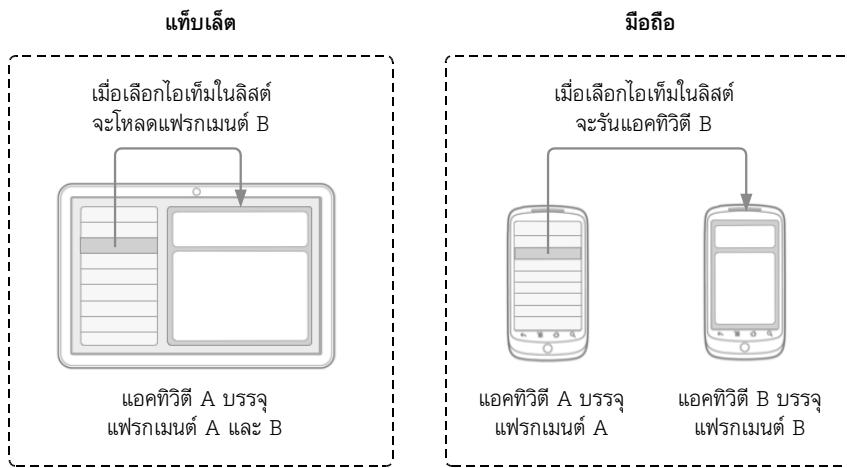


- ◆ การสร้าง Swipe Views
- ◆ การสร้าง ActionBar Tabs
- ◆ การใช้ Swipe Views ร่วมกับ Tabs

## การสร้าง UI Module โดยใช้ Fragment

แฟร์กเมนต์ (Fragment) เป็นไฟล์เอกสารที่เพิ่มเข้ามาในแอนดรอยด์ 3.0 (API level 11) โดยมีจุดประสงค์เพื่อการสร้าง UI ที่มีความยืดหยุ่น เป็นโมดูล และนำมาใช้ซ้ำ (reuse) ได้ง่าย ประโยชน์ของแฟร์กเมนต์ก็คือ

- ◆ สร้าง UI ที่สามารถปรับเปลี่ยน layout ไปตามขนาดหน้าจอของอุปกรณ์ ซึ่งเรียกว่า adaptive layout หรือ responsive layout
- ◆ สร้าง UI module (ส่วนของ UI ที่มีทั้งเค้าโครงและ layout) ที่สามารถนำไปใช้ซ้ำในหลายๆ แอคทิวิตี้ หรือเราจะนำหลายๆ แฟร์กเมนต์มาใช้ประกอบกันในแอคทิวิตี้เดียว ก็ได้ เช่น กัน เช่น แอพที่มี UI แบบ multi-pane



ด้วยวิธีการสร้าง UI module 2 โมดูลที่แยกจาก 2 แฟร์กเมนต์ ช่วยให้ลดที่สุดลงในการนำมาร่วมกันในแอคทิวิตี้เดียว กับหน้าจอแท็บเล็ต แต่ถูกใช้แยกกันบนหน้าจอมือถือ

แฟร์กเมนต์นั้นมี layout, lifecycle รวมถึง input event เป็นของตัวเอง เช่นเดียวกับแอคทิวิตี้ แต่เราไม่สามารถใช้งานแฟร์กเมนต์ตามลำพังโดยไม่มีแอคทิวิตี้ใด แฟร์กเมนต์ต้องถูกกำหนดเป็นส่วนหนึ่งของแอคทิวิตี้เสมอ ดังนั้นจึงอาจคิดง่ายๆ ว่าแฟร์กเมนต์ก็คือ "แอคทิวิตี้ย่อย" (sub activity) นั่นเอง

## การสร้างแฟร์กเม้นต์

แฟร์กเม้นต์คือคลาส เมื่อต้องการสร้างแฟร์กเม้นต์ขึ้นในโปรเจ็ค ให้สร้างคลาสใหม่โดยสืบทอด (inherit หรือ extend) จากคลาส Fragment หรือคลาสอื่นๆ ที่สืบทอดมาจาก Fragment อีกที เช่น ListFragment, DialogFragment แล้วทำการ override เมธอดต่างๆ ใน lifecycle ของแฟร์กเม้นต์ (onCreate, onResume ฯลฯ) เพื่อระบุการทำงานที่ต้องการลงไว้ในแต่ละอีเวนต์ของแฟร์กเม้นต์ แบบเดียวกับแอคทิวิตี้

สิ่งหนึ่งที่แตกต่างจากแอคทิวิตี้คือ การกำหนด layout ให้กับแฟร์กเม้นต์จะต้องทำในเมธอด onCreateView (สำหรับแอคทิวิตี้ เรามักกำหนด layout ในเมธอด onCreate)

ตัวอย่างการสร้างแฟร์กเม้นต์อย่างง่าย

```
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class PlanetDetailsFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {
        // โหลด layout จากไฟล์ fragment_planet_details.xml
        View layout = inflater.inflate(R.layout.fragment_planet_details,
                                      container, false); ❶
        ...
        return layout; ❷
    }
}
```

จากโค้ด การกำหนด layout ให้กับแฟร์กเม้นต์ ทำได้โดยเรียกเมธอด inflate ของออบเจ็ค LayoutInflater ที่ส่งมาเป็นพารามิเตอร์ตัวแรก (ตัวแปร inflater) ❶ เพื่อแปลงข้อมูล XML ใน layout file ที่ระบุไปเป็นออบเจ็ค View และส่งคืนออกไปจากเมธอด onCreateView ❷ ซึ่งจะทำให้ออบเจ็ค View นั้นถูกกำหนดเป็น layout ของแฟร์กเม้นต์

## การเพิ่มแฟร์กเมนต์ลงในแอคทิวิตี้

ดังที่บอกแล้วว่าเราไม่สามารถใช้งานแฟร์กเมนต์ได้ตามลำพัง แต่จะต้องเพิ่มมันลงในแอคทิวิตี้ (ถ้าพูดให้ถูกต้องจริงๆ ก็คือ เพิ่มลงใน layout ของแอคทิวิตี้) ซึ่งจะทำให้แฟร์กเมนต์ถูกลายเป็นส่วนหนึ่งใน view hierarchy ของแอคทิวิตี้

การเพิ่มแฟร์กเมนต์ลงในแอคทิวิตี้สามารถทำได้ 2 วิธีคือ การใช้ XML และใช้โค้ด Java

- ◆ การเพิ่มแฟร์กเมนต์ลงในแอคทิวิตี้โดยใช้ XML

เป็นการเพิ่มแฟร์กเมนต์ตั้งแต่ช่วงเขียนโปรแกรม โดยใส่แท็ก `<fragment>` ไว้ภายใน layout file ของแอคทิวิตี้ ยกตัวอย่างเช่น

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

    <fragment
        android:id="@+id/list_fragment"
        android:name="com.example.fragmentdemo.PlanetListFragment" ❶
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1" />

    <FrameLayout
        android:id="@+id/details_fragment_container"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="2" />

</LinearLayout>
```

แอคทิวิตี้ `android:name` ❶ ใช้ระบุชื่อคลาสของแฟร์กเมนต์ที่ต้องการ

### NOTE »»

การเพิ่มแฟร์กเมนต์ลงในแอคทิวิตี้โดยใช้ XML นี้ ทำให้มองได้ว่าแฟร์กเมนต์มีบทบาทเทียบเท่ากับวิชnid ต่างๆ เช่น ปุ่ม, TextView เป็นต้น แต่ที่จริงแล้วแฟร์กเมนต์ไม่ใช่วิชnid เพราะไม่ได้สืบทอดมาจากคลาส View นอกจากนี้แฟร์กเมนต์ยังมี lifecycle ที่คล้ายคลึงกับแอคทิวิตี้ (onCreate, onStart, onResume ฯลฯ) หลายคนจึงรู้สึกว่าบทบาทของมันใกล้เคียงกับแอคทิวิตี้มากกว่า

### ◆ การเพิ่มแฟร์กเมนต์ลงในแอคทิวิตี้โดยใช้โค้ด JAVA

เป็นการเพิ่มแฟร์กเมนต์ในช่วงรันโปรแกรม (runtime) ตามเงื่อนไขที่เรากำหนด เช่นเมื่อผู้ใช้แตะไอเท็มในลิสต์ที่อยู่ฝั่งซ้ายของหน้าจอ เราจะค่อยแสดงแฟร์กเมนต์ที่เป็นรายละเอียดของไอเท็มนั้นออกมาทางฝั่งขวา เป็นต้น

ตัวอย่างการใช้โค้ด JAVA เพิ่มแฟร์กเมนต์ลงในแอคทิวิตี้

```
PlanetDetailsFragment fragment = new PlanetDetailsFragment();

FragmentManager fm = getFragmentManager();
FragmentTransaction transaction = fm.beginTransaction();
transaction.replace(R.id.details_fragment_container, fragment);
transaction.commit();
```

คือการสร้างแฟร์กเมนต์จากคลาส PlanetDetailsFragment และใส่ลงในวิวกรุ๊ปที่มี ID ว่า details\_fragment\_container

#### **NOTE»**

แฟร์กเมนต์ที่ถูกเพิ่มลงในแอคทิวิตี้โดยใช้โค้ด JAVA นี้จะเป็น dynamic fragment ซึ่งเราสามารถลบ (remove) ออกจากแอคทิวิตี้ รวมถึงแทนที่ด้วยแฟร์กเมนต์อื่นๆ ภายหลังได้ ในขณะที่การเพิ่มแฟร์กเมนต์ด้วย XML จะเป็น static fragment ซึ่งไม่สามารถลบหรือแทนที่ได้

## ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะสร้างแอพที่ให้ข้อมูลเกี่ยวกับดาวเคราะห์ในระบบสุริยะ ซึ่งมี UI navigation เป็นแบบ master-details (list-details) โดยเราแบ่ง UI ออกเป็น 2 มोดูลที่สร้างขึ้นจาก 2 แฟร์กเมนต์ ได้แก่ แฟร์กเมนต์ที่แสดงรายชื่อดาวเคราะห์ กับแฟร์กเมนต์ที่แสดงรายละเอียดของดาวเคราะห์ที่ถูกเลือก ซึ่งตัวอย่างนี้จะแสดงภาพดาวเคราะห์อย่างเดียว

ทั้งนี้ หากรันแอพบนอุปกรณ์ขอใหญ่ (ไม่ว่าแนวตั้งหรือแนวนอน) จะแสดงแฟร์กเมนต์ทั้งสองคู่กัน แต่หากรันบนอุปกรณ์จอลีกจะแสดงรายชื่อดาวเคราะห์ในหน้าจอแรก และเมื่อเลือกชื่อดาวเคราะห์แล้วจะึง แสดงภาพดาวเคราะห์นั้นในอีกหน้าจอหนึ่ง (ครูปในผลการรัน)

- 1 สร้างคลาสใหม่ชื่อ PlanetListFragment เพื่อให้เป็นแฟร์กเมนต์ที่แสดงรายชื่อดาวเคราะห์ โดยพิมพ์โค้ดเริ่มต้นดังนี้

**โปรเจ็ค FragmentDemo, ไฟล์ PlanetListFragment.java**

```
package com.example.fragmentdemo;

import android.app.Activity;
```

```
import android.app.ListFragment;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class PlanetListFragment extends ListFragment { ❶

    // ชื่อดาวเคราะห์ทั้งหมดในระบบสุริยะ
    protected static final String[] mPlanetTitles = new String[] {
        "Mercury", "Venus", "Earth", "Mars",
        "Jupiter", "Saturn", "Uranus", "Neptune"
    };

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // สร้าง adapter และกำหนด adapter ให้กับ ListView ของแฟร์กเมนต์
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(
            getActivity(), android.R.layout.simple_list_item_activated_1,
            mPlanetTitles); ❷
        setListAdapter(adapter); ❸
    }

    @Override
    public void onStart() {
        super.onStart();

        // ให้แสดงสถานการเลือกใน ListView กรณีเป็น Layout สำหรับจอยก高手
        if (getActivity().findViewById(R.id.details_fragment_container)
            != null) { ❹
            getListView().setChoiceMode(ListView.CHOICE_MODE_SINGLE); ❺
        }
    }
}
```

การกำหนดให้ `PlanetListFragment` สืบทอดจาก `ListFragment` ❶ ทำให้มันเป็นแฟร์กเมนต์ที่มี `ListView` ในตัว โดยที่เราไม่ต้องกำหนด `layout` ของแฟร์กเมนต์เอง (ทำเองเดียวกับ `ListActivity` ในบทที่ 4) เราจึงไม่ต้องเตรียมเมธอด `onCreateView` สำหรับแฟร์กเมนต์นี้ ภายในเมธอด `onCreate` เราสร้าง `adapter` ที่จะนำข้อมูลจากอาร์เรย์ `mPlanetTitles` มาแสดงผล ใน `ListView` ❷ ก่อนที่จะกำหนด `adapter` นั้นให้กับ `ListView` ของ `ListFragment` ❸

สำหรับเมธอด onStart เรายังตรวจสอบ layout ที่แอคทิวิตี้ (MainActivity) ใช้อยู่ในขณะนั้นว่าเป็น layout สำหรับจอใหญ่ (ไฟล์ layout-large\activity\_main.xml) ใช่หรือไม่ โดยค้นหาว่ามี ID ว่า details\_fragment\_container ④ ซึ่งจะมีใน layout สำหรับจอใหญ่ แต่ไม่มีใน layout สำหรับจอเล็ก (ไฟล์ layout\activity\_main.xml)

ถ้าหากพบว่าเป็น layout สำหรับจอใหญ่ เราจะกำหนดโหมดการเลือกของ ListView เป็น CHOICE\_MODE\_SINGLE ⑤ (ค่าดีฟอลต์คือ CHOICE\_MODE\_NONE) เพื่อแสดงสถานะการเลือก (แบบไฮไลท์) ที่ชื่อดาวเคราะห์ที่ถูกเลือก ส่วนกรณีจ่อเลือกเมื่อเลือกชื่อดาวเคราะห์จาก ListView ก็จะแสดงอีกหน้าจอหนึ่งขึ้นมา จึงไม่ต้องแสดงสถานะการเลือกให้เห็น

## 2 เพิ่มโค้ดในคลาส PlanetListFragment

```
โปรเจ็ค FragmentDemo, ไฟล์ PlanetListFragment.java
private OnPlanetListSelectedListener mCallback;

public interface OnPlanetListSelectedListener { ❶
    public void onPlanetListSelected(int position); ❷
}

@Override
public void onAttach(Activity activity) {
    super.onAttach(activity);

    try {
        mCallback = (OnPlanetListSelectedListener) activity; ❸
    } catch (ClassCastException e) {
        throw new ClassCastException(activity.toString() +
            " must implement OnPlanetListSelectedListener"); ❹
    }
}

@Override
public void onListItemClick(ListView l, View v, int position, long id) {
    mCallback.onPlanetListSelected(position); ❺
    getListView().setItemChecked(position, true); ❻
}
```

สิ่งที่ต้องทำเมื่อผู้ใช้เลือกชื่อดาวเคราะห์จากกลิสต์ก็คือ การโหลดแฟร์กเมนต์ที่แสดงภาพดาวเคราะห์ขึ้นมา อย่างไรก็ตาม เอกสารของแอนดรอยด์แนะนำว่าแฟร์กเมนต์ต่างๆ ในแอคทิวิตี้ไม่ควรติดต่อกัน เองโดยตรง เช่นกรณีแฟร์กเมนต์ที่แสดงรายชื่อดาวเคราะห์ไม่ควรไปโหลดแฟร์กเมนต์ที่แสดงภาพ

ดาวเคราะห์โดยตรง เพื่อให้แฟร์กเมนต์เป็นอิสระต่อ กัน และนำมาใช้ช้าได้ง่าย โดยวิธีที่แนะนำคือ ให้เราส่งต่อการทำงานไปยังแอคทิวิตี้ที่บรรจุแฟร์กเมนต์นั้นไว้

วิธีส่งต่อการทำงานไปยังแอคทิวิตี้ ทำได้โดยประกาศอินเทอร์เฟซ (Interface ในภาษาจawa) ขึ้นมา ในแฟร์กเมนต์ แล้วให้แอคทิวิตี้ที่บรรจุแฟร์กเมนต์นั้นทำการ implement อินเทอร์เฟชนั้น

ในที่นี่สร้างอินเทอร์เฟซ OnPlanetListSelectedListener ❶ ที่มีเมธอด onPlanetListSelected อยู่เพียงเมธอดเดียว ❷ จากนั้นในเมธอด onAttach ของแฟร์กเมนต์ เราจะแปลง (cast) แอคทิวิตี้ เป็นอินเทอร์เฟซ OnPlanetListSelectedListener ซึ่งถ้า cast สำเร็จแสดงว่าแอคทิวิตี้ได้ implement อินเทอร์เฟชดังกล่าว เราเก็บ reference ของแอคทิวิตี้ (ที่ cast แล้ว) ลงในตัวแปร mCallback ❸ แต่ถ้า cast ไม่สำเร็จแสดงว่าแอคทิวิตี้ไม่ได้ implement อินเทอร์เฟชนั้น เราเก็บ จะโยนข้อผิดพลาดชนิด ClassCastException ออกไป พร้อมทั้งระบุรายละเอียดในข้อผิดพลาด ให้รู้ ❹

#### NOTE >>>

เมธอด onAttach เป็นเมธอดหนึ่งใน lifecycle ของแฟร์กเมนต์ ซึ่งจะทำงานตอนที่แฟร์กเมนต์เชื่อมโยงเข้ากับแอคทิวิตี้ โดยแอคทิวิตี้จะถูกส่งผ่านมาเป็นพารามิเตอร์ของเมธอดนี้

เมื่อผู้ใช้เลือกชื่อดาวเคราะห์หนึ่งๆจากลิสต์ เราจะเรียกต่อไปยังเมธอด onPlanetListSelected ใน แอคทิวิตี้ ❺ แล้วกำหนดสถานะของไอเท็มนั้นให้ผู้ใช้รู้ว่าถูกเลือกแล้ว ❻ (เมธอด setItemChecked จะมีผลเมื่อโหมดการเลือกของ ListView เป็นแบบ CHOICE\_MODE\_SINGLE หรือ CHOICE\_MODE\_MULTIPLE เท่านั้น)

ถัดไปจะสร้างแฟร์กเมนต์ที่แสดงภาพดาวเคราะห์ที่ถูกเลือก

### 3 สร้างคลาสใหม่ชื่อ PlanetDetailsFragment และพิมพ์โค้ดดังนี้

#### โปรเจ็ค FragmentDemo, ไฟล์ PlanetDetailsFragment.java

```
package com.example.fragmentdemo;

import java.util.Locale;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;

public class PlanetDetailsFragment extends Fragment {
    // คีย์ของข้อมูล (ชื่อดาวเคราะห์) ที่จะเก็บไว้ที่ตัวแฟร์กเมนต์เอง
    private static final String ARG_PLANET_TITLE = "planet_title";
```

```

// ชื่อดาวเคราะห์ที่จะแสดงภาพอุอกมาในแฟร์กเมนต์นี้
private String mPlanetTitle;

/* Factory method ที่ใช้สร้างอินสแตนซ์ของแฟร์กเมนต์ แทนการใช้คอนสตรัคเตอร์
โดยจะกำหนดชื่อดาวเคราะห์เป็นอาร์กิวเมนต์ที่ดีป้าบันแฟร์กเมนต์ด้วย */
public static PlanetDetailsFragment newInstance(String planetTitle) {
    PlanetDetailsFragment fragment = new PlanetDetailsFragment(); ❶
    Bundle args = new Bundle();
    args.putString(ARG_PLANET_TITLE, planetTitle); ❷
    fragment.setArguments(args);
}

return fragment; ❸
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    mPlanetTitle = getArguments().getString(ARG_PLANET_TITLE); ❹
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                        Bundle savedInstanceState) {
    View layout = inflater.inflate(R.layout.fragment_planet_details,
                                   container, false); ❺
    ImageView planetImage = (ImageView)
        layout.findViewById(R.id.planet_image);

    int resourceId = getResources().getIdentifier(
        mPlanetTitle.toLowerCase(Locale.getDefault()),
        "drawable",
        getActivity().getPackageName());
    planetImage.setImageResource(resourceId); ❻
}

return layout;
}
}

```

คลาส PlanetDetailsFragment จะรับชื่อดาวเคราะห์เข้ามาเป็นพารามิเตอร์ เพื่อใช้ระบุชื่อไฟล์ภาพดาวเคราะห์ที่จะแสดงอุอกมา

การสร้างอินสแตนซ์ของ PlanetDetailsFragment และรับชื่อดาวเคราะห์เข้ามาในแฟร์กเมนต์นั้น จะใช้ static method ชื่อ newInstance แทนการใช้คอนสตรัคเตอร์ตามปกติ (เป็นวิธีที่เอกสารของ

แอนดรอยด์แนะนำไว้ ซึ่งเป็น design pattern ของ OOP ที่เรียกว่า factory method หรือเมธอดที่ทำหน้าที่เป็น "โรงงานสร้างออบเจ็ค"

เมธอด newInstance ดังกล่าวจะสร้างอินสแตนซ์ของ PlanetDetailsFragment ขึ้น ❶ ก่อนนำชื่อดาวเคราะห์มากำหนดเป็นอาร์กิวเมนต์ของแฟร์กเมนต์ ❷ แล้วส่งคืนแฟร์กเมนต์กลับออกไป

❸ วิธีนี้ทำให้ชื่อดาวเคราะห์ถูกเก็บไว้ที่ตัวแฟร์กเมนต์ ซึ่งจะไม่หายไปเมื่อแฟร์กเมนต์ถูกทำลายแล้วสร้างขึ้นใหม่ เช่นเมื่อผู้ใช้หมุนจอ เป็นต้น

ภายในเมธอด onCreate เราอ่านชื่อดาวเคราะห์ที่เก็บไว้ที่ตัวแฟร์กเมนต์เอง มากำหนดให้ตัวแปร mPlanetTitle ❹

#### **NOTE »»**

เหตุผลที่เราไม่กำหนดชื่อดาวเคราะห์ให้ตัวแปร mPlanetTitle ดังแต่ในเมธอด newInstance เลย

เพราะเมื่อแอนดรอยด์ทำลายแล้วสร้างแฟร์กเมนต์ขึ้นใหม่ แอนดรอยด์ไม่ได้เรียกใช้เมธอด newInstance

ของเราระบบสร้างแฟร์กเมนต์ แต่จะใช้ชุดนอฟต์แวร์ตามปกติ ดังนั้นวิธีของเรางานได้ไม่ดีว่า

แฟร์กเมนต์จะมีชื่อดาวเคราะห์เก็บอยู่ที่ตัวแปร mPlanetTitle แน่นอน ทั้งกรณีที่เราสร้างแฟร์กเมนต์เอง และกรณีที่แอนดรอยด์สร้างแฟร์กเมนต์ให้ใหม่อัตโนมัติหลังจากทำลายไป

ภายในเมธอด onCreateView เรากำหนด layout ให้กับแฟร์กเมนต์โดยใช้ไฟล์

fragment\_planet\_details.xml ❺ ซึ่งมี ImageView อัญชังใน จานวนแสดงรูปภาพตามชื่อดาวเคราะห์ในตัวแปร mPlanetTitle อ กมากที่ ImageView นี้ ❻

- 4 สร้าง layout file ชื่อ fragment\_planet\_details.xml เพื่อใช้เป็น layout ของ PlanetDetailsFragment

#### โปรเจ็ค FragmentDemo, ไฟล์ fragment\_planet\_details.xml

```
<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/planet_image"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#000000"
        android:gravity="center"
        android:padding="32dp" />

</LinearLayout>
```

ตอนนี้เราระบุ Fragment ที่เป็น UI module เสร็จเรียบร้อยแล้ว ประกอบด้วย

- ◆ คลาส `PlanetListFragment` คือ Fragment ที่แสดงรายการดาวเคราะห์ทั้งหมด, เป็น `ListFragment` ที่มี `ListView` อยู่ในตัว (ไม่ต้องใช้ layout file)
- ◆ คลาส `PlanetDetailsFragment` คือ Fragment ที่แสดงภาพดาวเคราะห์ที่ถูกเลือก, ใช้ layout จากไฟล์ `fragment_planet_details.xml`

ถัดไปจะสร้างแอคทิวิตี้และ layout ของแอคทิวิตี้ เพื่อนำ Fragment ทั้งสองมาใช้งานให้เหมาะสมกับขนาดหน้าจอของอุปกรณ์

## 5 กำหนด layout ของหน้าจอหลัก สำหรับจอเล็กและใหญ่ ตามลำดับ

### โปรเจ็ค FragmentDemo, ไฟล์ layout\activity\_main.xml

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

    <fragment ❶
        android:id="@+id/list_fragment"
        android:name="com.example.fragmentdemo.PlanetListFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

### โปรเจ็ค FragmentDemo, ไฟล์ layout-large\activity\_main.xml

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

    <fragment ❷
        android:id="@+id/list_fragment"
        android:name="com.example.fragmentdemo.PlanetListFragment"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1" />

    <FrameLayout ❸
        android:id="@+id/details_fragment_container"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="2" />

</LinearLayout>
```

สังเกตว่า layout สำหรับจอเล็กมีแฟร์กเมนต์ที่ระบุถึง PlanetListFragment เพียงอย่างเดียว ❶ ในขณะที่ layout สำหรับจอใหญ่มีแฟร์กเมนต์ดังกล่าว เช่น กัน ❷ และยังมี FrameLayout ที่เตรียมไว้บรรจุ PlanetDetailsFragment ในช่วง runtime ด้วย ❸ โดยแฟร์กเมนต์กับ FrameLayout จะวางคู่กันในแนวนอน (ซ้าย-ขวา) และอัตราส่วนความกว้างของทั้งสองส่วนนี้เป็น 1:2 (กำหนดโดย layout\_weight)

layout ทั้งสองมีชื่อไฟล์เหมือนกันคือ activity\_main.xml แต่อยู่คนละโฟลเดอร์ ซึ่งในช่วง runtime แอนดรอยด์จะตรวจสอบขนาดหน้าจอของอุปกรณ์ แล้วเลือกใช้ layout ที่เหมาะสมให้อัตโนมัติ โดย layout ในโฟลเดอร์ layout-large จะถูกใช้เมื่อจอมาตราฐาน 5-7 นิ้วโดยประมาณ

บนอุปกรณ์จอเล็ก เมื่อเลือกชื่อดาวเคราะห์จากลิสต์แล้วจะแสดงภาพดาวเคราะห์ในหน้าจอใหม่ที่ซ่อนขึ้นมาบนหน้าจอลิสต์ นั่นหมายความว่าเราต้องสร้างแอคทิวิตี้อีกอันหนึ่งนอกเหนือจาก MainActivity ที่เอาไว้ใช้ในกรณีจอเล็กโดยเฉพาะ ตามขั้นตอนที่จะอธิบายต่อไปนี้

## 6 สร้างคลาสใหม่ชื่อ PlanetDetailsActivity แล้วพิมพ์โค้ดดังนี้

### โปรเจ็ค FragmentDemo, ไฟล์ PlanetDetailsActivity.java

```
package com.example.fragmentdemo;

import android.app.Activity;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.content.Intent;
import android.os.Bundle;

public class PlanetDetailsActivity extends Activity {
    protected static final String PLANET_TITLE = "planet_title";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_planet_details);

        Intent intent = getIntent();
        String planetTitle = intent.getStringExtra(PLANET_TITLE); ❶
        setTitle(planetTitle); ❷

        /* ถ้าหากแอคทิวิตี้ถูกสร้างขึ้นครั้งแรก (ไม่ได้ถูกสร้างขึ้นใหม่หลังจากเพิ่งถูกทำลาย
        เนื่องจากมีการเปลี่ยนคอนฟิกของเครื่อง เช่น ผู้ใช้หมุนจอ เป็นต้น) เราจะสร้างอินสแตนซ์
        ของ PlanetDetailsFragment และเพิ่งลงใน layout

        ส่วนกรณีที่แอคทิวิตี้ถูกทำลายแล้วสร้างขึ้นใหม่นั้น แอนดรอยด์จะสร้างอินสแตนซ์ของ
```

```
PlanetDetailsFragment ให้ใหม่ และเพิ่มลงใน layout ให้อัตโนมัติ เราจึงไม่ต้อง
ทำอะไร */
if (savedInstanceState == null) {
    PlanetDetailsFragment fragment =
        PlanetDetailsFragment.newInstance(planetTitle); ③

    FragmentManager fm = getFragmentManager();
    FragmentTransaction transaction = fm.beginTransaction();
    transaction.replace(R.id.details_fragment_container,
                        fragment);
    transaction.commit(); ④
}
}
```

คลาส `PlanetDetailsActivity` จะรับชื่อดาวเคราะห์มาจาก `MainActivity` โดยผ่านทางอินเทนต์ ❶ จากนั้นจะแสดงชื่อดาวเคราะห์ที่นั่นบน title bar ❷, สร้างอินสแตนซ์ของ `PlanetDetailsFragment` โดยเรียกเมธอด `newInstance` พร้อมทั้งส่งชื่อดาวเคราะห์ที่ไปให้ ❸ และเพิ่มแฟร์กเมนต์ที่สร้างนั้นลงใน layout ❹ ซึ่งแอคทิวิตี้นี้จะใช้ layout จากไฟล์ `activity_planet_details.xml`

7 สร้างไฟล์ activity\_planet\_details.xml ที่เป็น layout ของ PlanetDetailsActivity

```
โปรเจ็ค FragmentDemo, ไฟล์ activity_planet_details.xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <FrameLayout ❶
        android:id="@+id/details_fragment_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

layout file นี้จะมีเพียง `FrameLayout` ❶ ที่เตรียมไว้บรรจุ `PlanetDetailsActivity` ในช่วง runtime เท่านั้น เพียงเท่ากับฝั่งขวาใน layout สำหรับจอยักษ์ของ `MainActivity` นั้นเอง ถัดไปจะเป็นการเขียนโค้ดใน `MainActivity` เพื่อระบุโค้ดการทำงานเมื่อเลือกชื่อดาวเคราะห์ จากลิสต์

8 แก้ไขโค้ดใน MainActivity เป็นดังนี้

โปรเจ็ค FragmentDemo, ไฟล์ MainActivity.java

```
package com.example.fragmentdemo;

import android.app.Activity;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.content.Intent;
import android.os.Bundle;

import com.example.fragmentdemo.PlanetListFragment.
    OnPlanetListSelectedListener;

public class MainActivity extends Activity
    implements OnPlanetListSelectedListener { ❶

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public void onPlanetListSelected(int position) { ❷
        String planetTitle = PlanetListFragment.mPlanetTitles[position];

        if (findViewById(R.id.details_fragment_container) == null) { ❸
            Intent intent = new Intent(this,
                PlanetDetailsActivity.class);
            intent.putExtra(PlanetDetailsActivity.PLANET_TITLE,
                planetTitle);
            startActivity(intent);
        } else {
            PlanetDetailsFragment fragment =
                PlanetDetailsFragment.newInstance(planetTitle); ❹

            FragmentManager fm = getFragmentManager();
            FragmentTransaction transaction = fm.beginTransaction();
            transaction.replace(R.id.details_fragment_container,
                fragment);
            transaction.commit(); ❺
        }
    }
}
```

❶ ❷ ❸ ❹ ❺ ❻ ❽

MainActivity จะต้อง implement อินเทอร์เฟซ OnPlanetListSelectedListener ❶ ที่เราประกาศไว้ใน PlanetListFragment แล้วระบุคือด้วยกับเมธอด onPlanetListSelected ❷ ตามที่อธิบายก่อนหน้านี้ ซึ่งเมธอด onPlanetListSelected นี้จะถูกเรียกมาจากเมธอด onListItemClick ใน PlanetListFragment อีกที ตอนที่ผู้ใช้เลือกข้อมูลดาวเคราะห์จากลิสต์ภายในเมธอด onPlanetListSelected เราตรวจสอบว่า layout ที่ใช้อยู่ขณะนั้นคือ layout สำหรับจอเล็กหรือใหญ่ ❸ ถ้าหากเป็นจอเล็กจะส่งรัน PlanetDetailsActivity ขึ้นมา โดยส่งชื่อดาวเคราะห์ไปทางอินเทนต์ ❹ แต่หากเป็นจอใหญ่จะสร้างอินสแตนซ์ของ PlanetDetailsFragment ❺ และเพิ่มลงใน FrameLayout ที่อยู่ทางขวาของ layout ❻

9 อย่าลืมประกาศแอ็คทิวิตี้ PlanetDetailsActivity ในไฟล์ AndroidManifest.xml โดยเพิ่มแท็กต่อไปนี้ไว้ก่อนแท็คปิด </application>

#### โปรเจ็ค FragmentDemo, ไฟล์ AndroidManifest.xml

```
<activity android:name=".PlanetDetailsActivity" >
</activity>
```

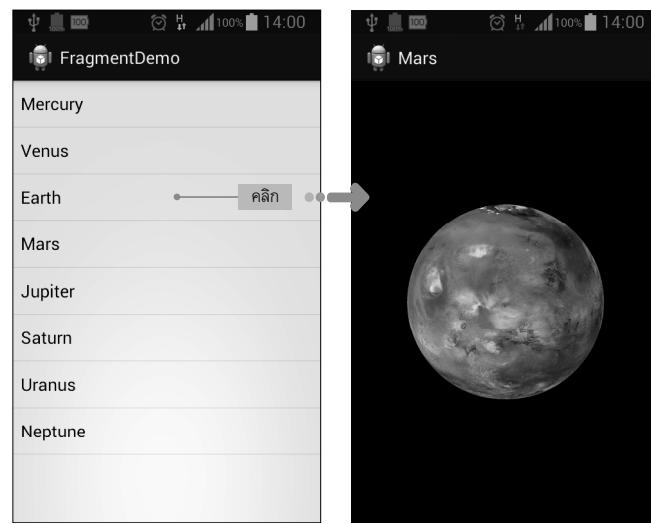
#### NOTE»»

ดาวน์โหลดไฟล์ภาพดาวเคราะห์ได้จาก GitHub repository ที่รวมรวมโปรเจ็คทั้งหมดในหนังสือเล่มนี้ ที่ <https://github.com/3Bugs/Android-Cookbook-Book>

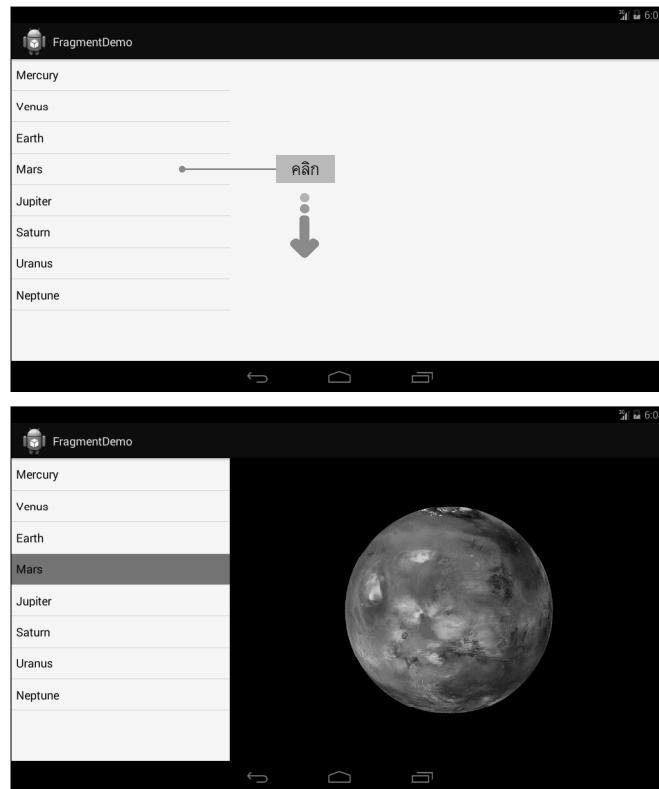
## ผลการรับ

การทดสอบแอพนี้บนอีमูเลเตอร์ คุณจะต้องสร้างอีมูเลเตอร์ 2 เครื่อง เครื่องหนึ่งให้มีขนาดหน้าจอไม่เกิน 4 นิ้ว เช่นระบบเป็น Nexus 4 หรือ 4" WVGA (480 x 800: hdpi) กับอีกเครื่องให้มีขนาดหน้าจอสัก 7 นิ้วขึ้นไป เช่นระบบเป็น Nexus 7 หรือ 7" WSVGA (1024 x 600: mdpi)

- ผลการรันบนหน้าจอ 4 นิ้ว ภาพดาวเคราะห์จะแสดงในแอ็คทิวิตี้ใหม่ (กด Back เพื่อย้อนกลับไปยังหน้าลิสต์)



- ◆ ผลการรันบนหน้าจอ 7 นิ้ว ภาพดาวเคราะห์จะแสดงทางด้านขวาของแอคทิวิตี้เดิม



## ॥ Ferguson ॥ និងការទទួលបានប្រមូល Back

เราจึงสามารถนำแฟร์กเมนต์มาใช้งานในกรณีที่ต้องการเปลี่ยนหน้าจอการทำงานจากหน้าจอหนึ่งไปเป็นอีกหน้าจอหนึ่ง โดยไม่ต้องรันแอคทิวิตี้ใหม่ แต่ใช้การเพิ่มแฟร์กเมนต์ลงใน layout แทนที่แฟร์กเมนต์ที่แสดงผลอยู่ก่อน

นอกจากนี้หากเราต้องการให้ผู้ใช้สามารถกดปุ่ม Back ย้อนกลับไปยังหน้าจอ (แฟร์กเม้นต์) ก่อนหน้าที่สามารถทำได้ โดยเรียกเมธอด addBackStack บนออบเจ็ค FragmentTransaction ก่อนสั่ง commit เพื่อบันทึกการดำเนินการเกี่ยวกับแฟร์กเม้นต์ (fragment transaction) ที่เกิดขึ้นลงใน back stack หลังจากนั้นเมื่อผู้ใช้กดปุ่ม Back แอนดรอยด์จะดำเนินการย้อนกลับให้เอง เช่น ถ้าเราเพิ่มแฟร์กเม้นต์ B ลงไปแทนที่แฟร์กเม้นต์ A, เมื่อผู้ใช้กด Back แอนดรอยด์จะลบแฟร์กเม้นต์ B ออกแล้วโหลดแฟร์กเม้นต์ A ให้ใหม่ เป็นต้น

ព័ត៌មានបច្ចុប្បន្ន

ตัวอย่างนี้จะกำหนดให้หน้าจอเป็น `FrameLayout` ที่เตรียมไว้บรรจุแฟร์กเม้นต์ โดยเมื่อเลือกคำสั่ง `Add Fragment` จากเมนู (บน action bar) เราจะสร้างอินสแตนซ์ของ `CountingFragment` แล้วพิมลงใน `FrameLayout` แทนที่แฟร์กเม้นต์เดิม และหากกดปุ่ม `Back` จะย้อนกลับไปยังแฟร์กเม้นต์ก่อนหน้านั้นที่ลีบแฟร์กเม้นต์ตามลำดับ

## 1 กำหนด layout ของหน้าจอหลัก

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

## 2 เขียนโค้ดใน MainActivity

## โปรเจ็ค BackStackDemo, ไฟล์ MainActivity.java

```
package com.example.backstackdemo;

import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentManager;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends Activity {

    private static final int MENU_ADD_FRAGMENT = 1;
```

```
/* หมายเหตุ Fragment ที่เรากำหนดขึ้นเอง ซึ่งเป็นเลขจำนวนเต็มที่เพิ่มขึ้นเรื่อยๆ
   ในแต่ละครั้งที่อินสแตนซ์ของ CountingFragment ถูกสร้างขึ้น */
private int mCount;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // สร้างคำสั่ง Add Fragment ในเมนู
    MenuItem item = menu.add(1, MENU_ADD_FRAGMENT, 0, "Add Fragment");
    // แสดงคำสั่งนี้บน action bar ถ้ามีที่ว่างพอ
    item.setShowAsAction(MenuItem.SHOW_AS_ACTION_IF_ROOM);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        // เมื่อเลือกคำสั่ง Add Fragment จากเมนู
        case MENU_ADD_FRAGMENT:
            addFragmentToStack();
            return true;
        default:
            return true;
    }
}

void addFragmentToStack() {
    mCount++;
    // สร้างอินสแตนซ์ของ CountingFragment โดยลั่ง mCount ไปเป็นพารามิเตอร์
    Fragment fragment = CountingFragment.newInstance(mCount);

    FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
    // เพิ่มแฟร์กเม้นต์ที่เพิ่งสร้างลงใน FrameLayout
    ft.replace(R.id.fragment_container, fragment);
    // กำหนดโอนเมื่อขึ้นในการแสดงแฟร์กเม้นต์ใหม่
    ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN);
    /* บันทึกการดำเนินการเกี่ยวกับแฟร์กเม้นต์ลงใน back stack เพื่อให้สามารถใช้ปุ่ม Back
       ย้อนกลับไปยังแฟร์กเม้นต์ก่อนหน้าได้ */
    ft.addToBackStack(null);
    ft.commit();
}
}
```

### 3 สร้างคลาสใหม่ชื่อ CountingFragment

ไปริจีค BackStackDemo, ไฟล์ CountingFragment.java

```
package com.example.backstackdemo;

import java.util.Random;

import android.app.Fragment;
import android.graphics.Color;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

public class CountingFragment extends Fragment {

    private static final String ARG_NUM = "num";
    private static final String ARG_BACKGROUND_COLOR = "background_color";

    private int mNum;
    private int mBackgroundColor;

    public static CountingFragment newInstance(int num) {
        CountingFragment fragment = new CountingFragment();

        Bundle args = new Bundle();
        // เก็บหมายเลข (ค่า mCount ที่ส่งมาจาก MainActivity) เป็นอาร์กิวเมนต์ของเฟρกเมνต์
        args.putInt(ARG_NUM, num);

        // สุ่มค่าสีที่จะกำหนดเป็นพื้นหลังของเฟρกเมνต์
        Random rnd = new Random();
        int randomColor = Color.argb(255, rnd.nextInt(256),
                                     rnd.nextInt(256), rnd.nextInt(256));
        // เก็บค่าสีเป็นอาร์กิวเมนต์ของเฟρกเมනต์
        args.putInt(ARG_BACKGROUND_COLOR, randomColor);

        fragment.setArguments(args);

        return fragment;
    }
}
```

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // อ่านหมายเลขและค่าสีจากอาร์กิวเมนต์ มากำหนดให้ตัวแปร (ฟิลด์) ในคลาส
    mNum = getArguments().getInt(ARG_NUM);
    mBackgroundColor = getArguments().getInt(ARG_BACKGROUND_COLOR);
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
    View layout = inflater.inflate(R.layout.fragment_counting,
            container, false);
    layout.setBackgroundColor(mBackgroundColor);
    TextView tv = (TextView) layout.findViewById(R.id.text);
    tv.setText("Fragment #" + mNum);

    return layout;
}
}
```

- 4 สร้าง layout file ชื่อ fragment\_couting.xml เพื่อกำหนดเป็น layout ของ CountingFragment

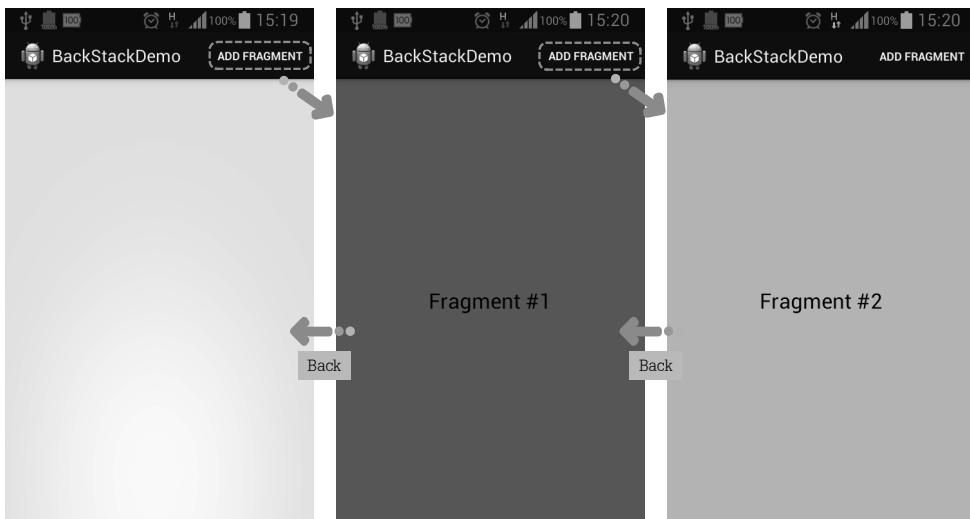
**โปรเจ็ค BackStackDemo, ไฟล์ fragment\_\_counting.java**

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

    <TextView
        android:id="@+id/text"
        style="@android:style/TextAppearance.Large "
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="16dp" />

</LinearLayout>
```

## ผลการรัน



## การสร้าง Navigation Drawer

Navigation drawer คือแบบเมนูที่เลื่อนออกมายังด้านซ้ายของหน้าจอ (ปกติคือด้านซ้าย) เมื่อเราใช้นิ้วปัด (swipe) จากด้านนั้นเข้ามาหรือแตะไอคอนของแอปบน action bar การใช้แบบเมนูชนิดนี้ช่วยให้แอปของเรามีพื้นที่แสดงผลมากขึ้น เพราะโดยปกติแบบเมนูจะถูกซ่อนอยู่เมื่อได้ถูกเรียกใช้งาน

### ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะสร้างแอปที่ให้ข้อมูลเกี่ยวกับดาวเคราะห์ เช่นเดียวกับโปรเจ็ค FragmentDemo โดยแบบ drawer ที่ซ่อนอยู่ทางซ้ายจะแสดงรายชื่อดาวเคราะห์ทั้งหมด และเมื่อเลือกชื่อดาวเคราะห์หนึ่งๆ ก็จะแสดงรูปภาพดาวเคราะห์นั้นในพื้นที่หลักของหน้าจอ

#### 1 กำหนด layout

##### โปรเจ็ค NavigationDrawerDemo, ไฟล์ activity\_main.xml

```
<android.support.v4.widget.DrawerLayout ①
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <!-- พื้นที่หลัก --&gt;
    &lt;FrameLayout ②
        android:id="@+id/content_frame"</pre>

```

```

        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <!-- ແນວ drawer -->
    <ListView ❸
        android:id="@+id/left_drawer"
        android:layout_width="160dp"
        android:layout_height="match_parent"
        android:layout_gravity="start" ❹
        android:background="#ccffff"
        android:choiceMode="singleChoice"
        android:divider="@android:color/transparent"
        android:dividerHeight="0dp" />

</android.support.v4.widget.DrawerLayout>

```

การสร้าง navigation drawer จะต้องกำหนด DrawerLayout ❶ เป็น root view ของ layout 急剧นั้นภายใน DrawerLayout ให้ระบุว่าที่เป็นพื้นที่หลักของแอป (วิวที่ต้องการแสดงเมื่อແນບ drawer ถูกชื่อน้อย) ในที่นี้คือ FrameLayout ❷ ซึ่งเตรียมไว้สำหรับเมนูที่แสดงภาพดาวเคราะห์ ในช่วง runtime, ถ้ามาจึงระบุว่าที่ทำหน้าที่เป็นແນບ drawer ซึ่งนิยมใช้ ListView ❸

การกำหนดแอ็ตทริบิวต์ layout\_gravity ของແນບ drawer เป็นค่า start ❹ จะทำให้ແນບ drawer ปรากฏอยู่จากด้านซ้ายของหน้าจอ ยกเว้นว่าภาษาที่ใช้ในเครื่องนั้นเป็นภาษาที่เขียนจากขวาไปซ้าย ก็จะปรากฏແນບ drawer ทางด้านขวาแทน (แต่หากกำหนดเป็นค่า left จะทำให้ແນບ drawer ปรากฏมาทางด้านซ้ายเสมอ)

#### NOTE»»

- วิวที่เป็นพื้นที่หลักต้องถูกระบุเป็นวิวแรกภายใน DrawerLayout เสมอ เนื่องจาก XML มีการจัดลำดับแบบ z-ordering (วิวที่ถูกระบุที่หลังจะอยู่ข้างบนวิวที่ถูกระบุก่อน) และแอนดรอยด์กำหนดว่า ແນບ drawer จะต้องอยู่ข้างบนพื้นที่หลัก
- โดยปกติจะกำหนดความกว้างและความสูงของพื้นที่หลักเป็น match\_parent เนื่องจากวิวนี้จะใช้พื้นที่หน้าจอทั้งหมดตอนที่ແນບ drawer ถูกชื่อน้อย
- โดยปกติจะกำหนดความสูงของແນບ drawer เป็น match\_parent และกำหนดความกว้างไม่เกิน 320dp เพื่อให้ผู้ใช้สามารถเห็นบางส่วนของพื้นที่หลักตอนที่ແນບ drawer ปรากฏอยู่

#### 2 เพิ่มเติมโค้ดในแอ็คทิวิตี้เป็นดังนี้

##### ໂປຣັ້ງ NavigationDrawerDemo, ໄຟຣ໌ MainActivity.java

```

package com.example.navigationdrawerdemo;

import android.app.Activity;
import android.app.FragmentManager;

```

```
import android.app.FragmentTransaction;
import android.os.Bundle;
import android.support.v4.widget.DrawerLayout;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class MainActivity extends Activity {
    // ชื่อด้าเคราะห์ทั้งหมดในระบบสุริยะ
    protected static final String[] mPlanetTitles = new String[] {
        "Mercury", "Venus", "Earth", "Mars",
        "Jupiter", "Saturn", "Uranus", "Neptune"
    };
    // อ้างอิงไปยัง root view ของ layout
    private DrawerLayout mDrawerLayout;
    // อ้างอิงไปยัง ListView ที่เป็นแกน drawer
    private ListView mDrawerList;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
        mDrawerList = (ListView) findViewById(R.id.left_drawer);

        /* สร้างและกำหนด adapter ให้กับ ListView (แกน drawer)
         * เพื่อแสดงรายชื่อด้าเคราะห์ในลิสต์ */
        mDrawerList.setAdapter(new ArrayAdapter<String>(
            this,
            android.R.layout.simple_list_item_activated_1,
            mPlanetTitles));
    } ❶
    // กำหนด click listener ให้กับ ListView
    mDrawerList.setOnItemClickListener(❷
        new AdapterView.OnItemClickListener() {
            // ระบุการทำงานเมื่อไอเท็ม (ชื่อด้าเคราะห์) ในลิสต์ถูกคลิกเลือก
            @Override
            public void onItemClick(AdapterView<?> parent, View view,
                int position, long id) {
                // โหลดแฟร์กเมนต์ที่แสดงภาพดาวเคราะห์
                String planetTitle = mPlanetTitles[position];
                showPlanetDetails(planetTitle);
            }
        });
}
```

```

        // ไฮไลท์ไอเท็มที่ถูกเลือกในแบบ drawer
        mDrawerList.setItemChecked(position, true);
        // แสดงชื่อดาวเคราะห์บน title bar
        setTitle(mPlanetTitles[position]);
        // ปิด drawer
        mDrawerLayout.closeDrawer(mDrawerList);
    }
}
);

// สร้างอินสแตนซ์ของ PlanetDetailsFragment และเพิ่มลงใน layout
void showPlanetDetails(String planetTitle) {
    PlanetDetailsFragment fragment = PlanetDetailsFragment
        .newInstance(planetTitle);

    FragmentManager fm = getFragmentManager();
    FragmentTransaction transaction = fm.beginTransaction();
    transaction.replace(R.id.content_frame, fragment);
    transaction.commit();
}
}

```

ที่เมื่อรอด `onCreate` เราสร้าง adapter ( `ArrayAdapter`) ที่นำข้อมูลจากอาร์เรย์ `mPlanetTitles` มาแสดงใน `ListView` ❶ ซึ่งทำหน้าที่เป็นแบบ drawer จากนั้นกำหนด click listener ให้กับ `ListView` ❷ โดยเมื่อผู้ใช้เลือกชื่อดาวเคราะห์หนึ่งจากลิสต์ เราจะโหลดแฟร์กเมนต์ที่แสดงภาพดาวเคราะห์มาเพิ่มลงใน layout (คือส่วนนี้แยกไปเป็นเมธอด `showPlanetDetails`), ไฮไลท์ชื่อดาวเคราะห์นั้นในลิสต์, แสดงชื่อดาวเคราะห์บน title bar แล้วปิด drawer กลับลงไปตามเดิม (ถ้าไม่สั่งปิด drawer มันจะแสดงค้างอยู่ร่องนั้นจนกว่าผู้ใช้จะปิดเอง)

- 3 ก็อปปี้ไฟล์ `PlanetDetailsFragment.java`, `fragment_planet_details.xml` และไฟล์รูปภาพดาวเคราะห์ทั้งหมดจากโปรเจ็ค `FragmentDemo` มายังโปรเจ็คนี้

#### TIP >>>

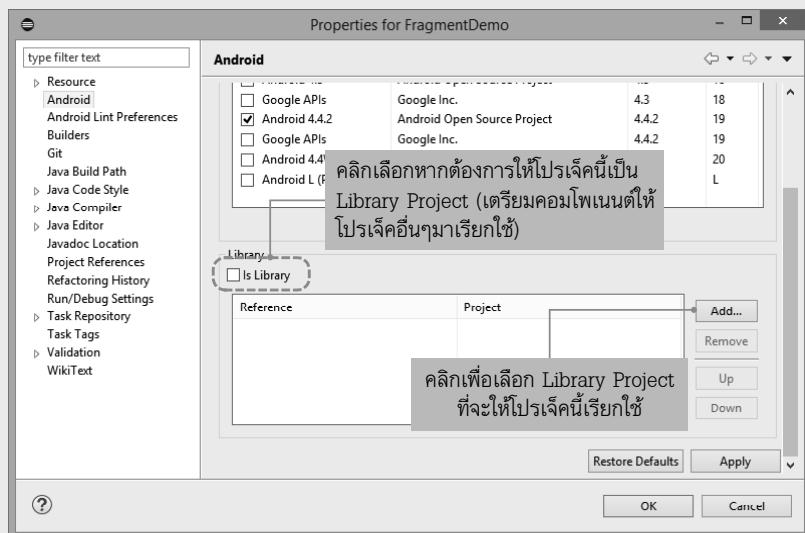
ถ้าหากกำหนดโปรเจ็ค `FragmentDemo` เป็น library project เราจะสามารถเรียกใช้คลาส

`PlanetDetailsFragment` ได้โดยไม่ต้องก็อปปี้ไฟล์มายังโปรเจ็คปัจจุบัน แต่จะให้วิธีเพิ่มการอิมพอร์ตใน `MainActivity` ที่ระบุถึงคลาส `PlanetDetailsFragment` ในโปรเจ็คนั้นแทน เช่น

```
import com.example.fragmentdemo.PlanetDetailsFragment;
```

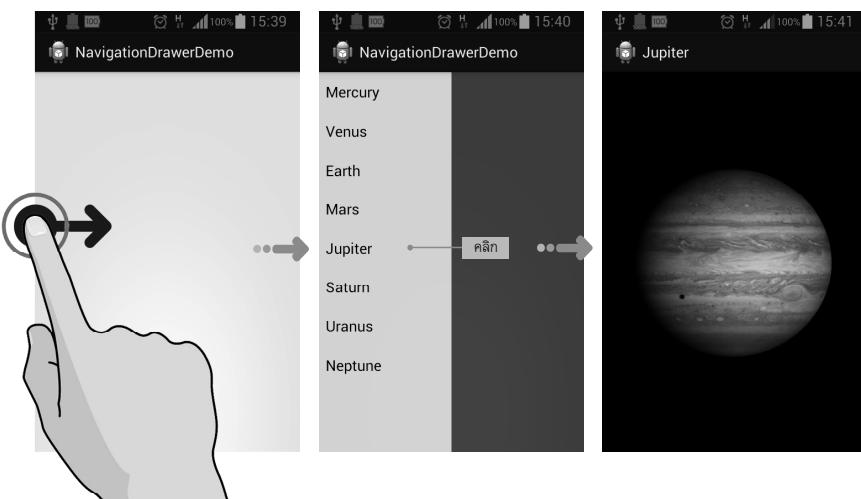
ด้วยวิธีนี้ แฟร์กเมนต์จะเป็น UI module ที่ไม่พึ่งเอาไว้ใช้ในโปรเจ็คเดียวกันเท่านั้น แต่ยังเรียกใช้จากโปรเจ็คอื่นได้ด้วย

อธิบาย การกำหนดโปรเจคเป็น library project ทำได้โดยคลิกขวาที่ชื่อโปรเจค ► Properties ► หัวข้อ Android ► คลิกเลือก Is Library ดังรูป



สำหรับโปรเจคอื่นๆ ที่ต้องการเรียกใช้ชุด component ใน library project จะต้องเพิ่มการอ้างอิงไปยัง library project โดยคลิกขวาที่ชื่อโปรเจค ► Properties ► หัวข้อ Android ► คลิกปุ่ม Add ในกรอบ Library และเลือก library project ที่ต้องการ

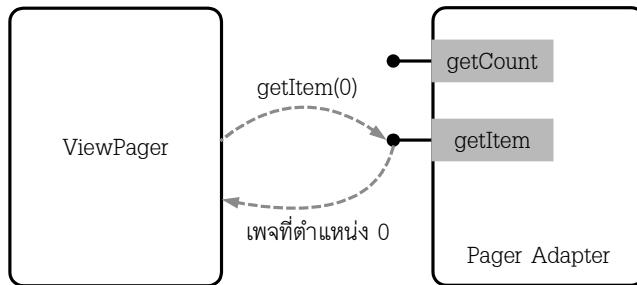
## ผลการรัน



## การสร้าง Swipe Views

Swipe views คือรูปแบบของ UI navigation ซึ่งใช้การปัดนิ้ว (swipe) ไปทางซ้ายหรือขวา เพื่อเลื่อน (navigate) ไปยังหน้าจอต่างๆที่อยู่ในระดับเดียวกัน ( sibling screen) บางครั้งจึงเรียกว่า horizontal paging โดยแต่ละหน้าจอที่แสดงออกมานั้นจะเรียกว่า เพจ (page)

การสร้าง swipe views ในแอนดรอยด์จะใช้ ViewPager ร่วมกับคลาสที่เป็น pager adapter ซึ่งแต่ละเพจที่ ViewPager แสดงออกมานั้นได้มาจากการเรียกเมธอด getItem ของ pager adapter โดยระบุตำแหน่งของเพจที่ต้องการ ดังรูป



พูดง่ายๆว่า pager adapter คือผู้จัดเตรียมเพจต่างๆที่จะแสดงออกมานั้นใน ViewPager นั่นเอง ในตัวอย่างของเรา จะต้องเพจที่แสดงใน ViewPager คือแฟร์กเมนต์ แอนดรอยด์เตรียม pager adapter ที่ใช้ทำงานกับแฟร์กเมนต์ (pager adapter ที่ส่งแฟร์กเมนต์ไปให้ ViewPager) ไว้ 2 คลาสคือ

- ◆ **FragmentPagerAdapter** หมายสำหรับกรณีที่มีจำนวนเพจไม่มาก pager adapter ชนิดนี้จะเก็บเพจ (แฟร์กเมนต์) ทั้งหมดไว้ในหน่วยความจำ
- ◆ **FragmentStatePagerAdapter** หมายสำหรับกรณีที่มีจำนวนเพจมาก เนื่องจาก pager adapter ชนิดนี้จะทำลายแฟร์กเมนต์ที่ไม่ได้ใช้ให้อัตโนมัติขณะที่ถูกใช้แล้วไปยังเพจต่างๆ โดยจะเก็บเฉพาะสถานะ (instance state) ของแฟร์กเมนต์ไว้ เพื่อให้เราเรียกคืนสถานะเหล่านั้นตอนที่แฟร์กเมนต์ถูกสร้างขึ้นมาใหม่ (ตอนที่ผู้ใช้เลื่อนกลับมายังเพจเดิม) การใช้ pager adapter ชนิดนี้จะลดการใช้หน่วยความจำของระบบได้มากเมื่อเทียบกับ

### FragmentPagerAdapter

ในตัวอย่างนี้จะใช้ FragmentStatePagerAdapter

#### NOTE »»

เนื่องจากตัวอย่างนี้ใช้คลาส FragmentStatePagerAdapter ใน support library v13 ของแอนดรอยด์ ดังนั้นคุณจะต้องเพิ่มไฟล์ .jar ของ support library ดังกล่าวเข้ามาในโปรเจ็คก่อน โดยไปที่ไฟล์เดอร์

ไฟล์เดอร์ที่ติดตั้ง ADT หรือ *Android Studio\adt\sdk\extras\android\support\v13*

จากนั้นให้คลิกขวาไฟล์ชื่อ android-support-v13.jar มาปล่อยลงในไฟล์เดอร์ libs ภายใต้โปรเจ็ค

SwipeViewsDemo นั้น

## ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะสร้างแอพที่เราสามารถ swipe หน้าจอไปทางซ้ายหรือขวา เพื่อเลื่อนดูภาพดาวเคราะห์ต่างๆได้

### 1 กำหนด layout

#### โปรเจ็ค SwipeViewsDemo, ไฟล์ activity\_main.xml

```
<android.support.v4.view.ViewPager
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/pager"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

ใน layout มีเพียง ViewPager ที่มีความกว้างและความสูงเต็มจอ เนื่องจากเราต้องการให้แต่ละเพจใน swipe view ใช้พื้นที่หน้าจอทั้งหมด

### 2 เพิ่มเติมโค้ดในแอคทิวิตี้เป็นดังนี้

#### โปรเจ็ค SwipeViewsDemo, ไฟล์ MainActivity.java

```
package com.example.swipeviewsdemo;

import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentManager;
import android.os.Bundle;
import android.support.v13.app.FragmentStatePagerAdapter;
import android.support.v4.view.ViewPager;

public class MainActivity extends Activity {

    // ชื่อดาวเคราะห์ทั้งหมดในระบบสุริยะ
    protected static final String[] mPlanetTitles = new String[] {
        "Mercury", "Venus", "Earth", "Mars",
        "Jupiter", "Saturn", "Uranus", "Neptune"
    };

    PlanetPagerAdapter mAdapter; // pager adapter
    ViewPager mViewPager;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        /* สร้างอุปกรณ์ที่เป็น pager adapter (อินสแตนซ์ของ PlanetPagerAdapter)
           และนำไปกำหนดให้กับ ViewPager */
        mAdapter = new PlanetPagerAdapter(getFragmentManager()); ⑤
        mViewPager = (ViewPager) findViewById(R.id.pager);
        mViewPager.setAdapter(mAdapter); ⑥
        // ยังคงได้ดีอีก...
    }
}
```

```
// pager adapter ซึ่งจะส่งอินสแตนซ์ของ PlanetDetailsFragment ไปให้ ViewPager
public class PlanetPagerAdapter extends FragmentStatePagerAdapter { ❶

    public PlanetPagerAdapter(FragmentManager fm) {
        super(fm);
    }

    @Override
    public Fragment getItem(int i) { ❷
        // สร้างแฟร์กเม้นท์ที่แสดงภาพดาวเคราะห์ แล้วส่งคืนแฟร์กเม้นท์นั้นกลับออกไป
        String planetTitle = mPlanetTitles[i];
        Fragment fragment = PlanetDetailsFragment
            .newInstance(planetTitle);
        return fragment;
    }

    @Override
    public int getCount() { ❸
        return mPlanetTitles.length;
    }

    @Override
    public CharSequence getPageTitle(int position) { ❹
        return mPlanetTitles[position];
    }
}
```

ในแอ็คทิวิตี้ เราสร้างคลาส PlanetPagerAdapter ที่สืบทอดจาก FragmentStatePagerAdapter  
❶ แล้วทำการ override เมธอด getItem ❷ เพื่อส่งอินสแตนซ์ของ PlanetDetailsFragment (แฟร์กเม้นท์ที่แสดงภาพดาวเคราะห์) ออกไป ซึ่งจะถูกaley เป็นเพจใหม่ใน ViewPager นอกเหนือนี้ยังต้อง override เมธอด getCount ❸ เพื่อบอกจำนวนเพจทั้งหมดให้ ViewPager รู้ ในที่นี้กำหนดจำนวนเพจเท่ากับจำนวนสมาชิกของอาร์เรย์ mPlanetTitles

สำหรับเมธอด getPageTitle ❹ นั้นจะ override หรือไม่ก็ได้ ในที่นี้เราส่งชื่อดาวเคราะห์ที่ออกไปเป็น title ของเพจ เพราะอีกสักครู่เราจะเพิ่มโค้ดที่แสดง title ของเพจบน title bar

หลังจากกำหนดคลาสที่เป็น pager adapter เรียบร้อย เราเก็บรังอินสแตนซ์ของคลาสนี้ในเมธอด onCreate ของแอ็คทิวิตี้ ❺ ก่อนนำไปกำหนดให้ ViewPager ❻

### 3 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตี้

#### โปรเจ็ค SwipeViewsDemo, ไฟล์ MainActivity.java

```
ViewPager.OnPageChangeListener pageChangeListener =
    new ViewPager.OnPageChangeListener() { ❶
        @Override
        public void onPageSelected(int position) {
            setTitle(mAdapter.getPageTitle(position)); ❷
        }

        /* สองเมธอดนี้ไม่ได้ใช้ แต่ต้อง override ตามข้อกำหนดของ
         * OnPageChangeListener */
        @Override
        public void onPageScrolled(int position,
                                   float positionOffset, int positionOffsetPixels) { }
        @Override
        public void onPageScrollStateChanged(int state) { }
    };
}
```

โค้ดส่วนนี้จะสร้างอบเจ็ค OnPageChangeListener ❶ เก็บไว้ที่ตัวแปร mPageChangeListener ซึ่งเป็น listener สำหรับบุํโค้ดการทำงานเมื่อมีการเปลี่ยนเพจใน ViewPager ในที่นี้เราอ่าน title ของเพจมาแสดงบน title bar หลังจากเปลี่ยนไปยังเพจใหม่แล้ว ❷

### 4 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตี้

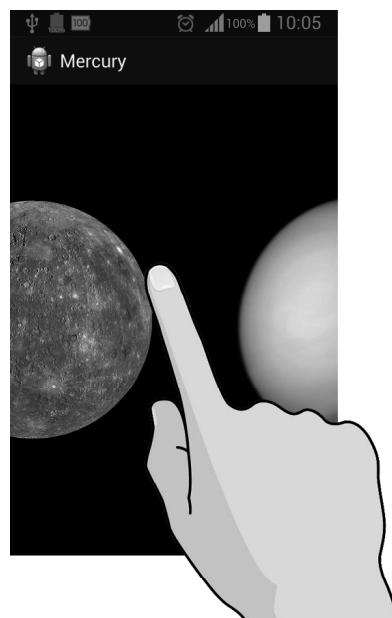
#### โปรเจ็ค SwipeViewsDemo, ไฟล์ MainActivity.java

```
mViewPager.setOnPageChangeListener(pageChangeListener);
pageChangeListener.onPageSelected(0);
```

บรรทัดแรกคือการนำอบเจ็ค OnPageChangeListener เมื่อครู่นี้มากำหนดให้กับ ViewPager ส่วนบรรทัดถัดมาคือการสั่งให้แสดงชื่อดาวเคราะห์แรกบน title bar เพราะตอนเริ่มต้นจะแสดงภาพดาวเคราะห์แรกทันที แต่แอนดรอยด์ไม่ได้เรียกมาบัญเมธอด onPageSelected ใน OnPageChangeListener ทำให้มีชื่อดาวเคราะห์บน title bar เราจึงต้องเรียกเมธอดนี้เองตอนเริ่มต้น

### 5 ก็อบปี้ไฟล์ PlanetDetailsFragment.java, fragment\_planet\_details.xml และไฟล์รูปภาพดาวเคราะห์ทั้งหมดจากโปรเจ็ค FragmentDemo มายังโปรเจ็คนี้

## ผลการรัน



### การแสดง Title ของเพจโดยใช้ PagerTitleStrip

เราอาจใช้ PagerTitleStrip ในการแสดง title ของเพจแทนการแสดง title บน title bar ได้ ซึ่งเมื่อตีคือ PagerTitleStrip จะแสดง title ของเพจที่อยู่ติดกันด้วย ทำให้ผู้ใช้รู้ว่าสามารถ swipe ไปยังเพจอื่นๆ ได้อีก

การใช้งาน PagerTitleStrip สามารถทำได้ง่ายๆ โดยระบุ PagerTitleStrip เป็น child ของ ViewPager ใน layout file ดังตัวอย่าง

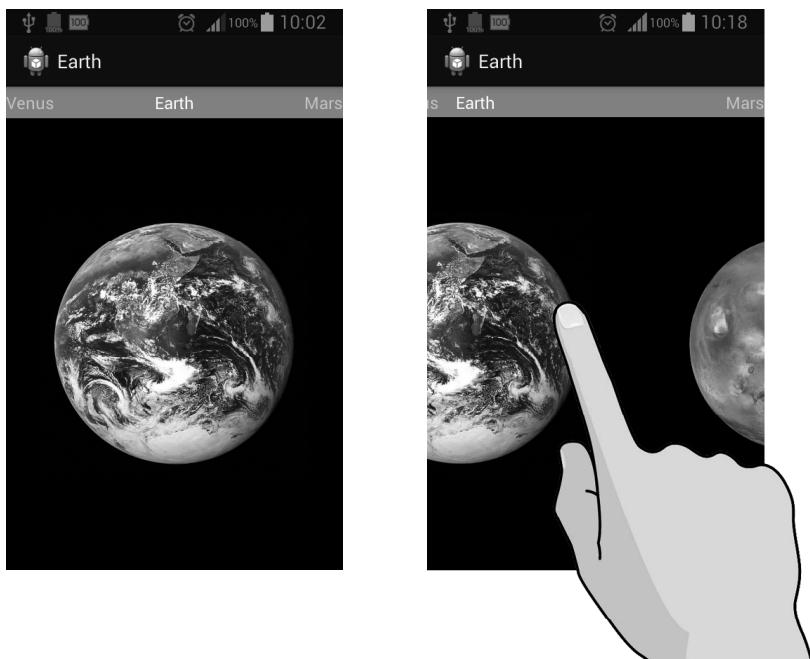
ไปรษณีย์ SwipeViewsDemo, ไฟล์ activity\_main.xml

```
<android.support.v4.view.ViewPager
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/pager"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <android.support.v4.view.PagerTitleStrip
        android:id="@+id/pager_title_strip"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="top" ❶
        android:background="#33b5e5"
        android:paddingBottom="4dp"
        android:paddingTop="4dp"
        android:textColor="#fff" />

</android.support.v4.view.ViewPager>
```

ในที่นี่เราวาง PagerTitleStrip ชิดด้านบนของ ViewPager โดยกำหนด layout\_gravity ของ PagerTitleStrip เป็นค่า top ❶ ซึ่งจะได้ผลลัพธ์ดังรูป



## การสร้าง ActionBar Tabs

แท็บ (tab) คือ UI อิกรูปแบบหนึ่งที่ใช้ navigate ระหว่างหน้าจอต่างๆ ที่อยู่ระดับเดียวกัน

แท็บที่จะอธิบายในหัวข้อนี้คือส่วนหนึ่งของ ActionBar การทำงานหลายๆอย่างเกี่ยวกับแท็บ จึงต้องใช้คลาสหรือเมธอดที่กำหนดไว้ในคลาส ActionBar

### ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะสร้างแอพซึ่งประกอบด้วยแท็บต่างๆ ที่จะแสดงภาพดาวเคราะห์แต่ละดวงออกมายังหน้าจอ

#### 1 กำหนด layout

โปรเจ็ค TabsDemo, ไฟล์ activity\_main.xml

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

ใน layout จะมีแค่เพียง FrameLayout ที่เตรียมไว้สำหรับแท็บของ PlanetDetailsFragment ในช่วง runtime

- 2 เพิ่มเติมโค้ดในแอคทิวิตี้เป็นดังนี้

โปรเจ็ค TabsDemo, ไฟล์ MainActivity.java

```
package com.example.tabsdemo;

import android.app.ActionBar;
import android.app.ActionBar.Tab;
import android.app.Activity;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.os.Bundle;

public class MainActivity extends Activity {

    // ชื่อดาเคราะห์ทั้งหมดในระบบสุริยะ
    protected static final String[] mPlanetTitles = new String[] {
        "Mercury", "Venus", "Earth", "Mars",
        "Jupiter", "Saturn", "Uranus", "Neptune"
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        final ActionBar actionBar = getActionBar();
        actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS); ❶

        // ยังไม่ได้อีก...
    }

    // สร้างอินสแตนซ์ของ PlanetDetailsFragment และเพิ่มลงใน layout
    void showPlanetDetails(String planetTitle) { ❷
        PlanetDetailsFragment fragment = PlanetDetailsFragment
            .newInstance(planetTitle);

        FragmentManager fm = getFragmentManager();
        FragmentTransaction transaction = fm.beginTransaction();
        transaction.replace(R.id.fragment_container, fragment);
        transaction.commit();
    }
}
```

เมื่อต้องการสร้างแท็บ สิ่งแรกที่ต้องทำคือการกำหนด navigation mode ของ ActionBar เป็น NAVIGATION\_MODE\_TABS ①

สำหรับเมธอด showPlanetDetails ② จะเหมือนในตัวอย่างอื่นๆที่ผ่านมา คือทำหน้าที่สร้างและเพิ่มแฟร์มเมนต์ที่แสดงภาพดาวเคราะห์ลงใน layout เราจะเรียกใช้เมธอดนี้ตอนที่ผู้ใช้เลือกแท็บหนึ่งๆ ดัง코드ในขั้นตอนถัดไป

- 3 เพิ่มโค้ดในเมธอด onCreate เพื่อเตรียม listener ที่จะนำไปกำหนดให้กับแท็บแต่ละแท็บ

#### โปรเจ็ค TabsDemo, ไฟล์ MainActivity.java

```
// ອອນເຈັດ TabListener ຄໍາຫວັບຮາງໂດຍດາກທຳກຳມີກົດລືບຕ່າງໆເກີດວິເວນດີຕ່າງໆໄດ້ຢັກກັບແທັບ
ActionBar.TabListener tabListener = new ActionBar.TabListener() {
    @Override
    public void onTabSelected(ActionBar.Tab tab, FragmentTransaction ft) {
        // ໂພດແພຣມເນັດທີ່ແສດງກາພດາວເຄຣະໜີ
        String planetTitle = mPlanetTitles[tab.getPosition()];
        showPlanetDetails(planetTitle);
    }

    // ສອງມີມີດນີ້ໄດ້ໃຊ້ ແຕ່ຕ້ອງ override ດາມຂໍ້ກຳນົດຂອງ TabListener
    @Override
    public void onTabUnselected(ActionBar.Tab tab,
                               FragmentTransaction ft) { }
    @Override
    public void onTabReselected(ActionBar.Tab tab,
                               FragmentTransaction ft) { }
};
```

- 4 เพิ่มโค้ดในเมธอด onCreate เพื่อสร้างแท็บ, กำหนดชื่อความและ listener ให้กับแท็บ แล้วเพิ่มแท็บลงใน ActionBar โดยจำนวนแท็บที่สร้างจะขึ้นอยู่กับจำนวนข้อมูลในอาร์เรย์ mPlanetTitles (ในที่นี้คือ 8 แท็บ)

#### โปรเจ็ค TabsDemo, ไฟล์ MainActivity.java

```
// ສ້າງແທັບຕາມຈຳນວນຂໍ້ມູນໃນອົບເຮົາ mPlanetTitles
for (int i = 0; i < mPlanetTitles.length; i++) {
    // ສ້າງແທັບໃໝ່ (ອອນເຈັດ ActionBar.Tab)
    Tab tab = actionBar.newTab();
    // ກຳນົດຊ່ອງມີກົດລືບຕ່າງໆໃຫ້ກັບ Tab
    tab.setText(mPlanetTitles[i]);
    tab.setTabListener(tabListener);

    // ເພີ່ມແທັບລົງໃນ ActionBar
    actionBar.addTab(tab);
}
```

**TIP »»**

เนื่องจากเมธอด setText และ setTabListener ของคลาส ActionBar.Tab จะส่งคืน reference ของแท็บนั้นๆกลับมาให้ ดังนั้นโค้ดในลูป for ข้างบนจึงอาจยุบเป็นประโยชน์อย่างคำสั่งเดียวได้ดังนี้

```
    actionBar.addTab(actionBar.newTab()
        .setText(mPlanetTitles[i])
        .setTabListener(tabListener)
    );
```

- 5 ก็อปปี้ไฟล์ PlanetDetailsFragment.java, fragment\_planet\_details.xml และไฟล์รูปภาพดาวเคราะห์ทั้งหมดจากโปรเจ็ค FragmentDemo มาอีกโปรเจ็คนี้

**ผลการรัน**

## การใช้ Swipe Views ร่วมกับ Tabs

การใช้ swipe views ร่วมกับ ActionBar tabs จะเพิ่มความสะดวกให้ผู้ใช้ในการ navigate ไปยังเพจต่างๆ เพราะสามารถ swipe เลื่อนไปที่ลักษณะ หรือคลิกแท็บเพื่อเข้าถึงเพจที่ต้องการได้ทันที

### ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะนำโค้ดจากโปรเจ็ค SwipeViewsDemo และ TabsDemo มาประกอบกัน โดยแก้ไขเพิ่มเติมบางจุดเพื่อให้การทำงานของ ViewPager สอดคล้องกับแท็บ

1 กำหนด layout ให้มีเพียง ViewPager อياงเดียว (ไม่ต้องมี PagerTitleStrip)

โปรเจ็ค SwipeWithTabsDemo, ไฟล์ activity\_main.xml

```
<android.support.v4.view.ViewPager
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/pager"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

2 เพิ่มโค้ดในแอคทิวิตี้

โปรเจ็ค SwipeWithTabsDemo, ไฟล์ MainActivity.java

```
package com.example.swipewithtabsdemo;

import android.app.ActionBar;
import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.app.ActionBar.Tab;
import android.os.Bundle;
import android.support.v13.app.FragmentStatePagerAdapter;
import android.support.v4.view.ViewPager;

public class MainActivity extends Activity {
    // ชื่อดาวเคราะห์ทั้งหมดในระบบสุริยะ
    protected static final String[] mPlanetTitles = new String[] {
        "Mercury", "Venus", "Earth", "Mars",
        "Jupiter", "Saturn", "Uranus", "Neptune"
    };

    PlanetPagerAdapter mAdapter; // pager adapter
    ViewPager mViewPager;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        setupViewPager();
        setupTabs();
    }

    // เมธอด setupViewPager
    // คลาส PlanetPagerAdapter
    // เมธอด setupTabs
}
```

เราจะนำโค้ดที่ใช้เตรียม ViewPager จากโปรเจ็ค SwipeViewsDemo มาสร้างเป็นเมธอด setupViewPager และนำโค้ดที่ใช้เตรียมแท็บจากโปรเจ็ค TabsDemo มาสร้างเป็นเมธอด setupTabs แล้วจึงเรียกใช้เมธอดทั้งสองนี้ใน onCreate

### 3 เพิ่มเมธอด setupViewPager และคลาส PlanetPagerAdapter ในแอคทิวิตี้

#### โปรเจ็ค SwipeWithTabsDemo, ไฟล์ MainActivity.java

```
private void setupViewPager() {
    /* สร้างออบเจ็คที่เป็น pager adapter (อินสแตนซ์ของ PlanetPagerAdapter)
       แล้วนำไปกำหนดให้กับ ViewPager */
    mAdapter = new PlanetPagerAdapter(getFragmentManager());
    mViewPager = (ViewPager) findViewById(R.id.pager);
    mViewPager.setAdapter(mAdapter);

    /* ອອນເຈັດ OnPageChangeListener ສໍາຫລວບຮູບໂຄດກາທຳງານເນື້ອມືກາເປີຍແພິໃນ
       ViewPager */
    ViewPager.OnPageChangeListener pageChangeListener =
        new ViewPager.OnPageChangeListener() {
            @Override
            public void onPageSelected(int position) {
                // ເນື້ອເລືອນໄປຢັງເພຈຕ່າງໆ ໃຫ້ເລືອກແທັບທີ່ສ້າມພັນນີ້ກັນ
                getActionBar().setSelectedNavigationItem(position); ❶
            }

            /* ສອນເນັດນີ້ໄມ້ໄດ້ໃໝ່ ແຕ່ຕ້ອງ override ຕາມຂ້ອກການດຂອງ
               OnPageChangeListener */
            @Override
            public void onPageScrolled(int position,
                                       float positionOffset, int positionOffsetPixels) { }
            @Override
            public void onPageScrollStateChanged(int state) { }
        };

    // ກຳນົດອອນເຈັດ OnPageChangeListener ໃຫ້ກັນ ViewPager
    mViewPager.setOnPageChangeListener(pageChangeListener);
}

// pager adapter ທີ່ຈະສ່າງອິນສແນ່ນຂອງ PlanetDetailsFragment ໄປໃຫ້ ViewPager
public class PlanetPagerAdapter extends FragmentStatePagerAdapter {

    public PlanetPagerAdapter(FragmentManager fm) {
        super(fm);
    }

    @Override
    public Fragment getItem(int i) {
```

```
// สร้างแฟร์กเมนต์ที่แสดงภาพดาวเคราะห์ แล้วส่งคืนแฟร์กเมนต์นั้นกลับออกไป
String planetTitle = mPlanetTitles[i];
Fragment fragment = PlanetDetailsFragment.newInstance(planetTitle);
return fragment;
}

@Override
public int getCount() {
    return mPlanetTitles.length;
}

@Override
public CharSequence getPageTitle(int position) {
    return mPlanetTitles[position];
}
}
```

จุดเดียวกับที่แตกด้วยมือในหน้าจอในตอนที่เราต้องกำหนดให้แท็บที่สัมพันธ์กับเพจนั้นๆ ถูกเลือก ❶ จากเดิมที่จะแสดง title ของเพจ (ชื่อดาวเคราะห์) บน title bar

#### 4 เพิ่มเมธอด setupTabs ในแอ็คทิวิตี้

##### โปรเจ็ค SwipeWithTabsDemo, ไฟล์ MainActivity.java

```
private void setupTabs() {
    final ActionBar actionBar = getActionBar();
    actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);

    // ขอบเจ็ค TabListener สำหรับบุํคิดการทำงานเมื่อกดอีวีเอนต์ต่างๆ กีวยังกับแท็บ
    ActionBar.TabListener tabListener = new ActionBar.TabListener() {
        @Override
        public void onTabSelected(ActionBar.Tab tab,
                                 FragmentTransaction ft) {
            // เมื่อเลือกแท็บจะกำหนดให้ ViewPager แสดงเพจที่สัมพันธ์กัน
            mViewPager.setCurrentItem(tab.getPosition()); ❶
        }

        // สองเมธอดนี้ไม่ได้ใช้ แต่ต้อง override ตามข้อกำหนดของ TabListener
        @Override
        public void onTabUnselected(ActionBar.Tab tab,
                                   FragmentTransaction ft) { }
        @Override
        public void onTabReselected(ActionBar.Tab tab,
                                   FragmentTransaction ft) { }
    };

    // สร้างแท็บตามจำนวนข้อมูลในอาร์เรย์ mPlanetTitles
    for (int i = 0; i < mPlanetTitles.length; i++) {
```

```
// สร้างแท็บใหม่ (ออบเจ็ค ActionBar.Tab)
Tab tab = actionBar.newTab();
// กำหนดชื่อความและ Listener ให้กับแท็บ
tab.setText(mPlanetTitles[i]);
tab.setTabListener(tabListener);

// เพิ่มแท็บลงใน ActionBar
actionBar.addTab(tab);
}
}
```

จุดที่แตกต่างจากโปรเจ็ค TabsDemo ก็คือ เมื่อผู้ใช้เลือกแท็บหนึ่งๆ เราไม่ต้องโหลด PlanetDetailsFragment เอง แต่จะสั่งให้ ViewPager แสดงเพจที่สัมพันธ์กับแท็บนั้นокما ❶ เพราะตอนนี้การโหลด PlanetDetailsFragment เป็นหน้าที่ของ pager adapter ที่ผูกกับ ViewPager ไปแล้ว

- 5 ก็อปปี้ไฟล์ PlanetDetailsFragment.java, fragment\_planet\_details.xml และไฟล์รูปภาพดาวเคราะห์ทั้งหมดจากโปรเจ็ค FragmentDemo นายังโปรเจ็คนี้

## ผลการรัน

