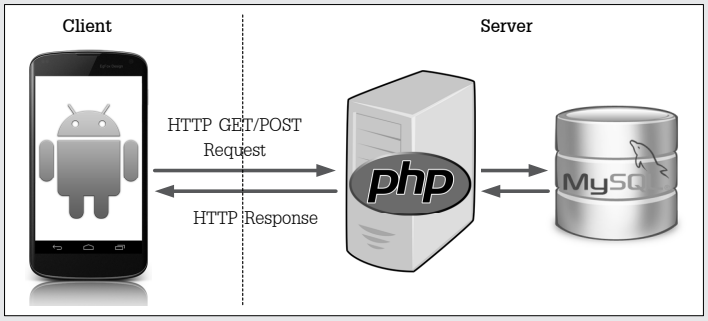
CHAPTER

12 Preference, ไฟล์ และฐานข้อมูล

เนื้อหาในบทนี้

- การเก็บข้อมูลด้วย SharedPreferences API
- การสร้างหน้าตัวเลือกโดยใช้ Preference API
- การอ่าน/เขียนไฟล์บน Internal Storage
- การอ่าน/เขียนไฟล์บน External Storage
- การทำงานกับฐานข้อมูล SQLite
- การติดต่อฐานข้อมูล MySQL บนเซิร์ฟเวอร์



การเก็บข้อมูลด้วย SharedPreferences API

วิธีการเก็บข้อมูลที่จะอธิบายในบทนี้ เป็นวิธีที่ทำให้เราอ่านข้อมูลกลับมาภายหลังได้แม้ว่าแอพของเราจะถูกปิดหรือเครื่องถูกปิดหรือรีสตาร์ท ต่างจากการเก็บข้อมูลในตัวแปรซึ่งข้อมูลจะหายไปเมื่อปิดแอพหรือปิด/รีสตาร์ทเครื่อง

ถ้าหากคุณมีข้อมูลในรูปแบบคีย์/ค่า (key/value) ยกตัวอย่างเช่น ค่าตัวเลือกต่างๆที่ผู้ใช้กำหนด หรือสถานะของแอพที่คุณต้องการจดจำไว้ วิธีที่ง่ายที่สุดในการเก็บข้อมูลเหล่านี้ก็คือการใช้ SharedPreferences API ซึ่งข้อมูลจะถูกเก็บไว้ในไฟล์ที่เรียกว่า shared preference file และมีเมธอดไว้สำหรับอ่าน/เขียนข้อมูลโดยที่เราไม่ต้องเข้าถึงไฟล์เองโดยตรง

ขั้นตอนการใช้งาน SharedPreferences API

1 เข้าถึงออบเจ็ค `SharedPreferences` โดยใช้เมธอดใดเมธอดหนึ่งต่อไปนี้

- **เมธอด getPreferences** ให้เรียกเมธอดนี้จากแอคทิวิตี ถ้าหากคุณต้องการใช้ shared preference file เพียงไฟล์เดียว (แต่สามารถเก็บข้อมูลแบบคีย์/ค่าได้หลายชุด) เมื่อใช้เมธอดนี้คุณไม่ต้องระบุชื่อไฟล์ เพราะแอนดรอยด์จะอ่าน/เขียนข้อมูลลงในไฟล์ที่เป็นดีฟอลต์ของแอคทิวิตีนั้นๆ

```
SharedPreferences sharedPref = getPreferences(Context.MODE_PRIVATE);
```

- **เมธอด getSharedPreferences** ให้เรียกเมธอดนี้จากคอนเท็กซ์ใดๆในแอพ (เช่น แอคทิวิตี หรือเซอร์วิส) ถ้าหากคุณต้องการใช้ shared preference file มากกว่า 1 ไฟล์ ซึ่งจะต้องระบุชื่อไฟล์เป็นพารามิเตอร์ตัวแรกของเมธอด

```
SharedPreferences sharedPref = คอนเท็กซ์.getSharedPreferences(
        ชื่อไฟล์, Context.MODE_PRIVATE);
```

สำหรับ `MODE_PRIVATE` คือการเปิด shared preference file ในโหมด Private ทำให้การเข้าถึงไฟล์ทำได้จากแอพของเราเท่านั้น แต่หากระบุ `Context.MODE_WORLD_READABLE` หรือ `Context.MODE_WORLD_WRITABLE` จะทำให้เข้าถึงจากแอพอื่นๆได้ด้วย (ถ้ารู้ชื่อไฟล์)

การตั้งชื่อไฟล์ แนะนำให้ตั้งในรูปแบบ *ชื่อแพคเกจ.ชื่อไฟล์* เช่น `com.example.myapp.mypref`

2 เมื่อต้องการเก็บ (เขียน) ข้อมูล ให้เรียกเมธอด `edit` บนออบเจ็ค `SharedPreferences` เพื่อสร้าง `SharedPreferences.Editor` ขึ้นมา แล้วจึงเรียกเมธอด `putString`, `putInt`, `putLong`, `putFloat` หรือ `putBoolean` เพื่อเขียนข้อมูลลงใน shared preference file และสุดท้ายให้เรียกเมธอด `commit` เพื่อบันทึกการเปลี่ยนแปลง

```
SharedPreferences.Editor editor = sharedPref.edit();
editor.putInt(คีย์, ค่า);
...
editor.commit();
```

3 เมื่อต้องการอ่านข้อมูล ให้เรียกเมธอด getString, getInt, getLong, getFloat, getBoolean หรือ getAll บนออบเจ็ค SharedPreferences

```
ตัวแปร = sharedPref.getInt(คีย์, ค่าดีฟอลต์);
```

ค่าดีฟอลต์ที่ระบุเป็นพารามิเตอร์ตัวที่สอง คือค่าที่เมธอดจะส่งคืนกลับมาถ้าหากไม่มีข้อมูลตามคีย์ที่ระบุด้วยพารามิเตอร์ตัวแรก

4 การลบข้อมูลจะมีขั้นตอนคล้ายกับการเขียนข้อมูล

```
SharedPreferences.Editor editor = sharedPref.edit();
editor.remove(คีย์); // ลบข้อมูลที่มีคีย์ตามที่ระบุ
editor.commit();
```

5 การลบข้อมูลทั้งหมดก็ทำได้ในลักษณะเดียวกัน

```
SharedPreferences.Editor editor = sharedPref.edit();
editor.clear();   // ลบข้อมูลทั้งหมดใน shared preference file นั้น
editor.commit();
```

NOTE

ให้สังเกตว่าการเขียนและลบข้อมูลจะใช้เมธอดบนออบเจ็ค SharedPreferences.Editor ในขณะที่การอ่านข้อมูลจะใช้เมธอดบนออบเจ็ค SharedPreferences

ตัวอย่าง

ตัวอย่างนี้จะแสดงการอ่านและเขียนข้อมูลโดยใช้ SharedPrerences API

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค SharedPreferencesDemo, ไฟล์ 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="vertical"
```

```
android:padding="16dp" >

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Name:" />

<EditText
    android:id="@+id/name"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:ems="10" >

    <requestFocus />
</EditText>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Age:" />

<EditText
    android:id="@+id/age"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:ems="10"
    android:inputType="number" />

<RadioGroup
    android:id="@+id/sex_radio_group"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >

    <RadioButton
        android:id="@+id/male_radio"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Male" />

    <RadioButton
        android:id="@+id/female_radio"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Female" />
</RadioGroup>
```

```
    <Button
        android:id="@+id/write_data_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="Write Data" />

    <Button
        android:id="@+id/read_data_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Read Data" />

</LinearLayout>
```

2 ที่แอคทิวิตี ให้เพิ่มค่าคงที่ระดับคลาส

โปรเจ็ค SharedPreferencesDemo, ไฟล์ MainActivity.java

```
private static final int MALE = 1;
private static final int FEMALE = 2;
```

3 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตี

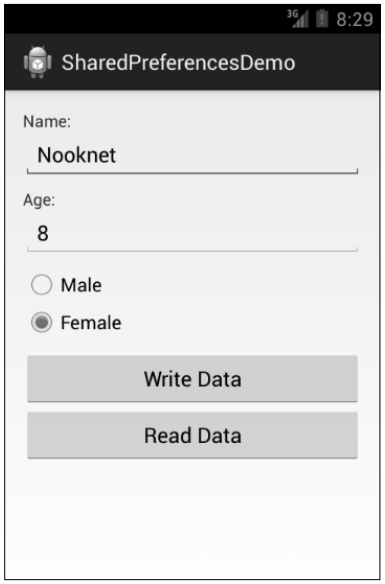
โปรเจ็ค SharedPreferencesDemo, ไฟล์ MainActivity.java

```
// เปิด shared preference file ที่เป็นดีฟอลต์ของแอคทิวิตีในโหมด Private
final SharedPreferences sharedPref = getPreferences(Context.MODE_PRIVATE);

final EditText etName = (EditText) findViewById(R.id.name);
final EditText etAge = (EditText) findViewById(R.id.age);
final RadioGroup radGroupSex = (RadioGroup) findViewById(
                                    R.id.sex_radio_group);

// ระบุการทำงานของปุ่ม Write Data
Button btnWriteData = (Button) findViewById(R.id.write_data_button);
btnWriteData.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        SharedPreferences.Editor editor = sharedPref.edit();

        // เก็บชื่อเป็นค่าสตริงลงใน shared preference file
        editor.putString("name", etName.getText().toString());
        // เก็บอายุเป็นค่าจำนวนเต็มลงใน shared preference file
        editor.putInt("age", Integer.valueOf(etAge.getText().toString()));

        // หาว่าปุ่มเรดิโออันใดถูกเลือก
        int selectedSex = radGroupSex.getCheckedRadioButtonId();
```

```
        // เก็บเพศเป็นค่าจำนวนเต็มลงใน shared preference file
        if (selectedSex == R.id.male_radio) {
            editor.putInt("sex", MALE);
        } else if (selectedSex == R.id.female_radio) {
            editor.putInt("sex", FEMALE);
        }

        editor.commit();
        Toast.makeText(MainActivity.this, "Data saved.",
                       Toast.LENGTH_SHORT).show();
    }
});

// ระบุการทำงานของปุ่ม Read Data
Button btnReadData = (Button) findViewById(R.id.read_data_button);
btnReadData.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        // อ่านชื่อจาก shared preference file
        String strName = sharedPref.getString("name", null);
        if (etName != null) {
            etName.setText(strName);
        }

        // อ่านอายุจาก shared preference file
        int intAge = sharedPref.getInt("age", -1);
        if (intAge != -1) {
            etAge.setText(String.valueOf(intAge));
        }

        // อ่านเพศจาก shared preference file
        int intSex = sharedPref.getInt("sex", -1);
        if (intSex == MALE) {
            radGroupSex.check(R.id.male_radio);
        } else if (intSex == FEMALE) {
            radGroupSex.check(R.id.female_radio);
        }
    }
});
```

ผลการรัน

ลองป้อนข้อมูล แล้วคลิกปุ่ม Write Data เพื่อเก็บข้อมูลลงใน shared preference file จากนั้นให้ออกจากแอพโดยคลิกปุ่ม Back (หรือกด Esc) แล้วรันแอพขึ้นมาใหม่จากหน้ารวมแอพ เมื่อคลิกปุ่ม Read Data ก็จะอ่านข้อมูลที่เก็บไว้มาแสดงบนหน้าจอ



การสร้างหน้าตัวเลือกโดยใช้ Preference API

แอพส่วนใหญ่มักจะมี “หน้าตัวเลือก” (Preferences หรือ Settings) ไว้ให้ผู้ใช้ตั้งค่าการทำงานของแอพได้ตามต้องการ เราอาจสร้างหน้าตัวเลือกโดยใช้ SharedPreferences API ดังรายละเอียดในหัวข้อที่แล้ว ซึ่งเราต้องออกแบบ Layout และควบคุมการเขียนและอ่านข้อมูลเองทั้งหมด

เพื่อลดความยุ่งยากในการสร้างหน้าตัวเลือกและเพื่อให้หน้าตัวเลือกของแอพต่างๆมี UI และลักษณะการใช้งานแบบเดียวกัน แอนดรอยด์จึงเตรียม API มาให้อีกตัวหนึ่ง นั่นคือ Preference API โดยเราจะกำหนดรายละเอียดของตัวเลือกต่างๆ แล้วแอนดรอยด์จะสร้าง Layout ของหน้าตัวเลือก รวมถึงจัดเก็บข้อมูลตอนที่ผู้ใช้กำหนดค่าในหน้าตัวเลือก (โดยใช้ SharedPreferences API) ให้อัตโนมัติ หลังจากนั้นเราสามารถอ่านข้อมูลเพื่อตรวจสอบว่าผู้ใช้กำหนดค่าไว้อย่างไร

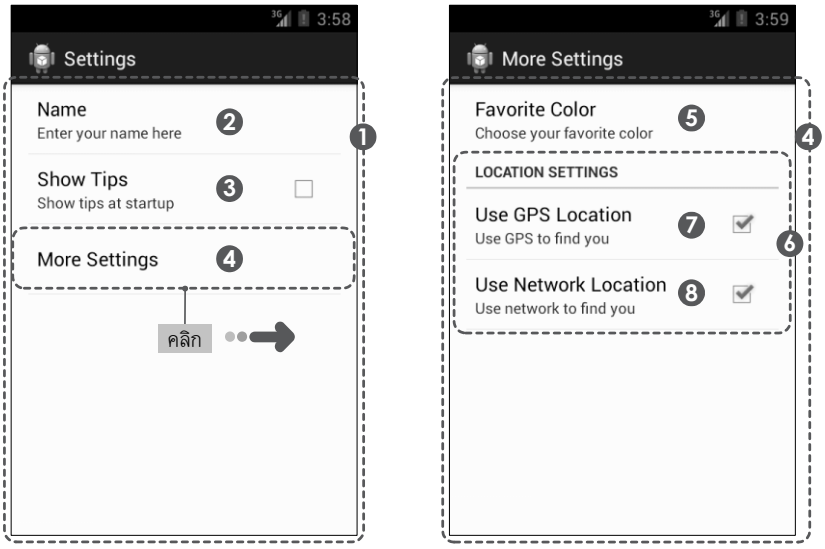
ตัวอย่างและคำอธิบาย

1 สร้างไฟล์ settings.xml ที่โฟลเดอร์ res\xml (ถ้ายังไม่มีโฟลเดอร์ xml ภายใต้โฟลเดอร์ res ก็ให้สร้างขึ้นมาก่อน) แล้วพิมพ์โค้ดดังนี้ เพื่อกำหนดรายละเอียดของตัวเลือก

โปรเจ็ค PreferenceDemo, ไฟล์ res\xml\settings

```
<PreferenceScreen ❶
    xmlns:android="http://schemas.android.com/apk/res/android" >
```

```
    <EditTextPreference ❷
        android:key="namePref"
        android:defaultValue="Anonymous"
        android:summary="Enter your name here"
        android:title="Name" />

    <CheckBoxPreference ❸
        android:key="tipsPref"
        android:defaultValue="false"
        android:summary="Show tips at startup"
        android:title="Show Tips" />

    <PreferenceScreen ❹
        android:key="moreScreen"
        android:title="More Settings" >

        <ListPreference ❺
            android:key="colorPref"
            android:defaultValue="GRN"
            android:entries="@array/color_names"
            android:entryValues="@array/color_values"
            android:summary="Choose your favorite color"
            android:title="Favorite Color" />

        <PreferenceCategory android:title="Location Settings" > ❻

            <CheckBoxPreference ❼
                android:key="gpsPref"
                android:defaultValue="true"
                android:summary="Use GPS to find you"
                android:title="Use GPS Location" />
            <CheckBoxPreference ❽
                android:key="networkPref"
                android:defaultValue="true"
                android:summary="Use network to find you"
                android:title="Use Network Location" />
        </PreferenceCategory>
    </PreferenceScreen>
</PreferenceScreen>
```



สังเกตว่าไฟล์ settings.xml ข้างต้นไม่ใช่ Layout File เนื่องจากอิลิเมนต์ในไฟล์ไม่ใช่วิว (ซับคลาสของคลาส `View`) แต่เป็น `Preference` ชนิดต่างๆ (ซับคลาสของคลาส `Preference`) เช่น `EditTextPreference`, `CheckBoxPreference`, `ListPreference` เป็นต้น ซึ่ง `Preference` แต่ละอันจะกลายเป็นไอเท็มหรือตัวเลือกหนึ่งๆในหน้าตัวเลือก

`Preference` ทุกชนิดจะมีแอตทริบิวต์พื้นฐาน ได้แก่

- `android:key` กำหนดคีย์ของตัวเลือก แอนดรอยด์จะระบุคีย์นี้เมื่อเก็บค่าตัวเลือกลงใน shared preference file และเมื่อเราจะอ่านค่าตัวเลือกมาใช้งานก็ต้องระบุคีย์นี้เช่นเดียวกัน
- `android:title` กำหนดข้อความของตัวเลือก
- `android:summary` กำหนดคำอธิบายเพิ่มเติมของตัวเลือก (ขอเรียกว่าข้อความ Summary)
- `android:defaultValue` กำหนดค่าเริ่มต้นของตัวเลือก

`ListPreference` ที่ใช้แสดงรายการตัวเลือกหรือลิสต์ ยังมีแอตทริบิวต์ที่สำคัญอีก 2 ตัวคือ

- `android:entries` กำหนดอาร์เรย์ซึ่งเป็นข้อความของตัวเลือกต่างๆในลิสต์
- `android:entryValues` กำหนดอาร์เรย์ซึ่งเป็นค่าของตัวเลือกต่างๆในลิสต์

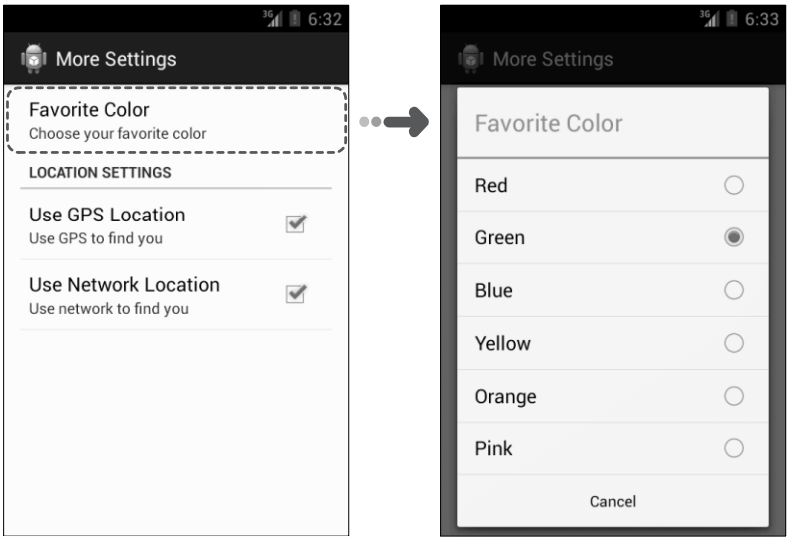
หน้าตัวเลือกจะมี `PreferenceScreen` เป็นรูทอิลิเมนต์ และหากมีอิลิเมนต์ `PreferenceScreen` ซ้อนอยู่ข้างในอีกก็จะกลายเป็นคำสั่งซึ่งเมื่อคลิกจะแสดงหน้าตัวเลือกอีกหน้าหนึ่งออกมา เช่น คำสั่ง More Settings ในตัวอย่างนี้ ❹

สำหรับอิลิเมนต์ `PreferenceCategory` ❻ จะใช้จัดกลุ่มให้กับตัวเลือกต่างๆ

2 สร้างไฟล์ arrays.xml ที่โฟลเดอร์ res\values แล้วพิมพ์โค้ดดังนี้ เพื่อกำหนดอาร์เรย์ที่ใช้กับ ListPreference

โปรเจ็ค PreferenceDemo, ไฟล์ res\values\arrays.xml

```
<resources>

    <string-array name="color_names">
        <item>Red</item>
        <item>Green</item>
        <item>Blue</item>
        <item>Yellow</item>
        <item>Orange</item>
        <item>Pink</item>
    </string-array>

    <string-array name="color_values">
        <item>RED</item>
        <item>GRN</item>
        <item>BLU</item>
        <item>YLW</item>
        <item>ORG</item>
        <item>PNK</item>
    </string-array>

</resources>
```



3 สร้างแอคทิวิตีใหม่ ตั้งชื่อว่า SettingsActivity แล้วแก้โค้ดเป็นดังนี้

โปรเจ็ค PreferenceDemo, ไฟล์ SettingsActivity.java

```
package com.example.preferencedemo;

import android.os.Bundle;
import android.preference.PreferenceActivity;

public class SettingsActivity extends PreferenceActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.activity_settings);
        addPreferencesFromResource(R.xml.settings);
    }
}
```

แอคทิวิตีนี้ก็คือหน้าตัวเลือกของเรา ซึ่งเราสร้าง UI ของแอคทิวิตีนี้โดยการ Inflate จากไฟล์ res\xml\settings.xml แทนที่จะเป็น Layout File เหมือนแอคทิวิตีทั่วไป

4 สร้างไฟล์ main.xml ที่โฟลเดอร์ res\menu แล้วพิมพ์โค้ดดังนี้ เพื่อกำหนดเมนูของแอคทิวิตี

โปรเจ็ค PreferenceDemo, ไฟล์ res\menu\main.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/settings_menu"
        android:showAsAction="ifRoom"
        android:title="Settings"/>

</menu>
```

เราสร้างคำสั่ง Settings ไว้ในเมนูเพียงคำสั่งเดียว

5 ที่แอคทิวิตีหลัก ให้เพิ่มเมธอด onCreateOptionsMenu และ onOptionsItemSelected เพื่อแสดงเมนู และระบุการทำงานเมื่อคำสั่ง Settings ในเมนูถูกคลิก

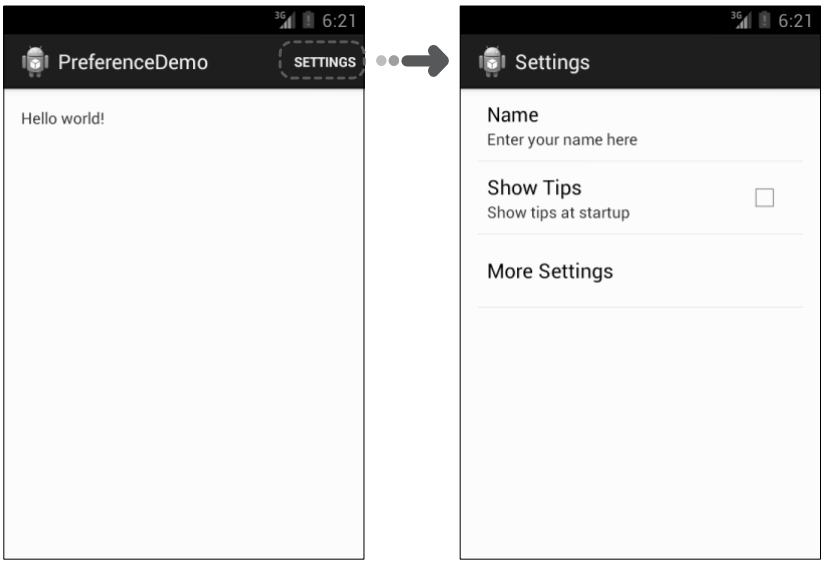
โปรเจ็ค PreferenceDemo, ไฟล์ MainActivity.java

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // สร้างเมนูโดย Inflate จากไฟล์ res\menu\main.xml
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}
@Override
```

```
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    case R.id.settings_menu:  // คำสั่ง Settings
        // รันแอคทิวิตี SettingsActivity (หน้าจอตัวเลือก)
        Intent i = new Intent(this, SettingsActivity.class);
        startActivity(i);
        return true;
    default:
        return super.onOptionsItemSelected(item);
    }
}
```

ผลการรัน

รันแอพ แล้วคลิกคำสั่ง Settings จากเมนูหรือ Action Bar เพื่อแสดงหน้าตัวเลือกออกมา



ทดลองกำหนดตัวเลือกต่างๆ จากนั้นคลิก Back (หรือกด Esc) เพื่อกลับไปยังหน้าหลักหรือออกจากแอพไปเลย แล้วเปิดแอพขึ้นมาใหม่ เข้าไปที่หน้าตัวเลือกอีกครั้ง จะเห็นว่าค่าตัวเลือกถูกจำไว้

การอ่านค่าตัวเลือกมาใช้งาน

Preference API เก็บข้อมูลที่ผู้ใช้กำหนดจากหน้าตัวเลือกไว้ใน shared preference file ซึ่งเราสามารถเข้าถึงไฟล์นี้ได้โดยใช้เมธอด `getDefaultSharedPreferences` ของ `PreferenceManager`

```
SharedPreferences settings = PreferenceManager
                        .getDefaultSharedPreferences(คอนเท็กซ์);
```

หลังจากนั้นสามารถใช้เมธอด เช่น getString, getBoolean ฯลฯ อ่านค่าตัวเลือกได้แบบเดียวกับ SharedPreferences API ในหัวข้อที่แล้ว

```
String name = settings.getString("namePref", "");
boolean showTips = settings.getBoolean("tipsPref", false);
```

ตัวอย่างนี้จะอ่านค่าตัวเลือกมาแสดงใน TextView

1 กำหนด Layout ของหน้าจอหลัก

โปรเจ็ค PreferenceDemo, ไฟล์ activity__main.xml

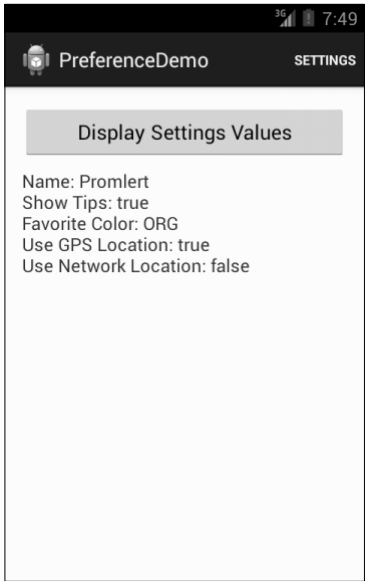
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp" >

    <Button
        android:id="@+id/display_values_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:text="Display Settings Values" />

    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        android:textSize="16sp" />

</LinearLayout>
```

2 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตีหลัก

โปรเจ็ค PreferenceDemo, ไฟล์ MainActivity.java

```
/* เรียกเมธอด setDefaultValues เพื่อกำหนดค่าดีฟอลต์ของตัวเลือกลงใน shared preference
   file สำหรับตัวเลือกที่มีการระบุแอตทริบิวต์ android:defaultValue ไว้ เนื่องจากแอพของเรา
   อาจจะอ่านค่าตัวเลือกโดยที่ผู้ใช้ยังไม่เคยเปิดหน้าตัวเลือกเลย (ซึ่งตัวเลือกต่างๆจะยังไม่มีค่าใน shared
   preference file) */
PreferenceManager.setDefaultValues(this, R.xml.settings, false);

// เข้าถึงออบเจ็ค SharedPreferences ซึ่งเป็นตัวแทนของ shared preference file
final SharedPreferences settings = PreferenceManager
                                    .getDefaultSharedPreferences(this);
```

```
// ระบุการทำงานของปุ่ม
Button btnDisplayValues = (Button) findViewById(
                                        R.id.display_values_button);
btnDisplayValues.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        // อ่านค่าตัวเลือกต่างๆ
        String name = settings.getString("namePref", "");
        boolean showTips = settings.getBoolean("tipsPref", false);
        String favoriteColor = settings.getString("colorPref", "");
        boolean useGPS = settings.getBoolean("gpsPref", false);
        boolean useNetwork = settings.getBoolean("networkPref", false);

        TextView text = (TextView) findViewById(R.id.text);

        // แสดงค่าตัวเลือกต่างๆใน TextView
        String msg = "Name: %s\nShow Tips: %b\nFavorite Color: %s\n
                     Use GPS Location: %b\nUse Network Location: %b\n";
        text.setText(String.format(msg, name, showTips, favoriteColor,
                                   useGPS, useNetwork));
    }
});
```



ระบุการทำงานเมื่อค่าตัวเลือกเปลี่ยนไป

เมื่อผู้ใช้แก้ไขค่าตัวเลือก เราอาจต้องการรับรู้ถึงการเปลี่ยนแปลงที่เกิดขึ้นในทันที โดยให้แอนดรอยด์เรียกมายังเมธอดที่เราเตรียมไว้

เราจะเพิ่มการทำงานในหน้าตัวเลือก (แอคทิวิตี `SettingsActivity`) โดยเมื่อผู้ใช้แก้ไขชื่อ (ตัวเลือก Name) จะให้แสดงชื่อนั้นเป็นข้อความ Summary ของตัวเลือกเลย (เดิมข้อความ Summary คือ “Enter your name here”)

1 เพิ่มโค้ดที่บรรทัดการประกาศคลาส `SettingsActivity` เพื่อกำหนดให้คลาสนี้ Implement อินเทอร์เฟส `SharedPreference.OnSharedPreferenceChangeListener`

โปรเจ็ค PreferenceDemo, ไฟล์ SettingsActivity.java

```
import android.content.SharedPreferences.OnSharedPreferenceChangeListener;
...

public class SettingsActivity extends PreferenceActivity
                              implements OnSharedPreferenceChangeListener {
    ...
}
```

2 เพิ่ม 2 เมธอดต่อไปนี้ใน `SettingsActivity`

โปรเจ็ค PreferenceDemo, ไฟล์ SettingsActivity.java

```
@Override
public void onSharedPreferenceChanged(SharedPreferences sharedPreferences,
                                      String key) {
    // ถ้าตัวเลือกที่ค่าเปลี่ยนไปคือตัวเลือก Name
    if (key.equals("namePref")) {
        // แสดงชื่อปัจจุบันเป็นข้อความ Summary ของตัวเลือก และแสดง Toast
        displayCurrentName(sharedPreferences);
        Toast.makeText(this, "Name changed", Toast.LENGTH_SHORT).show();
    }
}

private void displayCurrentName(SharedPreferences sharedPreferences) {
    // เข้าถึงออบเจ็ค Preference ที่เป็นตัวแทนของตัวเลือก Name
    Preference namePref = findPreference("namePref"); ❶
    // อ่านค่าปัจจุบันของตัวเลือก Name
    String msg = "Current name: " +
                 sharedPreferences.getString("namePref", ""); ❷
    // กำหนดข้อความ Summary ของตัวเลือก Name ใหม่
    namePref.setSummary(msg); ❸
}
```

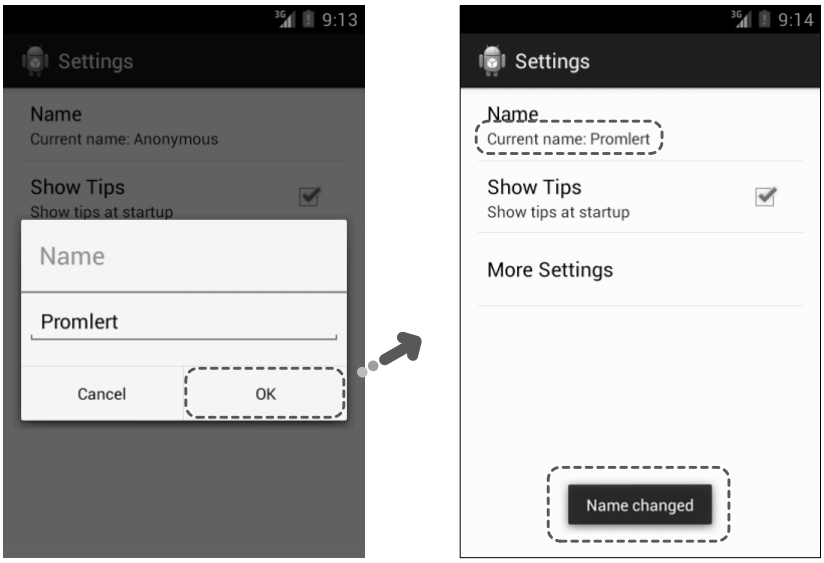
เมธอด onSharedPreferenceChanged คือ Callback ที่แอนดรอยด์จะเรียกกลับมาเมื่อค่าของตัวเลือกใดตัวเลือกหนึ่งเปลี่ยนไป ซึ่งแอนดรอยด์จะส่งออบเจ็ค SharedPreferences และคีย์ของตัวเลือกที่ถูกเปลี่ยนค่ามาเป็นพารามิเตอร์

เมื่อรู้คีย์แล้ว เราสามารถเข้าถึงตัวเลือก (ออบเจ็ค Preference) นั้นๆได้โดยใช้เมธอด findPreference ของ PreferenceActivity ❶ และสามารถกำหนดข้อความ Summary ของตัวเลือกใหม่โดยใช้เมธอด setSummary บนออบเจ็ค Preference นั้น ❸ อย่างไรก็ตาม การอ่านค่าปัจจุบันของตัวเลือก (เพื่อนำมากำหนดเป็นข้อความ Summary) ต้องใช้เมธอดบนออบเจ็ค SharedPreferences ซึ่งในที่นี้คือเมธอด getString ❷ เนื่องจาก EditTextPreference เก็บข้อมูลเป็นค่าสตริง

3 เพิ่มโค้ดในเมธอด onCreate ของ SettingsActivity

โปรเจ็ค PreferenceDemo, ไฟล์ SettingsActivity.java

```
// เข้าถึงออบเจ็ค SharedPreferences
SharedPreferences settings =
                    PreferenceManager.getDefaultSharedPreferences(this);

/* กำหนดให้คลาสปัจจุบัน (SettingsActivity) ทำหน้าที่เป็น OnSharedPreferenceChange
   Listener ของ SharedPreferences */
settings.registerOnSharedPreferenceChangeListener(this);

// ให้แสดงชื่อปัจจุบันตอนที่เปิดหน้าตัวเลือกขึ้นมาด้วย
displayCurrentName(settings);
```



การอ่าน/เขียนไฟล์บน Internal Storage

แอนดรอยด์อนุญาตให้เราอ่าน/เขียนไฟล์บนระบบไฟล์ของเครื่องได้เช่นเดียวกับแพลตฟอร์มอื่นๆ การอ่าน/เขียนไฟล์จะใช้ File API ในแพคเกจ `java.io` ตามมาตรฐานของภาษาจาวา โดยมีบางเมธอดที่แอนดรอยด์เพิ่มเข้ามาเพื่อช่วยให้เราทำงานกับไฟล์ในตำแหน่งต่างๆของระบบไฟล์ได้สะดวกขึ้น

ตำแหน่งที่เราสามารถอ่าน/เขียนไฟล์ได้แบ่งออกเป็น 2 ส่วนหลักๆ คือ Internal Storage (ที่เก็บข้อมูลภายใน) กับ External Storage (ที่เก็บข้อมูลภายนอก)

Internal Storage

- พร้อมใช้งานเสมอ
- ไฟล์ที่เราเก็บไว้ที่ตำแหน่งนี้จะเข้าถึงได้จากแอพของเราเท่านั้น
- เมื่อผู้ใช้ถอนการติดตั้งแอพของเรา แอนดรอยด์จะลบไฟล์ที่เราเก็บไว้ที่ตำแหน่งนี้
- เหมาะสำหรับการเก็บไฟล์ที่ไม่ต้องการให้ผู้ใช้หรือแอพอื่นมาเข้าถึงได้
- การอ่าน/เขียนไฟล์ที่ตำแหน่งนี้ไม่ต้องขอสิทธิ์จากแอนดรอยด์

External Storage

- อาจไม่พร้อมใช้งานในขณะนั้น เช่น ผู้ใช้ถอดการ์ด microSD ออก หรือเมาท์ (mount) External Storage ของแอนดรอยด์มาเป็นไดรว์ในพีซี
- แอพอื่นๆสามารถอ่านไฟล์ที่เราเก็บไว้ที่ตำแหน่งนี้ได้เสมอ (world-readable)
- เมื่อผู้ใช้ถอนการติดตั้งแอพของเรา แอนดรอยด์จะลบไฟล์ที่เราเก็บไว้ที่ตำแหน่งนี้ถ้าเราสร้างไฟล์โดยระบุไดเร็คทอรีด้วยเมธอด `getExternalFilesDir` แต่จะไม่ลบถ้าระบุไดเร็คทอรีด้วยเมธอด `getExternalStoragePublicDirectory`
- เหมาะสำหรับการเก็บไฟล์ที่ต้องการแชร์ให้แอพอื่นๆ หรือให้ผู้ใช้เข้าถึงจากพีซีได้
- การเขียนไฟล์ที่ตำแหน่งนี้จะต้องขอสิทธิ์ `WRITE_EXTERNAL_STORAGE` แต่การอ่านไฟล์ไม่ต้องขอสิทธิ์ (ในอนาคตอาจต้องขอสิทธิ์, ดู Note ข้อสุดท้าย)

NOTE

- External Storage ไม่จำเป็นต้องเป็นการ์ดหน่วยความจำที่ถอดได้ เครื่องแอนดรอยด์บางรุ่นอาจแบ่งพื้นที่เก็บข้อมูลภายในเครื่องออกเป็น Internal Storage กับ External Storage
- ความแตกต่างจริงๆระหว่าง Internal Storage กับ External Storage ก็คือ External Storage นั้นสามารถเมาท์ได้ (mountable) กล่าวคือ ผู้ใช้สามารถเชื่อมต่อพีซีกับเครื่องแอนดรอยด์ แล้วเมาท์ External Storage ของแอนดรอยด์ให้เป็นส่วนหนึ่งในระบบไฟล์ของพีซี เพื่อใช้พีซีเข้าถึงไฟล์บน

External Storage (นี่คือสาเหตุหนึ่งที่ทำให้ External Storage ไม่พร้อมใช้งาน และเราควรตรวจสอบก่อนทุกครั้งที่ต้องการทำงานกับไฟล์ที่ตำแหน่งนี้)

- เอกสารของแอนดรอยด์บอกว่าในอนาคตการอ่านไฟล์บน External Storage อาจต้องขอสิทธิ์ READ_EXTERNAL_STORAGE และทางที่ดีแอพที่ต้องการอ่านไฟล์บน External Storage ควรขอสิทธิ์ดังกล่าวตั้งแต่ตอนนี้เลย ไม่ต้องรอให้แอนดรอยด์บังคับก่อน อนึ่ง ถ้าขอสิทธิ์ในการเขียนแล้วก็ไม่จำเป็นต้องขอสิทธิ์ในการอ่านอีก เพราะสิทธิ์ในการเขียนจะทำได้ทั้งเขียนและอ่านไฟล์ การขอสิทธิ์ในการอ่านจะใช้เมื่อต้องการอ่านไฟล์อย่างเดียว (และยังไม่บังคับในตอนนี้)

หัวข้อนี้จะแสดงการอ่าน/เขียนไฟล์บน Internal Storage ส่วนการอ่าน/เขียนไฟล์บน External Storage จะอยู่ในหัวข้อถัดไป

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะแสดงการนำข้อความจาก `EditText` ไปเขียนลงไฟล์ และการอ่านกลับมาแสดงผลบนหน้าจอ

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค FileInternalDemo, ไฟล์ 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="vertical"
    android:padding="16dp" >

    <EditText
        android:id="@+id/input_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="12dp"
        android:ems="10" >

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/save_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Save" />

    <Button
        android:id="@+id/load_button"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:text="Load" />

    <TextView
        android:id="@+id/output_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="12dp" />

</LinearLayout>
```

2 ที่แอคทิวิตี ให้เพิ่มค่าคงที่ระดับคลาส

โปรเจ็ค FileInternalDemo, ไฟล์ MainActivity.java

```
private static final String FILENAME = "data.txt";
private static final int READ_BLOCK_SIZE = 100;
```

3 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตี เพื่ออ้างอิงวิวต่างๆ ใน Layout และระบุการทำงานของปุ่ม Save

โปรเจ็ค FileInternalDemo, ไฟล์ MainActivity.java

```
final EditText inputText = (EditText) findViewById(R.id.input_text);
final TextView outputText = (TextView) findViewById(R.id.output_text);
final Button btnSave = (Button) findViewById(R.id.save_button);
final Button btnLoad = (Button) findViewById(R.id.load_button);

// ระบุการทำงานของปุ่ม Save
btnSave.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        String data = inputText.getText().toString();
        try {
            // เขียนข้อความจาก EditText ลงไฟล์
            FileOutputStream fOut = openFileOutput(FILENAME,
                                                   MODE_PRIVATE); ❶
            OutputStreamWriter writer = new OutputStreamWriter(fOut); ❷

            writer.write(data); ❸
            writer.flush(); ❹
            writer.close(); ❺

            // ลบข้อความใน EditText และแสดง Toast
            inputText.setText("");
            Toast.makeText(MainActivity.this,
```

```
                    "File saved successfully!", Toast.LENGTH_SHORT)
                    .show();
        } catch (IOException ioe) {
            ioe.printStackTrace();
        }
    }
});
```

เมธอด `openFileOutput` ❶ จะเปิดไฟล์เพื่อเขียนข้อมูลในไดเร็คทอรีส่วนตัวของแอพบน Internal Storage (เอกสารของแอนดรอยด์เรียกไดเร็คทอรีนี้ว่า Internal Directory) โดยหากยังไม่มีไฟล์ที่ระบุจะสร้างไฟล์ขึ้นมาใหม่ และหากมีไฟล์อยู่แล้วจะเขียนทับข้อมูลเดิมในไฟล์ แต่ถ้าหากต้องการเขียนต่อท้ายข้อมูลเดิมให้ระบุพารามิเตอร์ตัวที่สองเป็น `MODE_APPEND` ทั้งนี้เมธอด `openFileOutput` จะส่งคืนออบเจ็ค `FileOutputStream` กลับมาให้

ถัดไปสร้าง `OutputStreamWriter` ขึ้นมาครอบ `FileOutputStream` ❷ แล้วเรียกเมธอด `write` เพื่อเขียนข้อมูลลงไฟล์ ❸ จากนั้นทำการ flush ข้อมูล ❹ แล้วจึงปิดไฟล์ ❺

4 เพิ่มโค้ดในเมธอด `onCreate` ถัดจากโค้ดที่เพิ่มก่อนหน้านี้ เพื่อระบุการทำงานของปุ่ม Load

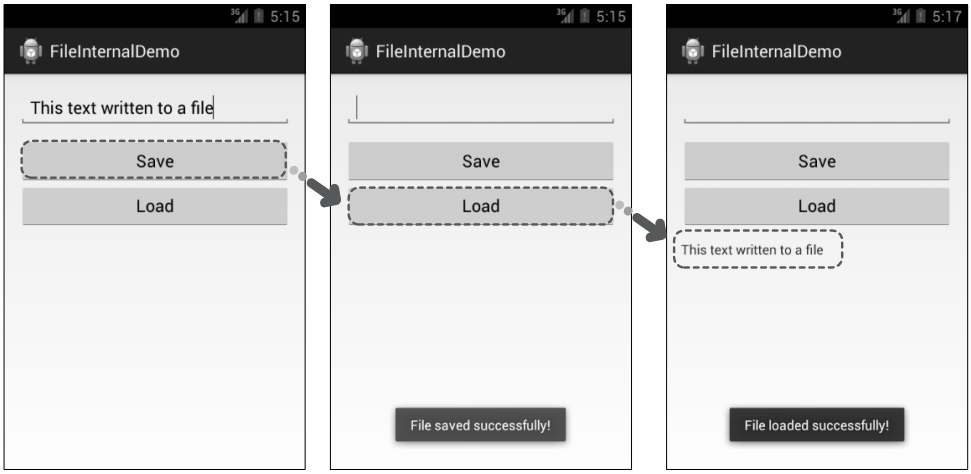
โปรเจ็ค FileInternalDemo, ไฟล์ MainActivity.java

```
// ระบุการทำงานของปุ่ม Load
btnLoad.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        try {
            FileInputStream fIn = openFileInput(FILENAME); ❶
            InputStreamReader reader = new InputStreamReader(fIn); ❷

            char[] buffer = new char[READ_BLOCK_SIZE];
            String data = "";
            int charReadCount;
            while ((charReadCount = reader.read(buffer)) > 0) {
                // แปลงไบต์ข้อมูลเป็นสตริง
                String readString = String.copyValueOf(buffer, 0,
                                        charReadCount);
                data += readString;
                buffer = new char[READ_BLOCK_SIZE];
            }                                                          ❸
            reader.close();

            outputText.setText(data); ❹
            Toast.makeText(MainActivity.this,
```

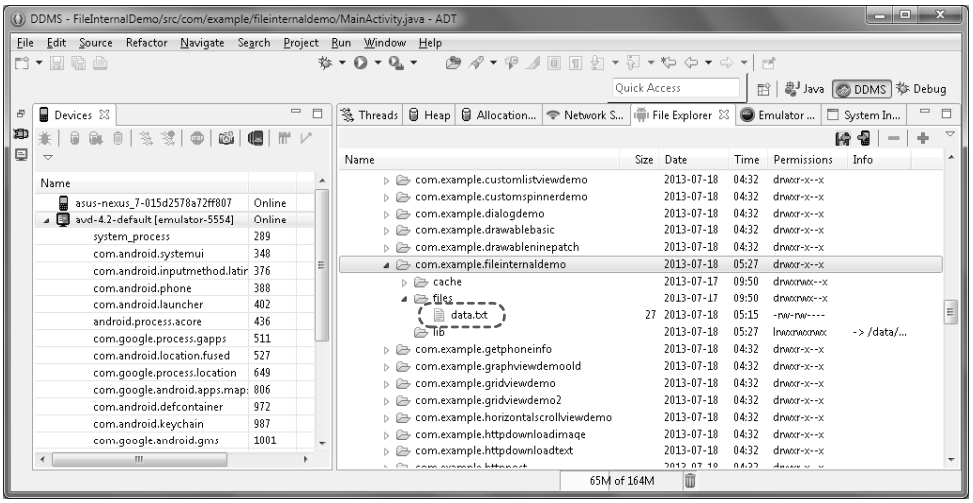
```
                    "File loaded successfully!", Toast.LENGTH_SHORT)
                    .show();
            } catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
});
```

การเปิดไฟล์เพื่ออ่านข้อมูลจะใช้เมธอด `openFileInput` ❶ ซึ่งไฟล์ที่จะเปิดด้วยเมธอดนี้ต้องอยู่ใน Internal Directory ของแอพ จากนั้นสร้าง `InputStreamReader` ครอบ `FileInputStream` ที่เมธอด `openFileInput` ส่งคืนมาให้ ❷ แล้วใช้เมธอด `read` อ่านข้อมูลจากไฟล์ทีละส่วน (ในที่นี้คือทีละ 100 ไบต์ ตามที่กำหนดด้วยค่าคงที่ `READ_BLOCK_SIZE`) จนครบทั้งไฟล์ ❸ แล้วจึงแสดงข้อมูลออกมาที่ `TextView` ❹

ผลการรัน



หลังจากบันทึกไฟล์แล้ว ให้ลองเปิด DDMS Perspective ใน Eclipse (เมนู **Window ▸ Open Perspective ▸ Other ▸ DDMS**) จากนั้นคลิกแท็บ **File Explorer** แล้วเข้าไปยัง Internal Directory ของแอพ คือที่ไดเร็คทอรี `\data\data\`*`ชื่อแพคเกจของแอพ`*`\files` คุณจะเห็นไฟล์ data.txt อยู่ในนี้



วิธีอื่นในการเข้าถึงไฟล์ใน Internal Directory

นอกจากเมธอด `openFileInput` และ `openFileOutput` แล้ว เรายังสามารถทำงานกับไฟล์ใน Internal Directory ของแอพได้อีกวิธีหนึ่ง โดยก่อนอื่นจะต้องเข้าถึง Internal Directory โดยใช้เมธอดใดเมธอดหนึ่งต่อไปนี้

- **`getFilesDir`** ให้ผลลัพธ์เป็นออบเจ็ค `File` ที่แทน Internal Directory ของแอพ (ไดเร็คทอรี `\data\data\`*ชื่อแพคเกจของแอพ*`\files`)
- **`getCacheDir`** ให้ผลลัพธ์เป็นออบเจ็ค `File` ที่แทน Internal Directory ที่ใช้เก็บไฟล์ชั่วคราวของแอพ (ไดเร็คทอรี `\data\data\`*ชื่อแพคเกจของแอพ*`\cache`) ซึ่งหาก Internal Storage มีที่ว่างเหลือน้อย แอนดรอยด์อาจลบไฟล์ในไดเร็คทอรีนี้โดยไม่แจ้งให้คุณรู้

จากนั้นเมื่อต้องการอ่าน/เขียนไฟล์ ให้เขียนโค้ดดังตัวอย่าง

```
// เข้าถึง Internal Directory
File dir = getFilesDir();
// เข้าถึงไฟล์ที่ต้องการอ่าน/เขียน
File file = new File(dir, FILENAME);
// สร้าง FileOutputStream เพื่อเขียนข้อมูลลงไฟล์
FileOutputStream fOut = new FileOutputStream(file);
...
```

วิธีนี้ยังมีข้อดีตรงที่เราสามารถสร้างไดเร็คทอรีย่อยขึ้นมาภายใต้ Internal Directory แล้วทำงานกับไฟล์ในไดเร็คทอรีย่อยนั้น เช่น สมมติต้องการเขียนไฟล์ในไดเร็คทอรีย่อย images (ไดเร็คทอรี `\data\data\`*ชื่อแพคเกจของแอพ*`\files\images`) จะเขียนโค้ดได้ดังนี้

```
// ชื่อพาธเต็มของไดเร็คทอรีย่อยที่จะสร้างขึ้นภายใต้ Internal Directory
String subDirPath = getFilesDir().getAbsolutePath() + "/images";
// สร้างออบเจ็ค File ที่เป็นตัวแทนของไดเร็คทอรีย่อยนั้น
File subDir = new File(subDirPath);
// ถ้ายังไม่มีไดเร็คทอรีย่อยนั้น ให้สร้างขึ้นมา
if (!subDir.exists()) {
    subDir.mkdir();
}
// เข้าถึงไฟล์ในไดเร็คทอรีย่อยที่ต้องการอ่าน/เขียน
File file = new File(subDir, FILENAME);
// สร้าง FileOutputStream เพื่อเขียนข้อมูลลงไฟล์
FileOutputStream fOut = new FileOutputStream(file);
...
```

การอ่าน/เขียนไฟล์บน External Storage

ความแตกต่างระหว่างการอ่าน/เขียนไฟล์บน External Storage กับการอ่าน/เขียนไฟล์บน Internal Storage มีเพียงเรื่องเดียวคือวิธีการเข้าถึงไดเร็คทอรีที่เก็บไฟล์

การเข้าถึงไดเร็คทอรีบน External Storage จะใช้เมธอดใดเมธอดหนึ่งต่อไปนี้

- **`Environment.getExternalStorageDirectory`** ใช้เข้าถึงรูทไดเร็คทอรีของ External Storage
- **`Environment.getExternalStoragePublicDirectory`** ใช้เข้าถึง Public Directory (ไดเร็คทอรีสาธารณะ) บน External Storage ซึ่งมีหลายไดเร็คทอรีขึ้นอยู่กับประเภทของไฟล์ที่เก็บ เช่น ไฟล์ภาพ, ไฟล์เพลง, ไฟล์วิดีโอ ฯลฯ เราจะต้องระบุพารามิเตอร์เป็นค่าคงที่ เช่น `Environment.DIRECTORY_PICTURES` เพื่อกำหนดว่าต้องการเข้าถึงไดเร็คทอรีใด
- **`Context.getExternalFilesDir`** ใช้เข้าถึง Private Directory (ไดเร็คทอรีส่วนตัวของแอพ) บน External Storage ซึ่งมีหลายไดเร็คทอรีขึ้นอยู่กับประเภทไฟล์เช่นเดียวกับ Public Directory โดยให้ระบุพารามิเตอร์เป็นค่าคงที่ เช่น `Environment.DIRECTORY_PICTURES` เพื่อกำหนดว่าต้องการเข้าถึงไดเร็คทอรีใด
- **`Context.getExternalCacheDir`** ใช้เข้าถึง Private Directory ที่เก็บไฟล์ชั่วคราวของแอพ บน External Storage

ทั้งนี้ Public Directory คือไดเร็คทอรีสำหรับเก็บไฟล์ที่ผู้ใช้และแอพอื่นๆสามารถมาใช้งานได้ ซึ่งไฟล์เหล่านี้จะไม่ถูกลบทิ้งเมื่อผู้ใช้ถอนการติดตั้งแอพของเรา เช่น ไฟล์ภาพที่ผู้ใช้ถ่ายหรือสร้างขึ้นด้วยแอพของเรา เป็นต้น

ส่วน Private Directory ใช้เก็บไฟล์ที่ไม่มีประโยชน์ต่อผู้ใช้หรือแอพอื่น (แต่ที่จริงแล้วผู้ใช้และแอพอื่นสามารถเข้าถึงได้) ซึ่งไฟล์เหล่านี้จะถูกลบทิ้งเมื่อผู้ใช้ถอนการติดตั้งแอพของเรา เช่น ไฟล์รีซอร์สของแอพ เป็นต้น

สิ่งสำคัญอีกเรื่องหนึ่งในการทำงานกับ External Storage ก็คือ ควรตรวจสอบว่า External Storage พร้อมใช้งานหรือไม่ก่อนที่จะดำเนินการใดๆ เนื่องจากผู้ใช้อาจถอด External Storage ออก (ถ้าถอดได้) หรือใช้พีซีเข้าถึง External Storage ซึ่งทำให้แอพของเราเข้าถึงไม่ได้

ตัวอย่างและคำอธิบาย

เราจะปรับปรุงตัวอย่างในหัวข้อที่แล้วให้อ่าน/เขียนไฟล์บน External Storage แทน (ในซอร์สโค้ดจะแยกออกมาเป็นโปรเจ็คใหม่ชื่อ FileExternalDemo)

1 ที่แอคทิวิตี ให้เพิ่มเมธอด `isExternalStorageAvailable` สำหรับตรวจสอบว่า External Storage พร้อมใช้งานหรือไม่

โปรเจ็ค FileExternalDemo, ไฟล์ MainActivity.java

```
public boolean isExternalStorageAvailable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}
```

การตรวจสอบสถานะของ External Storage ให้ใช้เมธอด `getExternalStorageState` ซึ่งถ้าสถานะคือ `MEDIA_MOUNTED` (คำว่า mount ตรงนี้หมายถึงเมาท์กับระบบไฟล์ของแอนดรอยด์) แสดงว่าเราสามารถอ่าน/เขียนไฟล์บน External Storage ได้

2 แก้ไขโค้ดในเมธอด `onCreate` ของแอคทิวิตี (โค้ดที่ถูกขีดฆ่าคือให้ลบทิ้ง, โค้ดที่มีแถบสีคือให้พิมพ์เพิ่ม)

โปรเจ็ค FileExternalDemo, ไฟล์ MainActivity.java

```
final EditText inputText = (EditText) findViewById(R.id.input_text);
final TextView outputText = (TextView) findViewById(R.id.output_text);
final Button btnSave = (Button) findViewById(R.id.save_button);
final Button btnLoad = (Button) findViewById(R.id.load_button);

/* ไดเร็คทอรีสำหรับเก็บรูปภาพภายใต้ Public Directory ของ External Storage (สมมติว่าไฟล์
   data.txt ที่เราอ่าน/เขียนในตัวอย่างนี้คือไฟล์ภาพ) */
final File dir = Environment.getExternalStoragePublicDirectory(
                    Environment.DIRECTORY_PICTURES);

// ไฟล์ที่จะเราจะอ่าน/เขียน
```

```
final File file = new File(dir, FILENAME);

// ระบุการทำงานของปุ่ม Save
btnSave.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        // ถ้า External Storage ไม่พร้อมใช้งาน ให้แสดง Toast แล้วออกจากเมธอดทันที
        if (!isExternalStorageAvailable()) {
            String msg = "External storage not available!";
            Toast.makeText(MainActivity.this, msg, Toast.LENGTH_LONG)
                    .show();
            return;
        }

        // ถ้ายังไม่มีไดเร็คทอรีดังกล่าว ให้สร้างขึ้นมา
        if (!dir.exists()) {
            dir.mkdir();
        }

        String data = inputText.getText().toString();
        try {
            FileOutputStream fOut = openFileOutput(FILENAME, MODE_PRIVATE);

            FileOutputStream fOut = new FileOutputStream(file);
            OutputStreamWriter writer = new OutputStreamWriter(fOut);

            ...

        } catch (IOException ioe) {
            ioe.printStackTrace();
        }
    }
});

// ระบุการทำงานของปุ่ม Load
btnLoad.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        // ถ้า External Storage ไม่พร้อมใช้งาน ให้แสดง Toast แล้วออกจากเมธอดทันที
        if (!isExternalStorageAvailable()) {
            String msg = "External storage not available!";
            Toast.makeText(MainActivity.this, msg, Toast.LENGTH_LONG)
                    .show();
            return;
```

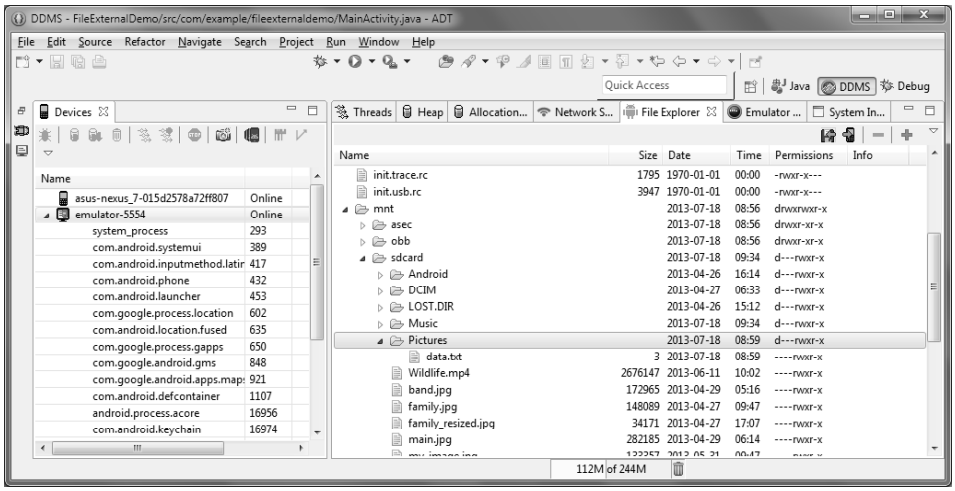
```
        }

        try {
            FileInputStream fIn = openFileInput(FILENAME);
            FileInputStream fIn = new FileInputStream(file);
            InputStreamReader reader = new InputStreamReader(fIn);

            ...

        } catch (IOException ioe) {
            ioe.printStackTrace();
        }
    }
});
```

3 เพิ่มบรรทัดต่อไปนี้ในไฟล์ AndroidManifest.xml โดยใส่ไว้ก่อนแท็ก `<application>` เพื่อขอสิทธิ์ในการเขียนไฟล์บน External Storage

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

ผลการรัน

ผลบนหน้าจอจะเหมือนตัวอย่างที่แล้ว แต่ไฟล์ data.txt จะถูกสร้างขึ้นในไดเร็คทอรี \mnt\sdcard\Pictures ดังรูป (รูทไดเร็คทอรีของ External Storage คือ \mnt\sdcard ในกรณีของอีมูเลเตอร์)



การทำงานกับฐานข้อมูล SQLite

ฐานข้อมูล (Database) เหมาะสำหรับการเก็บข้อมูลที่มีโครงสร้างแบบเดียวกัน เช่น ข้อมูลรายชื่อผู้ติดต่อ ซึ่งแต่ละรายชื่อจะประกอบด้วยชื่อ, เบอร์โทรศัพท์ และอีเมล์แอดเดรส เป็นต้น

แอนดรอยด์มีระบบฐานข้อมูล SQLite รวมอยู่ในตัว และจัดเตรียม API ไว้ให้เราสร้างฐานข้อมูลและทำงานกับข้อมูลในฐานข้อมูลได้อย่างง่ายดาย

ฐานข้อมูลที่แอพของเราสร้างขึ้นจะถูกเก็บไว้ใน Internal Directory (ไดเร็คทอรีส่วนตัวของแอพบน Internal Storage) ซึ่งโดยดีฟอลต์แล้วแอพอื่นๆจะไม่สามารถเข้าถึงได้

การใช้งานฐานข้อมูล SQLite นั้น โดยปกติเราจะไม่สร้างฐานข้อมูลเตรียมไว้ล่วงหน้าตั้งแต่ช่วงเขียนแอพ แต่จะสร้างขึ้นตอนที่แอพทำงานเลย (runtime) ซึ่งแอนดรอยด์มี “คลาสตัวช่วย” หรือ Helper Class ไว้ช่วยอำนวยความสะดวกในการสร้างฐานข้อมูล, อัพเกรดฐานข้อมูล และเข้าถึงฐานข้อมูลเพื่ออ่าน/เขียนข้อมูล

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะแสดงการสร้างฐานข้อมูลง่ายๆที่เก็บชื่อบุคคลและวันเวลาขณะที่ข้อมูลถูกเพิ่มลงในฐานข้อมูล

1 สร้างคลาสใหม่ชื่อ `MyDbHelper` แล้วพิมพ์โค้ดดังนี้

โปรเจ็ค DatabaseDemo, ไฟล์ MyDbHelper.java

```
package com.example.databasedemo;

import java.util.Date;
import java.text.SimpleDateFormat;

import android.content.ContentValues;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

// Helper Class
public class MyDbHelper extends SQLiteOpenHelper {

    private static final String DB_NAME = "mydb";       // ชื่อฐานข้อมูล
    private static final int DB_VERSION = 1;            // เวอร์ชั่นของฐานข้อมูล

    public static final String TABLE_NAME = "people";  // ชื่อเทเบิล
    public static final String COL_NAME = "pName";      // ชื่อฟิลด์ที่เก็บชื่อ
    public static final String COL_DATE = "pDate";      // ชื่อฟิลด์ที่เก็บวันเวลา
```

```
    // คำสั่ง SQL ที่ใช้สร้างเทเบิล
    private static final String STRING_CREATE =
        "CREATE TABLE " + TABLE_NAME
        + " (_id INTEGER PRIMARY KEY AUTOINCREMENT, "
        + COL_NAME + " TEXT, " + COL_DATE + " DATE);";

    public MyDbHelper(Context context) {
        /* เรียกไปยังคอนสตรัคเตอร์ของ SQLiteOpenHelper โดยระบุชื่อและเวอร์ชั่นของฐานข้อมูล
           เป็นพารามิเตอร์ตัวที่ 2 และ 4 ตามลำดับ */
        super(context, DB_NAME, null, DB_VERSION); ❶
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // สร้างเทเบิลในฐานข้อมูล
        db.execSQL(STRING_CREATE); ❷

        // สร้างออบเจ็ค ContentValues เพื่อเพิ่มข้อมูล 1 รายการ (แถว) ไว้ในฐานข้อมูลตั้งแต่แรก
        ContentValues cv = new ContentValues();

        // ใส่ชื่อ
        cv.put(COL_NAME, "Promlert Lovichit");
        // กำหนดรูปแบบของวันเวลา
        SimpleDateFormat dateFormat = new SimpleDateFormat(
                "yyyy-MM-dd HH:mm:ss");
        // ใส่วันเวลา ณ ขณะนั้นตามรูปแบบที่กำหนด
        cv.put(COL_DATE, dateFormat.format(new Date()));

        // เพิ่มข้อมูลลงฐานข้อมูล
        db.insert(TABLE_NAME, null, cv);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion,
                          int newVersion) {
        // ในที่นี้สมมติว่าเมื่ออัพเกรดฐานข้อมูล ให้ลบฐานข้อมูลเดิมทิ้งแล้วสร้างใหม่
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME); ❺
        onCreate(db); ❻
    }
}
```

❹ ❸

คลาส `MyDbHelper` นี้จะทำหน้าที่เป็น Helper Class ที่ช่วยในการสร้างและทำงานกับฐานข้อมูล เนื่องจากเรากำหนดมันเป็นซับคลาสของ `SQLiteOpenHelper`

ภายในคอนสตรัคเตอร์ของคลาส เราระบุชื่อและเวอร์ชั่นของฐานข้อมูลให้แอนดรอยด์รู้โดยเรียกไปยังคอนสตรัคเตอร์ของ SQLiteOpenHelper ❶ หลังจากนั้นเมื่อมีการเข้าถึงฐานข้อมูลเพื่ออ่าน/เขียนข้อมูล (โดยใช้เมธอด getReadableDatabase, getWritableDatabase ที่จะอธิบายต่อไป) แอนดรอยด์จะตรวจสอบว่ามีฐานข้อมูลหรือยัง ถ้ายังไม่มีก็จะสร้างฐานข้อมูลว่างๆให้ แล้วเรียกมายังเมธอด onCreate ใน Helper Class ของเรา พร้อมทั้งส่ง reference ของฐานข้อมูลมาเป็นพารามิเตอร์ เพื่อให้เราจัดเตรียมโครงสร้างของฐานข้อมูลตามที่ต้องการ ซึ่งในที่นี้เราจะสร้างเทเบิลขึ้นมา ❷ และเพิ่มข้อมูล 1 แถว (row) หรือ 1 รายการ (record) ไว้ตั้งแต่แรกเลย ❸

ค่าคงที่ STRING_CREATE ❹ คือคำสั่ง SQL ที่ใช้สร้างเทเบิล ซึ่งเมื่อนำค่าคงที่อื่นๆ มาแทนค่าลงไปจะได้คำสั่งที่สมบูรณ์ดังนี้ (ขอแบ่งคำสั่งเป็นหลายบรรทัดเพื่อให้เข้าใจง่าย)

```
CREATE TABLE people (
        _id INTEGER PRIMARY KEY AUTOINCREMENT,
        pName TEXT,
        pDate DATE
);
```

จะเห็นว่าเป็นคำสั่งสำหรับสร้างเทเบิลชื่อ people ซึ่งประกอบด้วย 3 ฟิลด์ คือ

_id	เก็บค่าจำนวนเต็ม, เป็น Primary Key และให้กำหนดค่าอัตโนมัติเมื่อเพิ่มแถวข้อมูลใหม่
pName	เก็บข้อความ (สตริง)
pDate	เก็บวันเวลา

เมื่อเราพัฒนาแอพเวอร์ชั่นใหม่ เราอาจออกแบบให้แอพทำงานกับฐานข้อมูลที่มีโครงสร้างเปลี่ยนไปจากเดิม เช่น มีการเพิ่มฟิลด์เข้ามา เป็นต้น ซึ่ง SQLiteOpenHelper มีเมธอด onUpgrade ไว้รองรับการอัพเกรดฐานข้อมูล โดยหากแอนดรอยด์พบว่าเลขเวอร์ชั่นของฐานข้อมูลที่ระบุในคอนสตรัคเตอร์มากกว่าเลขเวอร์ชั่นของฐานข้อมูลที่มีอยู่แล้ว มันจะเรียกมายังเมธอด onUpgrade ใน Helper Class ของเรา พร้อมทั้งส่งตัวฐานข้อมูล, เลขเวอร์ชั่นเดิม และเลขเวอร์ชั่นใหม่มาเป็นพารามิเตอร์ เพื่อให้เราแก้ไขโครงสร้างฐานข้อมูลหรือดำเนินการใดๆตามความเหมาะสม

ในที่นี้เราอัพเกรดฐานข้อมูลโดยลบฐานข้อมูลเดิมทิ้ง ❺ แล้วเรียกต่อไปยัง onCreate เพื่อให้สร้างฐานข้อมูลและเทเบิลขึ้นมาใหม่ ❻ แต่อย่างไรก็ตาม ในการพัฒนาจริงๆเราอาจแก้ไขโครงสร้างเทเบิลโดยใช้คำสั่ง ALTER ของภาษา SQL ยกตัวอย่างเช่น ถ้าพัฒนาแอพออกไปแล้ว 3 เวอร์ชั่น ดังนี้ (สมมติกำหนดเวอร์ชั่นของฐานข้อมูลตามเวอร์ชั่นของแอพ)

- **เวอร์ชั่น 1:** มีฟิลด์สำหรับเก็บชื่อและวันเวลาที่ข้อมูลถูกบันทึก
- **เวอร์ชั่น 2:** เพิ่มฟิลด์สำหรับเก็บหมายเลขโทรศัพท์
- **เวอร์ชั่น 3:** เพิ่มฟิลด์สำหรับเก็บอีเมล์แอดเดรส

เราจะเขียนโค้ดในเมธอด onUpgrade ของแอพเวอร์ชั่น 3 ดังนี้

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // ถ้าเวอร์ชั่นเดิมคือ 1 ให้เพิ่มหมายเลขโทรศัพท์
    if (oldVersion <= 1) {
        db.execSQL("ALTER TABLE " + TABLE_NAME
                + " ADD COLUMN phone_number TEXT;");
    }
    // ถ้าเวอร์ชั่นเดิมคือ 1 หรือ 2 ให้เพิ่มฟิลด์ email
    if (oldVersion <= 2) {
        db.execSQL("ALTER TABLE " + TABLE_NAME
                + " ADD COLUMN email TEXT;");
    }
}
```

จากโค้ด ถ้าหากฐานข้อมูลเดิมเป็นเวอร์ชั่น 2 จะเพิ่มฟิลด์ email ฟิลด์เดียว แต่หากฐานข้อมูลเดิมเป็นเวอร์ชั่น 1 จะเพิ่มทั้งฟิลด์ phone_number และ email

NOTE»»»

การอัพเกรดฐานข้อมูลต้องตระหนักว่าผู้ใช้อาจไม่ได้ติดตั้งแอพมาทีละเวอร์ชั่นตามลำดับ แต่อาจติดตั้งเวอร์ชั่น 1 แล้วข้ามมาเวอร์ชั่น 3 เลยโดยไม่ผ่านเวอร์ชั่น 2 มาก่อน

ถัดไปจะออกแบบ UI และเขียนโค้ดในแอคทิวิตีเพื่อใช้งานฐานข้อมูล โดยจะมี EditText และปุ่มไว้สำหรับเพิ่มชื่อลงฐานข้อมูล, มี ListView สำหรับแสดงข้อมูลทั้งหมดจากฐานข้อมูล ซึ่งหากคลิกไอเท็มใน ListView จะลบข้อมูลนั้นในฐานข้อมูล

2 กำหนด Layout ของหน้าจอ

โปรเจ็ค DatabaseDemo, ไฟล์ 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="vertical"
    android:padding="16dp" >

    <EditText
        android:id="@+id/person_name"
```

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

    <Button
        android:id="@+id/add_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="12dp"
        android:layout_marginTop="8dp"
        android:text="Add New Person" />

    <ListView
        android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

3 ที่แอคทิวิตี ให้เพิ่มโค้ดที่บรรทัดการประกาศคลาส

โปรเจ็ค DatabaseDemo, ไฟล์ MainActivity.java

```
public class MainActivity extends Activity
        implements View.OnClickListener, AdapterView.OnItemClickListener {
```

เรากำหนดให้แอคทิวิตีทำการ Implement อินเทอร์เฟส `View.OnClickListener` และ `AdapterView.OnItemClickListener` เพื่อที่จะเพิ่มเมธอด `onClick` สำหรับระบุการทำงานเมื่อปุ่มถูกคลิก และเมธอด `onItemClick` สำหรับระบุการทำงานเมื่อไอเท็มใน `ListView` ถูกคลิก ไว้ภายในแอคทิวิตีนี้

4 เพิ่มตัวแปรระดับคลาสของแอคทิวิตี

โปรเจ็ค DatabaseDemo, ไฟล์ MainActivity.java

```
EditText etPersonName;        // EditText สำหรับกรอกชื่อ
Button btnAddPerson;          // ปุ่มสำหรับเพิ่มข้อมูลลงฐานข้อมูล
ListView list;                // ListView สำหรับแสดงข้อมูลจากฐานข้อมูล

MyDbHelper dbHelper;          // Helper Object ที่ช่วยในการสร้างและเข้าถึงฐานข้อมูล
SQLiteDatabase db;            // ฐานข้อมูลที่เราจะอ่าน/เขียนข้อมูล
Cursor cursor;                /* ผลลัพธ์ (Result Set) ของคิวรี ซึ่งมีเมธอดสำหรับเลื่อน
                                 ไปยังแถวข้อมูลต่างๆใน Result Set */
SimpleCursorAdapter adapter; // Adapter สำหรับนำข้อมูลจากฐานข้อมูลมาแสดงใน ListView
```

5 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตี

โปรเจ็ค DatabaseDemo, ไฟล์ MainActivity.java

```
// อ้างอิง EditText
etPersonName = (EditText) findViewById(R.id.person_name);

// อ้างอิงปุ่มและกำหนด Listener ที่ระบุการทำงานเมื่อปุ่มถูกคลิก
btnAddPerson = (Button) findViewById(R.id.add_button);
btnAddPerson.setOnClickListener(this);

// อ้างอิง ListView และกำหนด Listener ที่ระบุการทำงานเมื่อไอเท็มใน ListView ถูกคลิก
list = (ListView) findViewById(R.id.list);
list.setOnItemClickListener(this);

// สร้าง Helper Object
dbHelper = new MyDbHelper(this);
```

6 เพิ่มเมธอด onResume และ onPause ในแอคทิวิตี

โปรเจ็ค DatabaseDemo, ไฟล์ MainActivity.java

```
@Override
public void onResume() {
    super.onResume();
    // เข้าถึงฐานข้อมูลเพื่ออ่าน/เขียนข้อมูล
    db = dbHelper.getWritableDatabase();
    // ชื่อฟิลด์ที่จะคิวรีข้อมูล
    String[] queryColumns = new String[] {
            "_id", MyDbHelper.COL_NAME, MyDbHelper.COL_DATE };
    // คิวรีฐานข้อมูล
    cursor = db.query(MyDbHelper.TABLE_NAME, queryColumns, null, null,
                      null, null, null); ❶
    // ชื่อฟิลด์ที่จะแสดงข้อมูลออกมาใน ListView
    String[] showColumns = new String[] { MyDbHelper.COL_NAME,
                                          MyDbHelper.COL_DATE };
    // ID ของวิวที่จะใส่ข้อมูลลงไป
    int[] views = new int[] { android.R.id.text1, android.R.id.text2 };
    // สร้าง Adapter ที่นำข้อมูลจาก Cursor (ผลของคิวรี) มาใส่ลงใน Layout
    adapter = new SimpleCursorAdapter(this,
                      android.R.layout.two_line_list_item, cursor,
                      showColumns, views); ❷
    // กำหนด Adapter ให้กับ ListView
    list.setAdapter(adapter); ❸
}
```

```
@Override
public void onPause() {
    super.onPause();
    // ปิด Cursor และการเชื่อมต่อกับฐานข้อมูล
    cursor.close();
    db.close();
}
```

เราทำการเชื่อมต่อและคิวรีฐานข้อมูลในเมธอด onResume และยกเลิกการเชื่อมต่อในเมธอด onPause เพื่อที่จะอัพเดทข้อมูลใน ListView ทุกครั้งที่แอพของเราแสดงผลออกมาใหม่ (เช่น เมื่อมีแอพอื่นรันขึ้นมาข้างหน้า และต่อมาผู้ใช้ออกจากแอพนั้นแล้วกลับมายังแอพของเรา)

ในที่นี้เราคิวรีข้อมูลทุกแถวจากเทเบิล people โดยระบุว่าต้องการข้อมูลจากฟิลด์ (คอลัมน์) _id, pName และ pDate ❶ สำหรับพารามิเตอร์ที่เหลือของเมธอด query จะใช้กำหนดเงื่อนไขของแถวข้อมูลที่จะเลือกมา, การเรียงลำดับ และอื่นๆ ซึ่งในที่นี้ระบุเป็นค่า null ทั้งหมด

ผลของการคิวรี (Result Set) จะถูกเก็บในออบเจ็ค Cursor หลังจากนั้นเราสร้าง Adapter ชนิด SimpleCursorAdapter เพื่อนำข้อมูลชื่อและวันเวลาของแถวข้อมูลแต่ละแถว ใส่ลงใน Layout แบบ 2 บรรทัดที่แอนดรอยด์เตรียมมาให้ (android.R.layout.two_line_list_item) ซึ่ง Layout นี้ประกอบด้วย TextView 2 อันที่มี ID ว่า android.R.id.text1 (บรรทัดบน) และ android.R.id.text2 (บรรทัดล่าง) โดยเราใส่ชื่อและวันเวลาลงไป ตามลำดับ ❷ สุดท้ายจึงกำหนด Adapter นี้ให้กับ ListView ❸

7 เพิ่มเมธอด onClick ในแอคทิวิตี เพื่อระบุการทำงานเมื่อปุ่มถูกคลิก

โปรเจ็ค DatabaseDemo, ไฟล์ MainActivity.java

```
@Override
public void onClick(View v) {
    // สร้าง ContentValues
    ContentValues cv = new ContentValues();

    // ใส่ชื่อ
    cv.put(MyDbHelper.COL_NAME, etPersonName.getText().toString());
    // กำหนดรูปแบบของวันเวลา
    SimpleDateFormat dateFormat = new SimpleDateFormat(
            "yyyy-MM-dd HH:mm:ss");
    // ใส่วันเวลา ณ ขณะนั้นตามรูปแบบที่กำหนด
    cv.put(MyDbHelper.COL_DATE, dateFormat.format(new Date()));

    // เพิ่มข้อมูลลงฐานข้อมูล
    db.insert(MyDbHelper.TABLE_NAME, null, cv);

    // คิวรีฐานข้อมูลใหม่ และแจ้ง Adapter ว่าข้อมูลมีการเปลี่ยนแปลง เพื่ออัพเดท ListView
```

```
    cursor.requery();
    adapter.notifyDataSetChanged();
    // ลบข้อความใน EditText
    etPersonName.setText(null);
}
```

เมื่อคลิกปุ่ม เราจะนำข้อความ (ชื่อ) จาก EditText และวันเวลาในขณะนั้นเพิ่มลงฐานข้อมูล แล้วคิวรีฐานข้อมูลใหม่ เพื่ออัพเดท ListView ให้มีข้อมูลที่เพิ่งจะเพิ่มเข้าไปด้วย

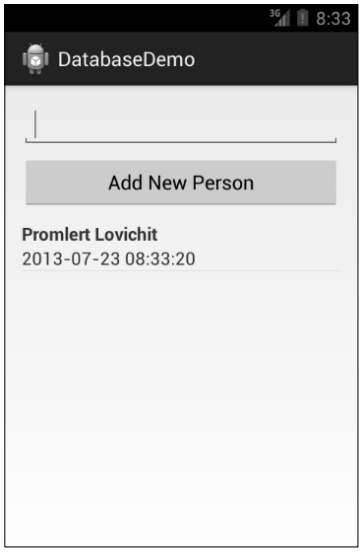
8 เพิ่มเมธอด onItemClick ในแอคทิวิตี เพื่อระบุการทำงานเมื่อไอเท็มใน ListView ถูกคลิก

```
@Override
public void onItemClick(AdapterView<?> parent, View v, int position,
                        long id) {
    // เลื่อนไปยังแถวข้อมูลที่จะลบ
    cursor.moveToPosition(position);
    // หา ID (ค่าของฟิลด์ _id) ของแถวข้อมูลนั้น
    String rowId = cursor.getString(0);
    // ลบแถวข้อมูลตาม ID ที่ระบุ
    db.delete(MyDbHelper.TABLE_NAME, "_id = ?", new String[] { rowId });
    // คิวรีฐานข้อมูลใหม่ และแจ้ง Adapter ว่าข้อมูลมีการเปลี่ยนแปลง เพื่ออัพเดท ListView
    cursor.requery();
    adapter.notifyDataSetChanged();
}
```

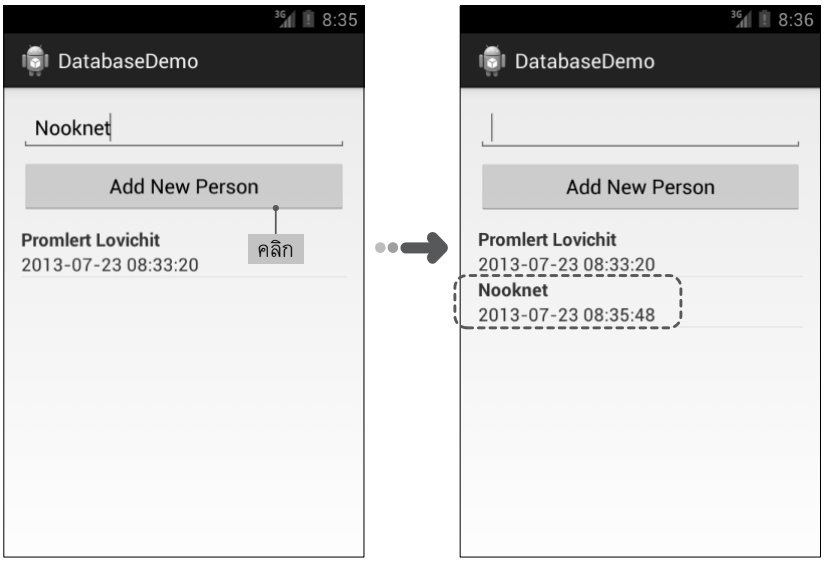
เมื่อคลิกไอเท็มใน ListView เราจะลบข้อมูลนั้นในฐานข้อมูล แล้วคิวรีฐานข้อมูลใหม่เพื่ออัพเดท ListView

ผลการรัน

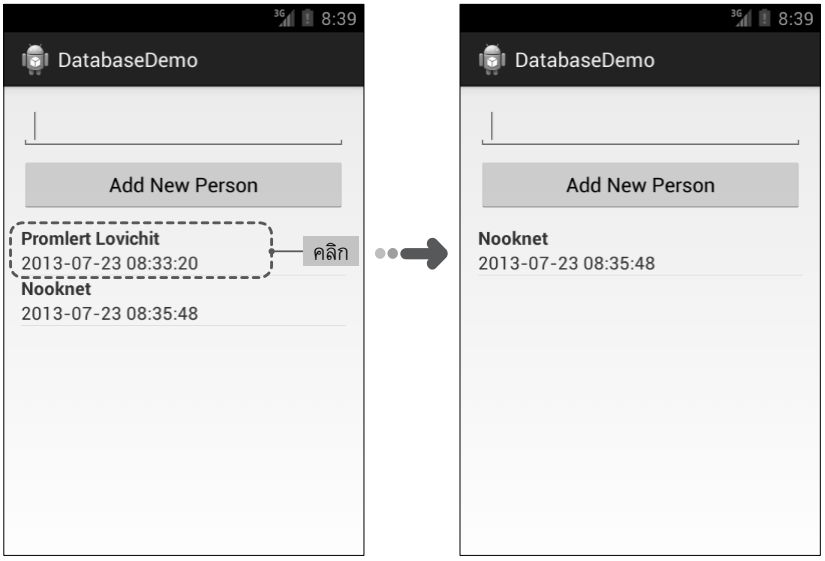
- **รันครั้งแรก**



- เพิ่มข้อมูล



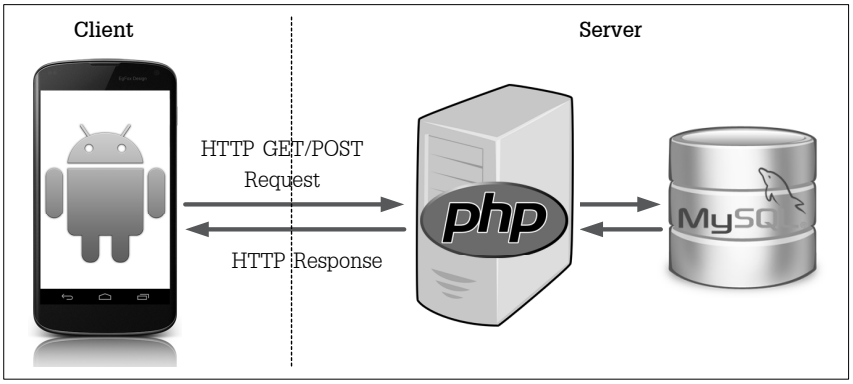
- ลบข้อมูล โดยคลิกไอเท็มใน `ListView`



การติดต่อฐานข้อมูล MySQL บนเซิร์ฟเวอร์

ฐานข้อมูล SQLite ที่อธิบายในบทก่อน จัดว่าเป็น Local Database เนื่องจากตัวฐานข้อมูลเก็บอยู่ในเครื่องแอนดรอยด์ของผู้ใช้เอง อย่างไรก็ตาม การพัฒนาแอพในชีวิตจริงนั้น บางครั้งเราอาจต้องการให้แอพแอนดรอยด์เก็บข้อมูลไว้บนเซิร์ฟเวอร์ในอินเทอร์เน็ต หรือเราเตรียมข้อมูลไว้บนเซิร์ฟเวอร์แล้วให้แอพแอนดรอยด์มาดึงข้อมูลเหล่านี้ไปใช้งาน ฯลฯ

ถึงแม้ว่าแอนดรอยด์จะไม่มี API สำหรับเข้าถึงฐานข้อมูลที่อยู่บนเครื่องเซิร์ฟเวอร์ หรือ Remote Database โดยตรง แต่ด้วยเทคโนโลยีของเว็บ เราสามารถใช้ Server-side script เป็นตัวเชื่อมโยงระหว่างแอพแอนดรอยด์กับฐานข้อมูลบนเซิร์ฟเวอร์ได้ โดยแอพแอนดรอยด์จะติดต่อกับ Server-side script ผ่านโปรโตคอล HTTP ดังรูป



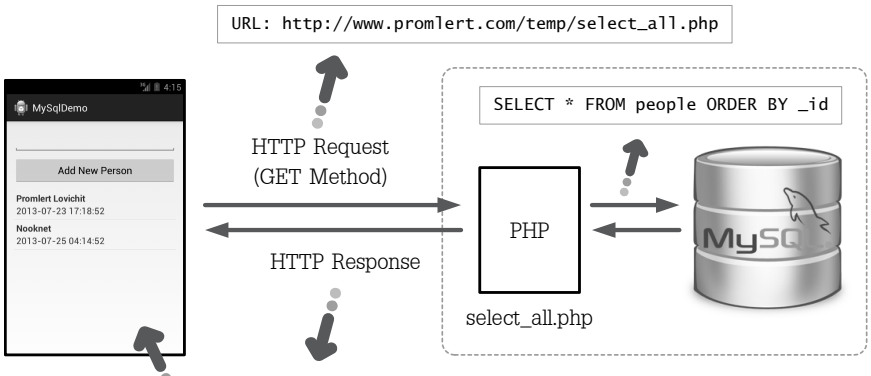
ฐานข้อมูลที่นิยมใช้บนเซิร์ฟเวอร์มากที่สุดฐานข้อมูลหนึ่งก็คือ MySQL ซึ่งมักใช้ร่วมกับ PHP ที่เป็นภาษาสำหรับเขียน Server-side script ดังนั้นในหัวข้อนี้จะแสดงตัวอย่างการสร้างแอพแอนดรอยด์ที่เชื่อมต่อกับ PHP+MySQL บนเซิร์ฟเวอร์

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้เป็นแอพที่มีหน้าตาและการทำงานเหมือนตัวอย่างที่แล้ว คือมี `EditText` และปุ่มสำหรับเพิ่มข้อมูลใหม่ลงฐานข้อมูล และมี `ListView` สำหรับแสดงข้อมูลทั้งหมดจากฐานข้อมูล แต่ความแตกต่างคือ ฐานข้อมูลที่ใช้ในตัวอย่างนี้จะอยู่บนเซิร์ฟเวอร์ในอินเทอร์เน็ต แทนที่จะอยู่ในเครื่องแอนดรอยด์ ดังนั้นการอ่าน/เขียนข้อมูลจึงต้องทำผ่านอินเทอร์เน็ต

เราจะเตรียมไฟล์ PHP 3 ไฟล์ไว้บนเซิร์ฟเวอร์ เพื่อเป็นตัวกลางระหว่างแอพแอนดรอยด์กับฐานข้อมูล MySQL โดยไฟล์ PHP ทั้งสามจะรับคำสั่งและข้อมูลที่จำเป็นจากแอพแอนดรอยด์ แล้วดำเนินการกับฐานข้อมูลตามหน้าที่ของแต่ละไฟล์

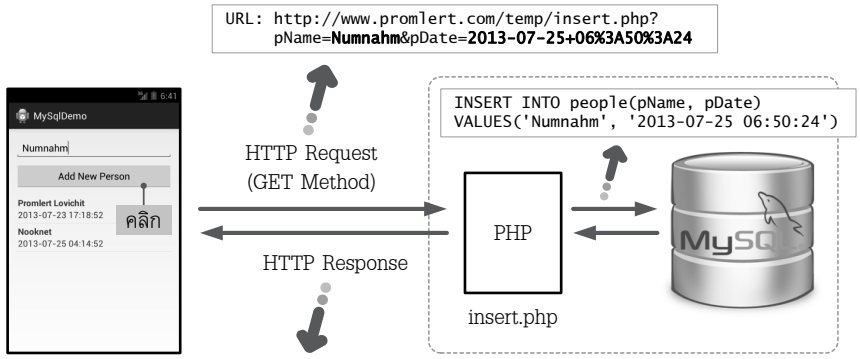
- **กรณีอ่านข้อมูลทั้งหมดจากฐานข้อมูล** แอพแอนดรอยด์จะเรียก (ส่ง HTTP Request) ไปยังไฟล์ select_all.php โดยไม่ระบุพารามิเตอร์ใดๆ จากนั้นไฟล์ PHP ดังกล่าวจะอ่านข้อมูลทั้งหมดจากฐานข้อมูล, แปลงข้อมูลเป็นรูปแบบ JSON แล้วส่งกลับไปให้แอพแอนดรอยด์ ดังรูป




```
{
  "people" : [
      { "_id" : "1",
        "pDate" : "2013-07-23 17:18:52",
        "pName" : "Promlert Lovichit"
      },
      { "_id" : "2",
        "pDate" : "2013-07-25 04:14:52",
        "pName" : "Nooknet"
      }
    ],
  "success" : 1
}
```

ตัวอย่างข้อมูล JSON ที่ select_all.php ส่งกลับมา

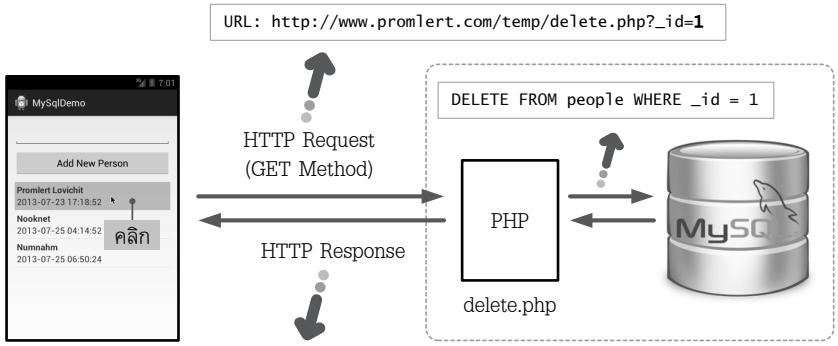
- **กรณีเพิ่มข้อมูลใหม่ลงฐานข้อมูล** แอพแอนดรอยด์จะเรียกไปยังไฟล์ insert.php พร้อมทั้งส่งข้อมูลที่ถูกกรอกใน `EditText` และวันเวลาขณะนั้นไปเป็นพารามิเตอร์ (โดยระบุ Query String ต่อท้าย URL ดังในรูป) ไฟล์ PHP ดังกล่าวจะนำข้อมูลนั้นเพิ่มลงฐานข้อมูล แล้วส่งผลลัพธ์เป็นข้อมูล JSON ที่บอกถึงความสำเร็จหรือล้มเหลวกลับไปให้แอพแอนดรอยด์ ดังรูปหน้าถัดไป




```
{
  "message" : "Data added successfully.",
  "success" : 1
}
```

ตัวอย่างข้อมูล JSON ที่ insert.php ส่งกลับมา

- **กรณีลบข้อมูลในฐานข้อมูล** แอพแอนดรอยด์จะเรียกไปยังไฟล์ delete.php พร้อมทั้งส่ง ID ของแถวข้อมูลที่จะลบไปเป็นพารามิเตอร์ ไฟล์ PHP ดังกล่าวจะลบข้อมูลนั้นในฐานข้อมูล แล้วส่งผลลัพธ์เป็นข้อมูล JSON ที่บอกถึงความสำเร็จหรือล้มเหลวกลับไปให้แอพแอนดรอยด์ ดังรูป




```
{
  "message" : "Data deleted successfully.",
  "success" : 1
}
```

ตัวอย่างข้อมูล JSON ที่ delete.php ส่งกลับมา

การเขียนโค้ดทางฝั่งเซิร์ฟเวอร์

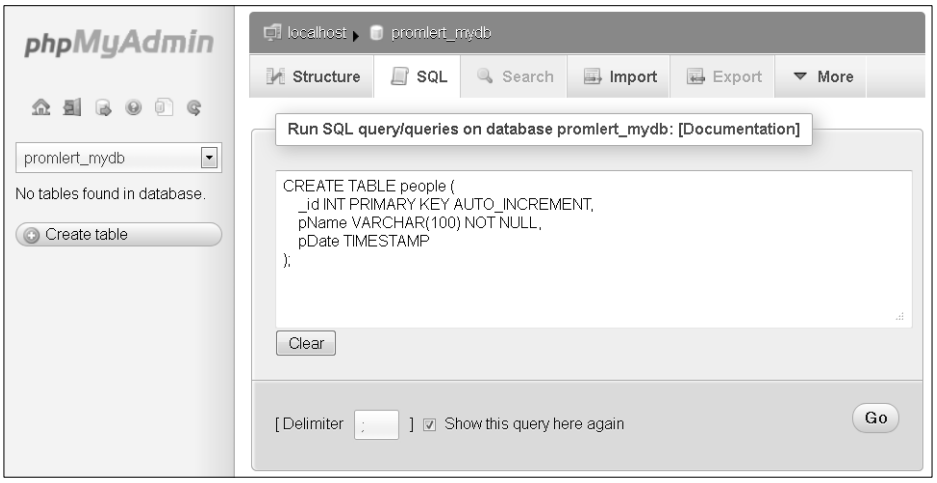
ที่ฝั่งเซิร์ฟเวอร์ เราจะต้องเตรียมไฟล์ PHP ต่างๆ รวมถึงสร้างฐานข้อมูล MySQL ไว้ให้เรียบร้อย (ซึ่งต่างจาก SQLite ที่เราสร้างฐานข้อมูลขึ้นมาในช่วง runtime เลย) ในที่นี้ผู้เขียนเตรียมไฟล์ PHP และฐานข้อมูล MySQL ไว้ที่เว็บไซต์ของผู้เขียนเอง คือที่ `www.promlert.com` แต่หากคุณผู้อ่านไม่มีเว็บไซต์ส่วนตัวก็อาจใช้วิธีจำลองเครื่องพีซีของตัวเองเป็นเซิร์ฟเวอร์ โดยติดตั้งชุดโปรแกรม AppServ (`www.appservnetwork.com`) ก็ได้เช่นกัน สำหรับรายละเอียดการติดตั้งจะไม่อธิบายในที่นี้

1 ก่อนอื่นให้สร้างฐานข้อมูล **promlert_mydb** (หรือชื่ออื่นๆตามต้องการ) และสร้างเทเบิล **people** ไว้ในฐานข้อมูลนั้น โดยใช้คำสั่ง SQL ดังนี้

```
CREATE DATABASE promlert_mydb;

CREATE TABLE people (
    _id INT PRIMARY KEY AUTO_INCREMENT,
    pName VARCHAR(100) NOT NULL,
    pDate TIMESTAMP
);
```

วิธีง่ายๆในการรันคำสั่ง SQL จะทำได้โดยใช้แท็บ SQL ของโปรแกรม phpMyAdmin ดังรูป ซึ่งชุดโปรแกรม AppServ และบริการ Web hosting ส่วนใหญ่จะมีโปรแกรมนี้ให้ใช้งาน

หมายเหตุ : ในรูปนี้ผู้เขียนสร้างฐานข้อมูล promlert_mydb ไว้ก่อนแล้ว ก็เลยพิมพ์เฉพาะคำสั่งที่ใช้สร้างเทเบิล people



2 สร้างไฟล์ PHP ชื่อ db_config.php ที่กำหนดค่าคงที่ต่างๆสำหรับการติดต่อฐานข้อมูล

ไฟล์ db__config.php

```
<?php

define('DB_USER', "admin");                 // ชื่อบัญชีผู้ใช้ฐานข้อมูล
define('DB_PASSWORD', "123456");            // รหัสผ่าน
define('DB_DATABASE', "promlert_mydb");     // ชื่อฐานข้อมูล
define('DB_SERVER', "localhost");           // ชื่อเซิร์ฟเวอร์ฐานข้อมูล (MySQL Server)

?>
```

3 สร้างไฟล์ select_all.php ที่ใช้อ่าน (คิวรี) ข้อมูลทั้งหมดจากฐานข้อมูล แล้วส่งข้อมูลเหล่านั้นออกไปในรูปแบบ JSON

ไฟล์ select_all.php

```
<?php

// รวมโค้ดจากไฟล์ db_config.php เข้ามา
require_once __DIR__ . '/db_config.php';
// เชื่อมต่อฐานข้อมูล
$db = new mysqli(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE);

// ถ้าการเชื่อมต่อผิดพลาดจะส่ง Error Message ออกไปแล้วจบการทำงานทันที
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

// สร้างอาร์เรย์ว่างๆสำหรับเก็บข้อมูลที่สุดท้ายแล้วจะแปลงเป็น JSON
$response = array();

// คิวรีข้อมูลทุกฟิลด์และทุกแถวจากเทเบิล people โดยให้เรียงลำดับตาม ID
if ($result = $db->query("SELECT * FROM people ORDER BY _id")) {
    $rowCount = $result->num_rows;  // จำนวนแถวข้อมูลใน Result Set

    if ($rowCount > 0) {  // ถ้าจำนวนแถวข้อมูลใน Result Set มากกว่า 0
        // สร้างอาร์เรย์ย่อยว่างๆขึ้นมาในอาร์เรย์ $response
        $response["people"] = array();

        // วนลูปเพื่อเข้าถึงแถวข้อมูลทุกแถวใน Result Set
        while ($row = $result->fetch_assoc()) {
            // สร้างอาร์เรย์ว่างๆสำหรับใช้ชั่วคราวภายในคำสั่ง while นี้
            $people = array();

            // อ่านค่าจากฟิลด์ทั้งสามของแต่ละแถวข้อมูลใน Result Set มาเก็บลงอาร์เรย์ชั่วคราว
            $people["_id"] = $row["_id"];
            $people["pName"] = $row["pName"];
            $people["pDate"] = $row["pDate"];

            // นำข้อมูลจากอาร์เรย์ชั่วคราวใส่ลงในอาร์เรย์ย่อยภายในอาร์เรย์ $response
            array_push($response["people"], $people);
        }
        $response["success"] = 1;
    } else {  // ถ้าจำนวนแถวข้อมูลใน Result Set เท่ากับ 0 (เทเบิล people ไม่มีข้อมูล)
        $response["success"] = 0;
        $response["message"] = "The database is empty.";
```

```
    }

    $result->close();  // ปิด Result Set
} else {
    $response["success"] = 0;
    $response["message"] = "An error occurred while retrieving data.";
}

$db->close();  // ปิดการเชื่อมต่อฐานข้อมูล
// นำข้อมูลในอาร์เรย์ $response มาจัดรูปแบบเป็น JSON แล้วส่งออกไป
echo json_encode($response);

?>
```

4 สร้างไฟล์ insert.php ที่ใช้เพิ่มข้อมูลลงฐานข้อมูล

ไฟล์ insert.php

```
<?php

// สร้างอาร์เรย์ว่างๆสำหรับเก็บข้อมูลที่สุดท้ายแล้วจะแปลงเป็น JSON
$response = array();

// ตรวจสอบว่ามีการส่งพารามิเตอร์ pName และ pDate มาให้หรือไม่
if (isset($_GET['pName']) && isset($_GET['pDate'])) {
    $name = $_GET['pName'];
    $date = $_GET['pDate'];

    // รวมโค้ดจากไฟล์ db_config.php เข้ามา
    require_once __DIR__ . '/db_config.php';
    // เชื่อมต่อฐานข้อมูล
    $db = new mysqli(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE);

    // ถ้าการเชื่อมต่อผิดพลาดจะส่ง Error Message ออกไปแล้วจบการทำงานทันที
    if (mysqli_connect_errno()) {
        printf("Connect failed: %s\n", mysqli_connect_error());
        exit();
    }

    // เพิ่มข้อมูลลงในเทเบิล people ของฐานข้อมูล
    if ($result = $db->query("INSERT INTO people(pName, pDate)
                              VALUES('$name', '$date')")) {  // เพิ่มข้อมูลสำเร็จ
        $response["success"] = 1;
        $response["message"] = "Data added successfully.";
    } else {
```

```
        $response["success"] = 0;
        $response["message"] = "An error occurred while adding data.";
    }

    $db->close();  // ปิดการเชื่อมต่อฐานข้อมูล
} else {  // เพิ่มข้อมูลไม่สำเร็จ
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing.";
}

// นำข้อมูลในอาร์เรย์ $response มาจัดรูปแบบเป็น JSON แล้วส่งออกไป
echo json_encode($response);

?>
```

5 สร้างไฟล์ delete.php ที่ใช้ลบข้อมูลในฐานข้อมูลตาม ID ที่ระบุ

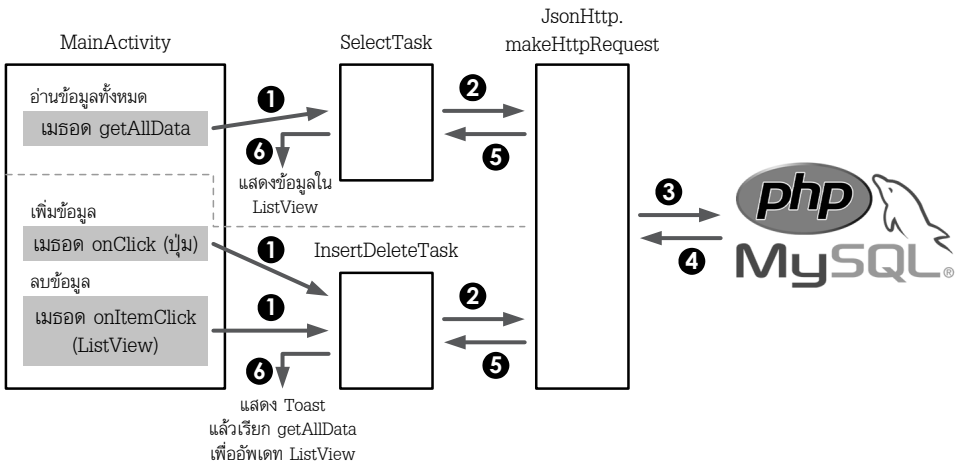
ไฟล์ delete.php

```
<?php

// สร้างอาร์เรย์ว่างๆสำหรับเก็บข้อมูลที่สุดท้ายแล้วจะแปลงเป็น JSON
$response = array();

// ตรวจสอบว่ามีการส่งพารามิเตอร์ _id มาให้หรือไม่
if (isset($_GET['_id'])) {
    $id = $_GET['_id'];

    // รวมโค้ดจากไฟล์ db_config.php เข้ามา
    require_once __DIR__ . '/db_config.php';
    // เชื่อมต่อฐานข้อมูล
    $db = new mysqli(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE);

    // ถ้าการเชื่อมต่อผิดพลาดจะส่ง Error Message ออกไปแล้วจบการทำงานทันที
    if (mysqli_connect_errno()) {
        printf("Connect failed: %s\n", mysqli_connect_error());
        exit();
    }

    // ลบข้อมูลตาม ID ที่ระบุจากเทเบิล people ของฐานข้อมูล
    if ($result = $db->query("DELETE FROM people WHERE _id = $id")) {
        $response["success"] = 1;
        $response["message"] = "Data deleted successfully.";
    } else {  // ลบข้อมูลไม่สำเร็จ
        $response["success"] = 0;
```

```
        $response["message"] = "An error occurred while deleting data.";
    }

    $db->close();  // ปิดการเชื่อมต่อฐานข้อมูล

} else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing.";
}

// นำข้อมูลในอาร์เรย์ $response มาจัดรูปแบบเป็น JSON แล้วส่งออกไป
echo json_encode($response);

?>
```

การเขียนโค้ดทางฝั่งแอนดรอยด์

ภาพรวมในการทำงานที่ฝั่งแอนดรอยด์จะเป็นดังรูป



นอกจาก Layout File กับแอคทิวิตีแล้วจะมีการสร้างไฟล์อื่นๆ ขึ้นมา ดังนี้

- **ไฟล์ JsonHttp.java (คลาส `JsonHttp`)** มีเมธอด `makeHttpRequest` สำหรับเรียกไปยัง URL ใดๆ และจะส่งผลลัพธ์ที่ได้จาก URL ออกไปเป็นค่าสตริง เราจะใช้เมธอดนี้เรียกไปยังไฟล์ PHP แล้วส่งค่าสตริงในรูปแบบ JSON ที่ได้จาก PHP ออกไป
- **ไฟล์ SelectTask.java (คลาส `SelectTask`)** คือ ซับคลาสของ `AsyncTask` (ดูบทที่ 7) ที่ใช้อ่านข้อมูลทั้งหมดจากฐานข้อมูล แล้วแสดงข้อมูลเหล่านั้นใน `ListView` (โดยใช้ Adapter ชนิด `SimpleAdapter` เป็นตัวกลาง)

- **ไฟล์ InsertDeleteTask.java (คลาส `InsertDeleteTask`)** คือซับคลาสของ `AsyncTask` ที่ใช้เพิ่มข้อมูลและลบข้อมูลในฐานข้อมูล แล้วแสดงข้อความที่บอกถึงความสำเร็จหรือล้มเหลวเป็น Toast ออกมา
- **ไฟล์ item.xml ในโฟลเดอร์ res\layout** กำหนด Layout ของแต่ละไอเท็มใน `ListView` ตัวอย่างนี้ไม่ได้ใช้ Layout สำเร็จรูปที่แอนดรอยด์เตรียมมาให้เหมือนตัวอย่างที่แล้ว เหตุผลคือเราต้องการอ่านค่า ID (ฟิลด์ _id ในฐานข้อมูล) มาใส่ลง `ListView` ด้วย (แต่ซ่อน ID ไว้ไม่ให้แสดงออกมา) เพื่อจะได้มี ID ไว้ระบุตอนลบข้อมูล ส่วนตัวอย่างที่แล้วเรามี Result Set อยู่ในออบเจ็ค `Cursor` ซึ่งเป็นตัวแปรในแอคทิวิตี เราจึงสามารถหา ID ได้จากออบเจ็ค `Cursor`

ขั้นตอนการเขียนโค้ดฝั่งแอนดรอยด์

1 กำหนด Layout ของหน้าจอ (เหมือนตัวอย่างที่แล้วทุกประการ)

โปรเจ็ค MySqlDemo, ไฟล์ 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="vertical"
    android:padding="16dp" >

    <EditText
        android:id="@+id/person_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/add_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="12dp"
        android:layout_marginTop="8dp"
        android:text="Add New Person" />

    <ListView
        android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

2 สร้างไฟล์ item.xml ในโฟลเดอร์ res\layout เพื่อกำหนด Layout ของแต่ละไอเท็มใน `ListView`

โปรเจ็ค MySqlDemo, ไฟล์ item.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="4dp"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/item_id"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:visibility="gone" />

    <TextView
        android:id="@+id/item_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="16sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/item_date"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="16sp" />

</LinearLayout>
```

สังเกตว่าเรากำหนด visibility ของ `TextView` อันแรกเป็น gone ซึ่งหมายถึงไม่ต้องแสดง `TextView` นี้ และไม่ต้องกันพื้นที่ไว้ให้ด้วย (แต่หากกำหนดเป็น invisible จะกันพื้นที่ไว้ให้)

ถึงแม้ `TextView` นี้จะไม่ถูกแสดงผลออกมา แต่เราสามารถกำหนดค่าและเรียกใช้ค่าของมันโดยใช้เมธอด `setText/getText` รวมถึงการผูกข้อมูลด้วย Adapter ในโค้ดจาวาได้

3 สร้างคลาส `JsonHttp` (ไฟล์ JsonHttp.java)

โปรเจ็ค MySqlDemo, ไฟล์ JsonHttp.java

```
package com.example.mysqldemo;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
```

```
import java.net.URL;

public class JsonHttp {

    // เมธอดสำหรับเรียกไปยัง URL ที่ระบุ แล้วนำผลลัพธ์ที่ได้จาก URL นั้น ส่งคืนเป็นค่าสตริงออกไป
    public static String makeHttpRequest(String url) {
        String strResult = "";

        try {
            URL u = new URL(url);
            // เชื่อมต่อ URL
            HttpURLConnection con = (HttpURLConnection) u.openConnection();
            // เรียกเมธอด readStream เพื่ออ่านผลลัพธ์จาก URL มาเก็บลงตัวแปรสตริง
            strResult = readStream(con.getInputStream());
        } catch (Exception e) {
            e.printStackTrace();
        }

        return strResult;
    }

    // เมธอดสำหรับอ่านข้อมูลจาก InputStream
    private static String readStream(InputStream in) {
        BufferedReader reader = null;
        StringBuilder sb = new StringBuilder();
        try {
            /* สร้าง InputStreamReader ครอบ InputStream แล้วสร้าง BufferedReader
               ครอบ InputStreamReader อีกที เพื่อใช้เมธอด readLine อ่านข้อมูลทีละบรรทัด */
            reader = new BufferedReader(new InputStreamReader(in));
            String line;
            // อ่านข้อมูลทีละบรรทัด แล้วเก็บรวบรวมไว้ใน StringBuilder
            while ((line = reader.readLine()) != null) {
                sb.append(line + "\n");
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (reader != null) {
                try {
                    reader.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
```

```
        }
        return sb.toString();  // ส่งคืนค่าสตริงใน StringBuilder ออกไป
    }
}
```

4 สร้างคลาส `SelectTask` (ไฟล์ SelectClass.java)

โปรเจ็ค MySqlDemo, ไฟล์ SelectTask.java

```
package com.example.mysqldemo;

import java.util.ArrayList;
import java.util.HashMap;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.content.Context;
import android.os.AsyncTask;
import android.widget.ListAdapter;
import android.widget.SimpleAdapter;
import android.widget.Toast;

public class SelectTask extends AsyncTask<Void, Void, String> {

    private Context mContext;
    private String mUrl;

    // คอนสตรัคเตอร์
    public SelectTask(Context context, String url) {
        super();

        mContext = context; ❶
        mUrl = url;
    }

    @Override
    protected String doInBackground(Void... params) {
        String jsonString = JsonHttp.makeHttpRequest(mUrl); ❷
        return jsonString; ❸
    }

    @Override
    protected void onPostExecute(String jsonString) {

        ArrayList<HashMap<String, String>> peopleList;
```

```
try {
    JSONObject json = new JSONObject(jsonString); ❹
    int success = json.getInt("success");

    if (success == 1) {  // มีข้อมูลในฐานข้อมูล
        JSONArray people = json.getJSONArray("people");
        peopleList = new ArrayList<HashMap<String, String>>();

        for (int i = 0; i < people.length(); i++) {
            JSONObject person = people.getJSONObject(i);

            String id = person.getString("_id");
            String name = person.getString("pName");
            String date = person.getString("pDate");

            HashMap<String, String> map =
                    new HashMap<String, String>();
            map.put("_id", id);
            map.put("pName", name);
            map.put("pDate", date);                          ❺

            peopleList.add(map); ❻

            String[] keys = new String[] {
                    "_id", "pName", "pDate" };
            int[] views = new int[] {
                    R.id.item_id,
                    R.id.item_name,
                    R.id.item_date };

            ListAdapter adapter = new SimpleAdapter(
                    mContext, peopleList,
                    R.layout.item, keys, views);

            ((MainActivity) mContext).list
                    .setAdapter(adapter);                    ❼
        }
    } else if (success == 0) {  // ไม่มีข้อมูลในฐานข้อมูล
        ((MainActivity) mContext).list.setAdapter(null);
        String msg = json.getString("message");
        Toast.makeText(mContext, msg, Toast.LENGTH_SHORT).show();
    }
} catch (JSONException e) {
```

```
            e.printStackTrace();
        }
    }
}
```

คอนสตรัคเตอร์ของคลาส `SelectTask` นี้มีพารามิเตอร์ 2 ตัว สำหรับระบุคอนเท็กซ์ และ URL ของไฟล์ PHP ที่จะเรียกใช้ เรานำค่าของพารามิเตอร์ทั้งสองมาเก็บลงตัวแปรระดับคลาส (`mContext` และ `mUrl`) ❶ เพื่อให้เรียกใช้ได้จากทุกเมธอดในคลาสนี้

ดังที่อธิบายในบทที่ 7 แล้วว่า การทำงานของ `AsyncTask` จะเริ่มต้นที่เมธอด `doInBackground` ซึ่งในที่นี้เราเรียกต่อไปยังเมธอด `makeHttpRequest` เพื่อส่ง HTTP Request ไปที่ URL นั้น ❷ และเมื่อได้ผลลัพธ์จาก `makeHttpRequest` แล้ว (ผลลัพธ์คือสตริงในรูปแบบ JSON) ก็จะส่งคืนผลลัพธ์นั้นออกไปจาก `doInBackground` ❸

ผลลัพธ์ของ `doInBackground` จะถูกส่งไปให้ `onPostExecute` จากนั้น ภายใน `onPostExecute` เราทำการ parse สตริง JSON ❹ แล้วอ่านข้อมูลแต่ละฟิลด์ของแถวข้อมูลหนึ่งๆ มาเก็บลง `HashMap` (ตัวแปร `map`) ❺ แล้วเก็บ `HashMap` ลงใน `ArrayList` (ตัวแปร `peopleList`) ❻ สุดท้ายจึงผูก `ArrayList` เข้ากับ `ListView` โดยใช้ `SimpleAdapter` เพื่อแสดงข้อมูลทั้งหมดออกมาใน `ListView` ❼

5 สร้างคลาส `InsertDeleteTask` (ไฟล์ InsertDeleteClass.java)

โปรเจ็ค MySqlDemo, ไฟล์ InsertDeleteTask.java

```
package com.example.mysqldemo;

import java.util.List;

import org.apache.http.NameValuePair;
import org.apache.http.client.utils.URLEncodedUtils;
import org.json.JSONException;
import org.json.JSONObject;

import android.content.Context;
import android.os.AsyncTask;
import android.widget.Toast;

public class InsertDeleteTask extends AsyncTask<Void, Void, String> {

    private Context mContext;
    private String mUrl;

    // คอนสตรัคเตอร์
    public InsertDeleteTask(Context context, String url,
                            List<NameValuePair> params) {
```

```
        super();

        mContext = context;
        mUrl = url;

        // ระบุพารามิเตอร์ (Query String) ต่อท้าย URL
        if (params != null) {
            String paramString = URLEncodedUtils.format(params, "utf-8");
            mUrl += "?" + paramString;
        }
    }

    @Override
    protected String doInBackground(Void... params) {
        String jsonString = JsonHttp.makeHttpRequest(mUrl); ❶
        return jsonString;
    }

    @Override
    protected void onPostExecute(String jsonString) {
        try {
            JSONObject json = new JSONObject(jsonString);

            // แสดงข้อความสถานะที่ส่งมาจากไฟล์ PHP ออกมาใน Toast
            String msg = json.getString("message"); ❷
            Toast.makeText(mContext, msg, Toast.LENGTH_SHORT).show(); ❸
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}
```

คอนสตรัคเตอร์ของคลาส `InsertDeleteTask` นี้ มีพารามิเตอร์ 3 ตัว สำหรับระบุคอนเท็กซ์, URL ของไฟล์ PHP ที่จะเรียกใช้ และ `List` ของพารามิเตอร์ที่จะส่งไปให้ไฟล์ PHP โดยระบุต่อท้าย URL การทำงานของ `InsertDeleteTask` จะส่ง HTTP Request ไปยัง URL ที่ระบุ ❶ จากนั้นเมื่อได้ผลลัพธ์จาก URL ก็จะอ่านข้อความสถานะที่บอกถึงความสำเร็จหรือล้มเหลวในการทำงาน (การเพิ่มหรือลบข้อมูล) ❷ มาแสดงใน Toast ❸

6 เพิ่มโค้ดในแอคทิวิตีจนเป็นดังนี้

โปรเจ็ค MySqlDemo, ไฟล์ MainActivity.java

```
package com.example.mysqldemo;

// บรรทัด import ต่างๆ...

public class MainActivity extends Activity
        implements View.OnClickListener, AdapterView.OnItemClickListener {

    EditText etPersonName;
    Button btnAddPerson;
    ListView list;

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

        etPersonName = (EditText) findViewById(R.id.person_name);
        btnAddPerson = (Button) findViewById(R.id.add_button);
        btnAddPerson.setOnClickListener(this);
        list = (ListView) findViewById(R.id.list);
        list.setOnItemClickListener(this);
    }

    // เมื่อแอคทิวิตีแสดงผลออกมา ให้อ่านข้อมูลทั้งหมดจากฐานข้อมูลมาแสดงใน ListView
    @Override
    public void onResume() {
        super.onResume();

        getAllData();
    }

    @Override
    public void onClick(View v) {
        // กำหนดรูปแบบวันเวลา
        SimpleDateFormat dateFormat = new SimpleDateFormat(
                "yyyy-MM-dd HH:mm:ss");
        // หาวันเวลาปัจจุบัน แล้วจัดรูปแบบตามที่กำหนด
        String strDate = dateFormat.format(new Date());

        // สร้าง List สำหรับเก็บข้อมูลที่จะส่งเป็นพารามิเตอร์ไปให้ไฟล์ PHP
        List<NameValuePair> params = new ArrayList<NameValuePair>();
        // ใส่ชื่อที่กรอกใน EditText ลงใน List
```

```
        params.add(new BasicNameValuePair("pName",
                etPersonName.getText().toString()));
        // ใส่วันเวลาปัจจุบันที่จัดรูปแบบแล้วลงใน List
        params.add(new BasicNameValuePair("pDate", strDate));

        // เรียกไปยัง InsertDeleteTask เพื่อเพิ่มข้อมูลลงฐานข้อมูล โดยระบุ URL และพารามิเตอร์
        String url = "http://www.promlert.com/temp/insert.php";
        InsertDeleteTask task = new InsertDeleteTask(this, url, params);
        task.execute();

        getAllData();  // อัพเดทข้อมูลใน ListView
        etPersonName.setText(null);  // ลบข้อความใน EditText
    }

    @Override
    public void onItemClick(AdapterView<?> parent, View v, int position,
                            long id) {
        // หา ID ของแถวข้อมูลที่จะลบ
        String rowId = ((TextView) v.findViewById(R.id.item_id)).getText()
                        .toString();
        // สร้าง List สำหรับเก็บข้อมูลที่จะส่งเป็นพารามิเตอร์ไปให้ไฟล์ PHP
        List<NameValuePair> params = new ArrayList<NameValuePair>();
        // ใส่ ID ของแถวข้อมูลที่จะลบลงใน List
        params.add(new BasicNameValuePair("_id", rowId));

        // เรียกไปยัง InsertDeleteTask เพื่อลบข้อมูลฐานข้อมูล โดยระบุ URL และพารามิเตอร์
        String url = "http://www.promlert.com/temp/delete.php";
        InsertDeleteTask task = new InsertDeleteTask(this, url, params);
        task.execute();

        getAllData();  // อัพเดทข้อมูลใน ListView
    }

    private void getAllData() {
        // เรียกไปยัง SelectTask เพื่ออ่านข้อมูลทั้งหมดจากฐานข้อมูล
        String url = "http://www.promlert.com/temp/select_all.php";
        SelectTask task = new SelectTask(this, url);
        task.execute();
    }
}
```

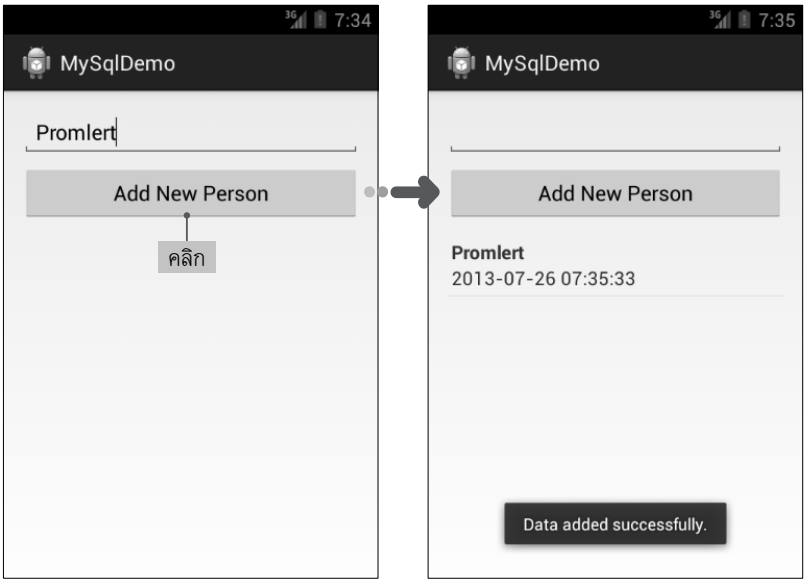
7 เพิ่มบรรทัดต่อไปนี้ในไฟล์ AndroidManifest.xml โดยใส่ไว้ก่อนแท็ก `<application>` เพื่อขอสิทธิ์ในการเข้าถึงอินเทอร์เน็ต

```
<uses-permission android:name="android.permission.INTERNET" />
```

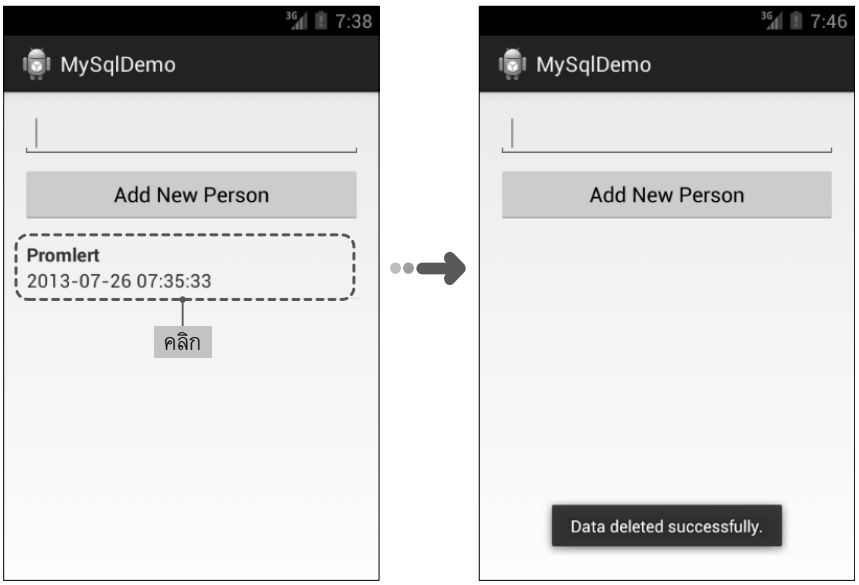
ผลการรัน

ตัวอย่างนี้เมื่อรันแอพครั้งแรกจะไม่มีข้อมูลใน `ListView` เพราะเราไม่ได้เพิ่มข้อมูลไว้ตั้งแต่แรก เหมือนตัวอย่างที่แล้ว

- **เพิ่มข้อมูล**



- **ลบข้อมูล**



ในมุมมองของการใช้งาน ตัวอย่างนี้กับตัวอย่างที่แล้วจะไม่ต่างกัน แต่แน่นอนว่าจริงๆแล้ว ตัวอย่างนี้ มีรายละเอียดเบื้องหลังซับซ้อนกว่ามาก เพราะการทำงานกับฐานข้อมูล MySQL จะต้องมีการติดต่อสื่อสารระหว่างแอพแอนดรอยด์กับไฟล์ PHP และระหว่างไฟล์ PHP กับฐานข้อมูล MySQL ตรงนี้แสดงให้เห็นว่าการทำงานของแอพที่เราดูเหมือนง่ายๆนั้น บางครั้งต้องแลกมาด้วยหยาดเหงื่อแรงงานของนักพัฒนาอย่างมากเลย