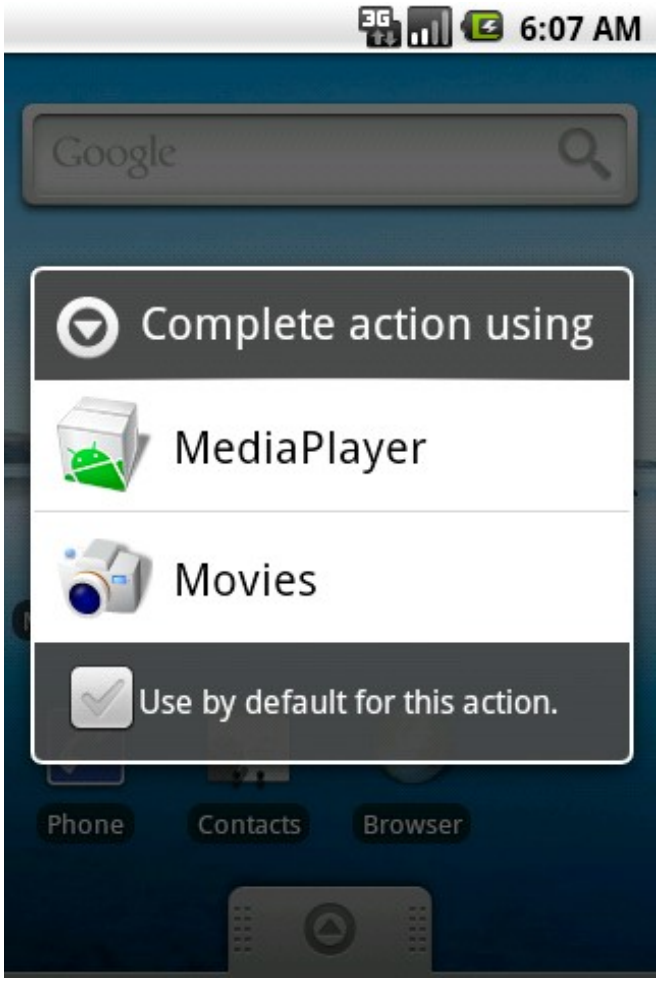
多媒体应用开发示范

Android 多媒体相关的接口在 android.media 包中。主要包含了以下几个部分的内容:

- 媒体播放
- 媒体录制
- 媒体扫描

媒体播放器是 Android 中常用的程序，可以进行音频和视频的播放。要实现一个媒体播放器，不仅具有播放功能，还需要获得媒体的接口，这样在恰当的地方调用需要播放媒体的时候，将可以调用到这个程序。

```
    <activity android:name="SimpleMediaPlayer1"
android:label="MediaPlayer">
                <intent-filter>
                    <action android:name="android.intent.action.VIEW" />
                    <category
android:name="android.intent.category.DEFAULT" />
                    <data android:mimeType="video/*" />
                    <data android:mimeType="audio/*" />
                 </intent-filter>
    <activity>
```



```
public class SimpleMediaPlayer1 extends Activity {
     private static final String TAG = "SimpleMediaPlayer1";
   public VideoView mVideoView;
   @Override
     public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);

         Intent intent = getIntent();    //  获取 Intent
         Uri uri = intent.getData();     //  从 Intent 中获取数据
         String path = uri.getPath();   //  获取路径
         setTitle(path);                        //  设置标题为媒体的路径
         mVideoView = new VideoView(this);   //  建立一个 VideoView
         setContentView(mVideoView);            //  设置 VideoView
         mVideoView.setVideoURI(uri);         //  设置 URI
         mVideoView.setMediaController(new MediaController(this));
                                                //  设置媒体控制条

         Log.v(TAG, "start");
         mVideoView.start();                        //  开始播放

     }
}
```


5:15 AM
/sdcard/test.mp4
COMPAQ


5:15 AM
/sdcard/test.mp4
00:10
COMPAQ


5:16 AM
/sdcard/test.mp4
00:10
COMPAQ
00:01
00:37

在使用媒体播放器的时候，更为直接使用 MediaPlayer 类， MediaPlayer 需要使用一个 SurfaceView 作为输出设备。

布局文件:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <SurfaceView android:id="@+id/SurfaceView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center">
    </SurfaceView>
</LinearLayout>
```

JAVA 源代码文件：

```
package com.android.basicapp;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;

import android.view.SurfaceHolder;
import android.view.SurfaceView;

import android.media.MediaPlayer;
import android.media.MediaPlayer.OnBufferingUpdateListener;
import android.media.MediaPlayer.OnCompletionListener;
import android.media.MediaPlayer.OnInfoListener;
import android.media.MediaPlayer.OnErrorListener;
import android.media.MediaPlayer.OnPreparedListener;
import android.media.MediaPlayer.OnVideoSizeChangedListener;

import java.io.*;
import java.util.Date;
import android.widget.MediaController;

import android.view.Menu;
import android.view.Gravity;
import android.view.MenuItem;

import android.util.Log;
```

```
public class SimpleMediaPlayer2 extends Activity {
    private static final String TAG = "SimpleMediaPlayer2";
    public static final int STOP_MENU_ID   = Menu.FIRST;
    public static final int START_MENU_ID  = Menu.FIRST + 1;

    private SurfaceView mSurfaceView;
    private SurfaceHolder mSurfaceHolder = null;
    private MediaPlayer mMediaPlayer;
    private String mPath;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Intent intent = getIntent();
        Uri uri = intent.getData();
        mPath = uri.getPath();

        setTitle(mPath);

        setContentView(R.layout.testsurfaceview);
        mSurfaceView = (SurfaceView)findViewById(R.id.SurfaceView);
        mSurfaceHolder = mSurfaceView.getHolder();
         mSurfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);

        mSurfaceHolder.addCallback(new SHCallback());
    }
```

```
public class SHCallback implements SurfaceHolder.Callback
   {
       public void surfaceChanged(SurfaceHolder holder, int format,
                       int width, int height) {
         }
           public void surfaceCreated(SurfaceHolder holder) {
               Log.v(TAG, "surfaceCreated");
               try {
                   mMediaPlayer = new MediaPlayer();
                  mMediaPlayer.setDataSource(mPath);
                   mMediaPlayer.setDisplay(mSurfaceHolder);
                   mMediaPlayer.prepare();
               } catch (Exception e) {
                  Log.e(TAG, "error: " + e.getMessage(), e);
               }
       }
         public void surfaceDestroyed(SurfaceHolder holder) {}
   }
```

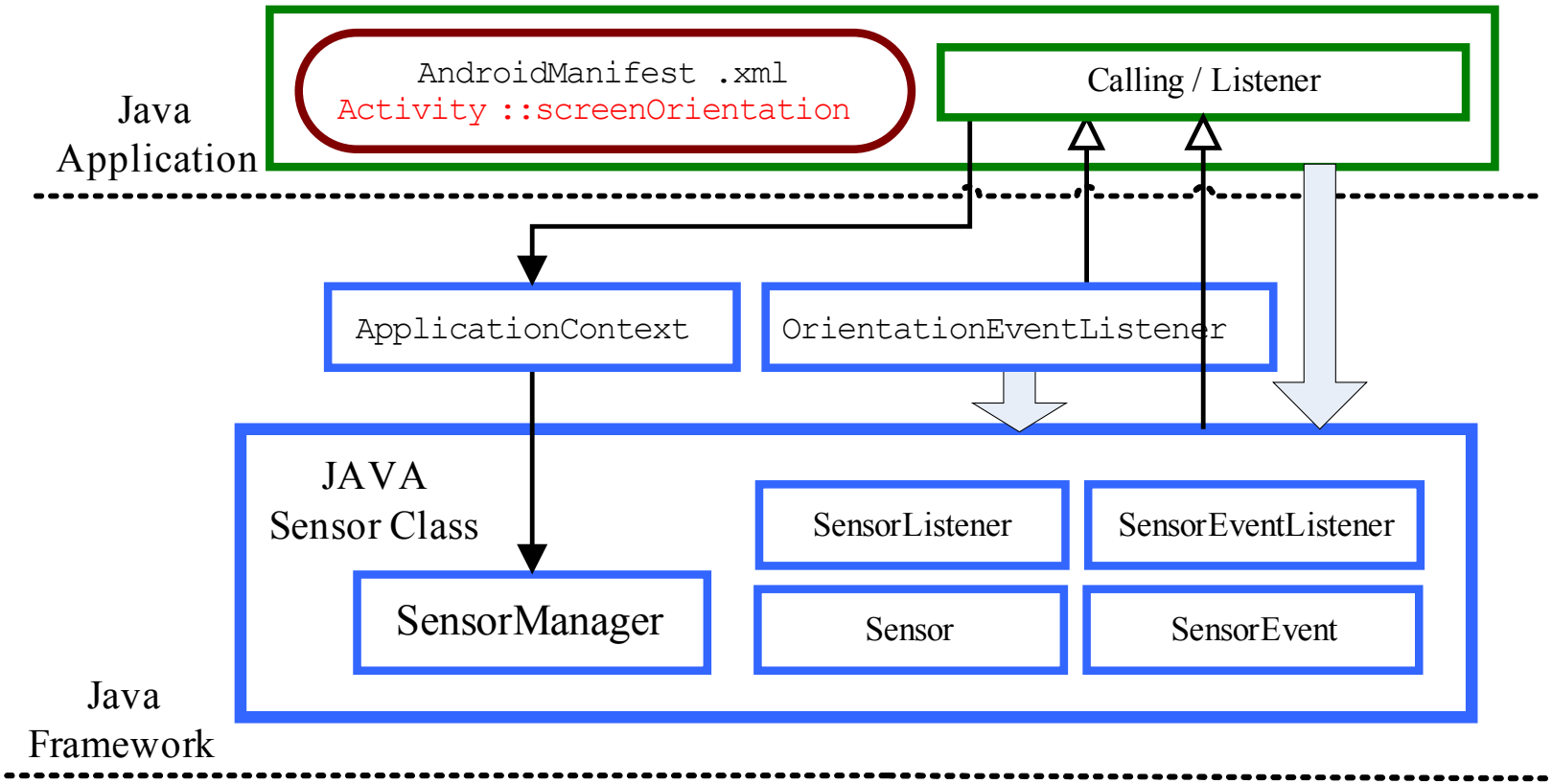
```
class  SimpleMediaPlayerOnCompletionListener  implements
                                    MediaPlayer.OnCompletionListener{
                public void onCompletion(MediaPlayer mp){
                }
        }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
        menu.add(0, STOP_MENU_ID, 0, R.string.pause);
        menu.add(0, START_MENU_ID, 0, R.string.start);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case STOP_MENU_ID:
            mMediaPlayer.pause();
            return true;
        case START_MENU_ID:
            mMediaPlayer.start();
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

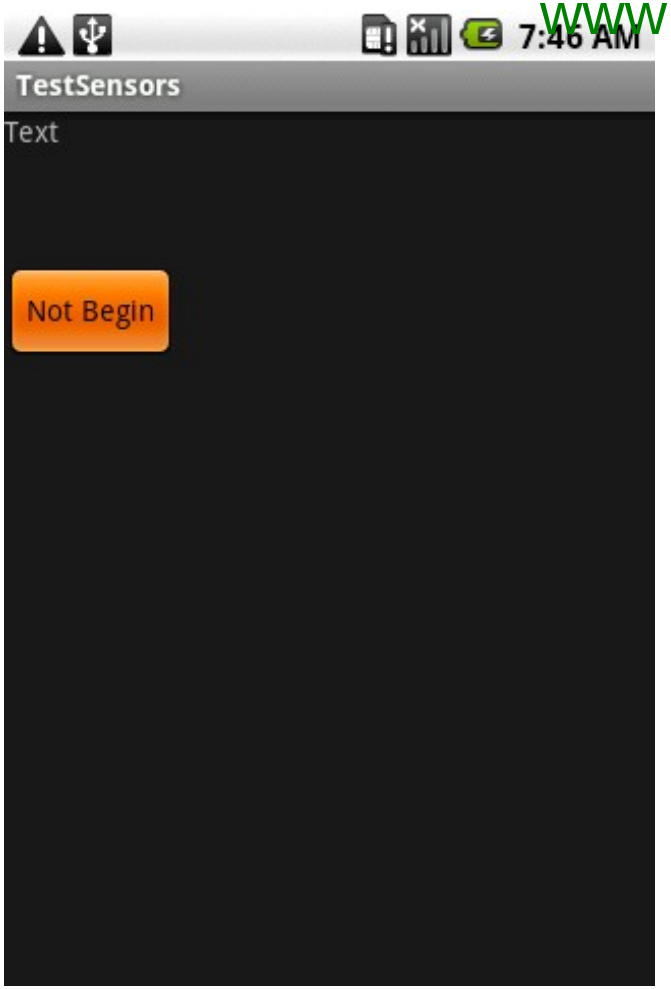
传感器的使用

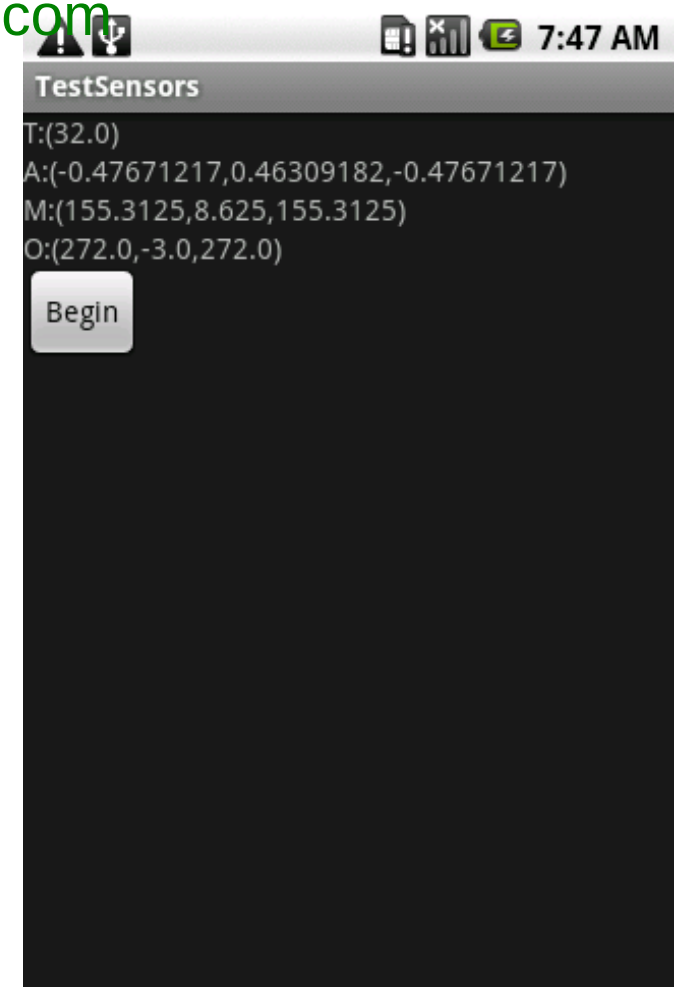
传感器是 Android 系统获取外部信息的重要手段， Android 系统支持 8 种传感器。其中加速度（ ACCELEROMETER ）、磁场（ MAGNETIC_FIELD ）、方向（ ORIENTATION ）是系统要求具有的。最为常用的是加速度传感器。 Android 的自动调整屏幕方向的功能就是由加速度传感器实现的，通过获得的 3 个方向加速度，比对重力加速度，计算得出了当前的方向。

传感器系统部分在的 android.hardware 包中，主要包括以下几个类和接口：

- SensorManager.java： 实现传感器系统核心的管理类 SensorManager
- Sensor.java： 单一传感器的描述性文件 Sensor
- SensorEvent.java： 表示传感器系统的事件类 SensorEvent；
- SensorEventListener.java： 传感器事件的监听者 SensorEventListener 接口


Java
Application
AndroidManifest .xml
Activity ::screenOrientation
Calling / Listener
ApplicationContext
OrientationEventListener
JAVA
Sensor Class
SensorManager
SensorListener
SensorEventListener
Sensor
SensorEvent
Java
Framework





这个程序可以获得当前传感器信息的信息，并且动态更新，点击左面的按钮开始获取传感器的信息，数据在屏幕上显示， T 为当前的温度、 A 为加速度传感器、 M 为磁场传感器、 O 为方向传感器，其中温度是一个数值，其他 3 个传感器都是包含了 3 个方向（ x,y,z ）的信息。

```
class SensorEventListenerImpl implements SensorEventListener {
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
           Log.v(TAG, "onAccuracyChanged");
    }
    public void onSensorChanged(SensorEvent event) {
           Log.v(TAG, "onSensorChanged");
           int type = event.sensor.getType();
           if(type == Sensor.TYPE_ACCELEROMETER) {
               float[] values = event.values;
               Log.v(TAG, "TYPE_ACCELEROMETER:("
                             +values[0]+","+values[1]+","+values[0]+")");
               mText1.setText("A:("+values[0]+","+values[1]+","+values[0]+")");
           }else if (type == Sensor.TYPE_MAGNETIC_FIELD){
               float[] values = event.values;
               Log.v(TAG, "TYPE_MAGNETIC_FIELD:("
                             +values[0]+","+values[1]+","+values[0]+")");
               mText2.setText("M:("+values[0]+","+values[1]+","+values[0]+")");
      }else if (type == Sensor.TYPE_ORIENTATION){
               float[] values = event.values;
               Log.v(TAG, "TYPE_ORIENTATION:("
                             +values[0]+","+values[1]+","+values[0]+")");
               mText3.setText("O:("+values[0]+","+values[1]+","+values[0]+")");
      }else if (type == Sensor.TYPE_TEMPERATURE){
                float[] values = event.values;
                Log.v(TAG, "TYPE_TEMPERATURE:("+values[0]+")");
                mText0.setText("T:("+values[0]+")");
      }
    }
}
```

```
@Override
    protected void onResume() {
        super.onResume();
        if (mSensorManager == null) {
            mSensorManager =  (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        }
       mSensorEventListener = new SensorEventListenerImpl();  //  传感器监听者
       mSensorA = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
       printSensorInfo(mSensorA,Sensor.TYPE_ACCELEROMETER);
       mSensorM = mSensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);
       printSensorInfo(mSensorM,Sensor.TYPE_MAGNETIC_FIELD);

    }
    private void beginSensors(){

       if (mSensorA != null) {     //  注册加速多传感器的监听者
           mSensorManager.registerListener(mSensorEventListener,
                                    mSensorA,SensorManager.SENSOR_DELAY_NORMAL);

       }
       if (mSensorM != null) {     //  注册磁场传感器的监听者
           mSensorManager.registerListener(mSensorEventListener,
                                    mSensorM,SensorManager.SENSOR_DELAY_NORMAL);

       }
       // … …
    }
```

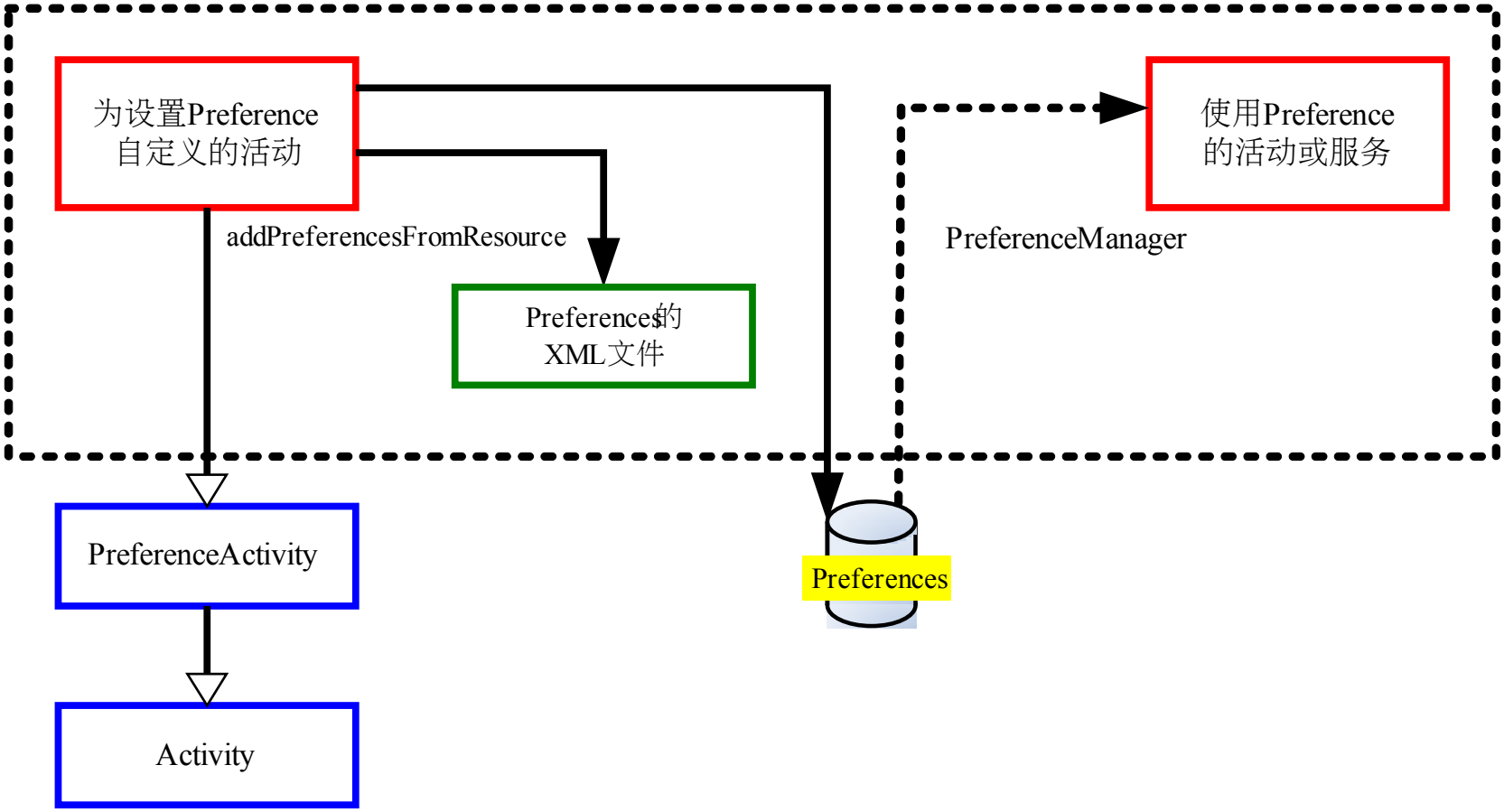
Preference 的使用

本部分的知识点包括以下内容:

- 使用 Preference 实现程序内的数据共享
- Preference 类与 UI 交互的关系

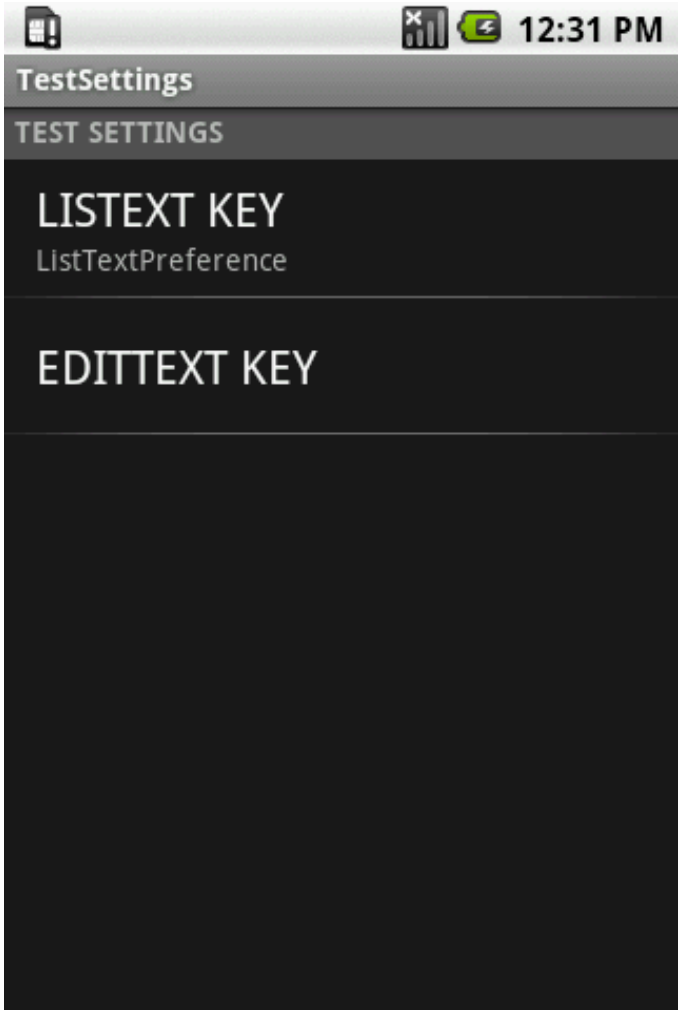
Preference 是 Android 中实现信息共享的一种方式。 Preference 可是很好地实现与 UI 和内部数据的结合。

Preference 的使用结构



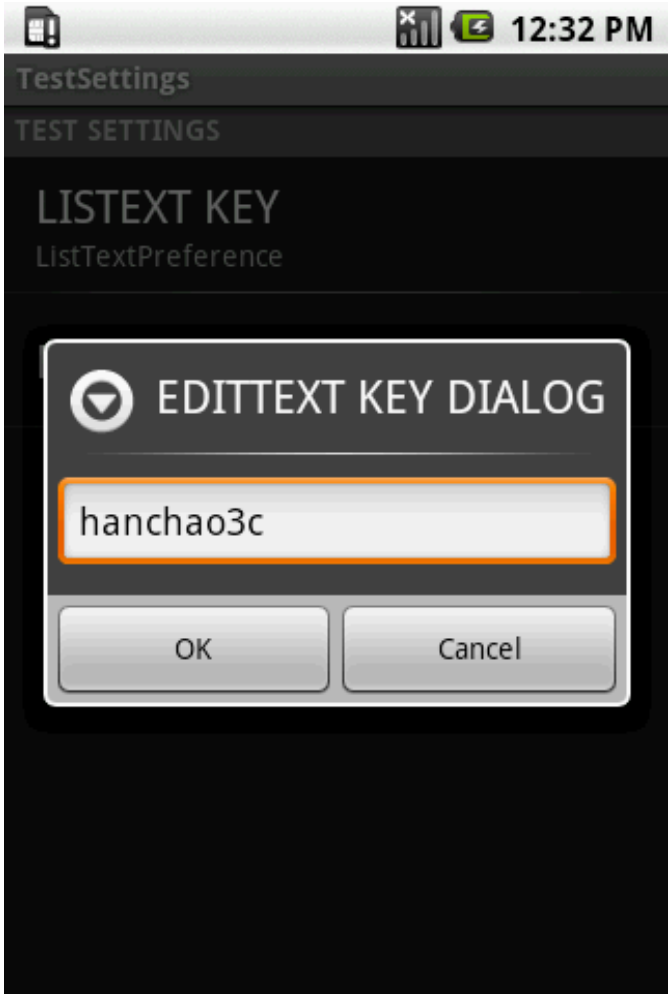
主屏幕和设置屏幕


12:31 PM
TestUseSettings
Listentryvalue =
Edittext =
SETTINGS

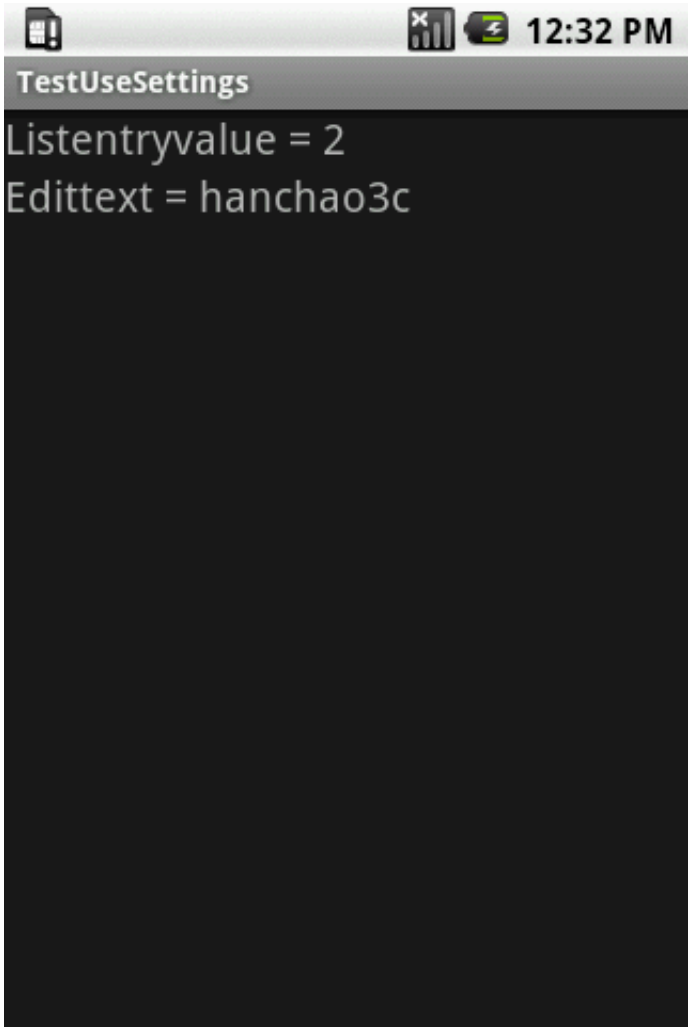

12:31 PM
TestSettings
TEST SETTINGS
LISTEXT KEY
ListTextPreference
EDITTEXT KEY

进行设置的过程


12:31 PM
TestSettings
TEST SETTINGS
LISTEXT KEY DIALOG
ENTRY-1
ENTRY-2
ENTRY-3
Cancel


12:32 PM
TestSettings
TEST SETTINGS
LISTEXT KEY
ListTextPreference
EDITTEXT KEY DIALOG
hanchao3c
OK
Cancel

设置完成后的主屏幕


12:32 PM
TestUseSettings
Listentryvalue = 2
Edittext = hanchao3c

本程序包含使用设置（TestUseSetting）和设置（TestSetting） 2 个屏幕， AndroidManifest.xml 的定义如下所示：

```
      <activity android:name="TestUseSettings"
android:label="TestUseSettings">
          <intent-filter>
              <action android:name="android.intent.action.MAIN"/>
              <category android:name="android.intent.category.LAUNCHER"/>
          </intent-filter>
      </activity>
      <activity android:name="TestSettings" android:label="TestSettings">
      </activity>
```

取出 Preferences 使用的过程：

```
@Override
  public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.testusesettings);
     mListEntryvalue = (TextView) findViewById(R.id.listentryvalue);
     mEditText       = (TextView) findViewById(R.id.edittext);
     mPreferences = PreferenceManager.getDefaultSharedPreferences(this);
  }
  @Override
  protected void onResume() {
     super.onResume();
     String edittext
           = mPreferences.getString(TestSettings.TEST_KEY_EDITTEXT,"");
     String listentryvalue
           = mPreferences.getString(TestSettings.TEST_KEY_LIST,"");
     Log.v(TAG, "Edittext = "+ edittext);
     Log.v(TAG, "Listentryvalue = "+ listentryvalue);
     mEditText.setText("Edittext = "+ edittext);
     mListEntryvalue.setText("Listentryvalue = "+ listentryvalue);
  }
```

TestSetting.java 的屏幕实现：

```
@Override
  public void onCreate(Bundle icicle)
  {
     super.onCreate(icicle);
     addPreferencesFromResource(R.xml.test_preferences);
     getPreferenceScreen().getSharedPreferences()
                  .registerOnSharedPreferenceChangeListener(this);
     mListPreference
                  = (ListPreference) findPreference(TEST_KEY_LIST);
     mListPreference.setSummary("ListTextPreference");
     mEditTextPreference
                  = (EditTextPreference) findPreference(TEST_KEY_EDITTEXT);
  }
  public void onSharedPreferenceChanged(SharedPreferences sharedPreferences,
        String key) {
          Log.v(TAG, "onSharedPreferenceChanged KEY = "+ key);
       if (key.equals(TEST_KEY_EDITTEXT)) {
          mEditTextPreference.setSummary(mEditTextPreference.getText());
       }
  }
```

res/xml/test_preferences.xml：

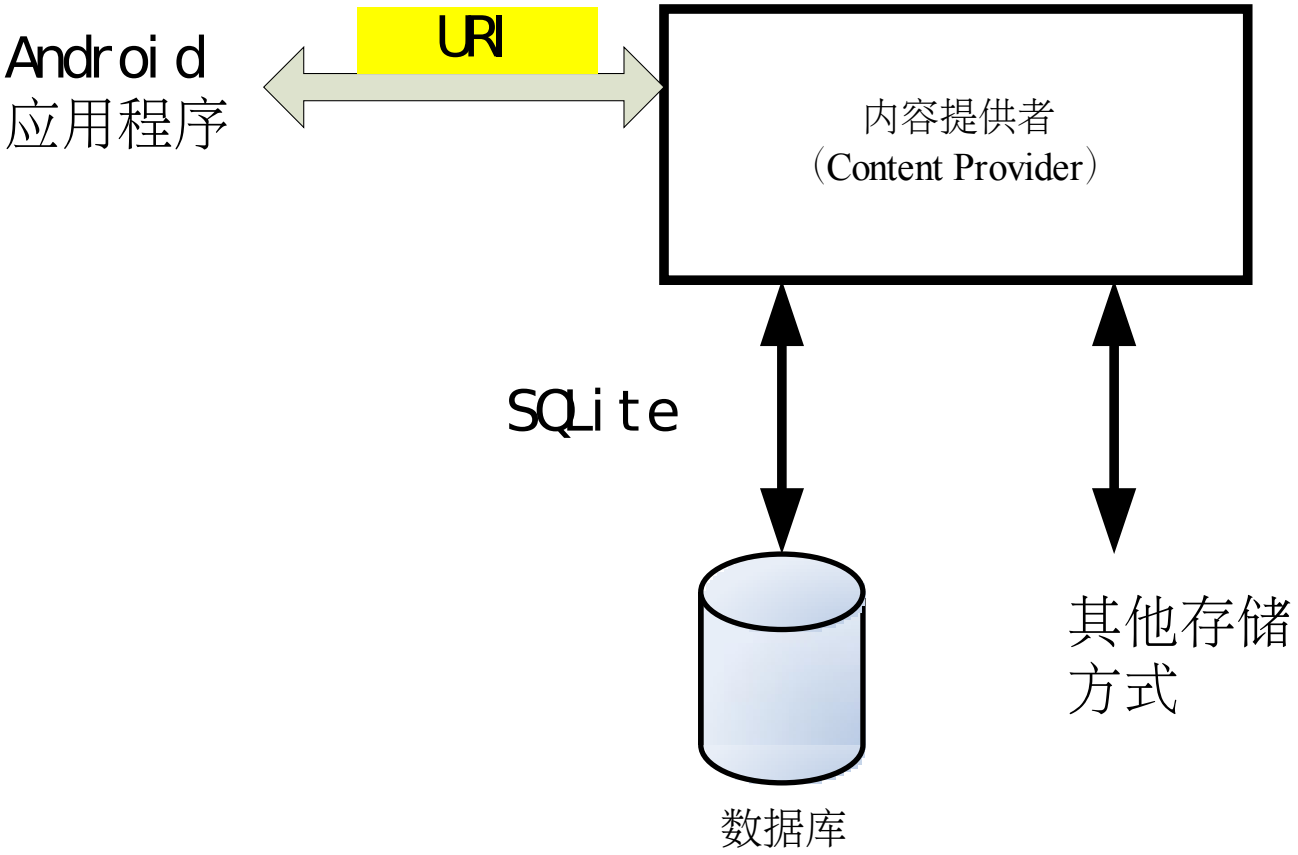
```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
       xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory
     android:title="@string/pref_test_settings">
     <ListPreference
          android:order="1"
          android:key="pref_list_key"
          android:defaultValue="@string/pref_list_key_default"
          android:title="@string/pref_list_key_title"
          android:dialogTitle="@string/pref_list_key_dialog_title"
          android:entries="@array/pref_list_key_entries"
          android:entryValues="@array/pref_list_key_entryvalues" />
       <EditTextPreference
          android:order="2"
          android:key="pref_edittext_key"
          android:title="@string/pref_edittext_key_title"
          android:summary=""
          android:dialogTitle="@string/pref_edittext_key_dialog_title" />
    </PreferenceCategory>
</PreferenceScreen>
```

values/res/array.xml：

```
<resources>
    <string-array name="pref_list_key_entries">
        <item>@string/pref_list_key_entry_1</item>
        <item>@string/pref_list_key_entry_2</item>
        <item>@string/pref_list_key_entry_3</item>
    </string-array>
    <string-array name="pref_list_key_entryvalues">
        <item>1</item>
        <item>2</item>
        <item>3</item>
    </string-array>
</resources>
```

数据存储和内容提供者

Provider 和 SQLite 数据库



AndroidManifest.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.datastorageapp">
    <application>
        <activity android:name="TestSqlite" android:label="TestSqlite">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <activity android:name="TestProvider" android:label="TestProvider">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <provider android:name="ExampleProvider"
           android:authorities="com.android.datastorageapp.exampleprovider" />
    </application>
</manifest>
```

TestSqlite


8:50 AM
TestSqlite
DataBase:/data/data/com.android.datastorageapp/databases/testsqlite.db
id name value
Insert
cupcake
15
Delete
Query


8:50 AM
TestSqlite
DataBase:/data/data/com.android.datastorageapp/databases/testsqlite.db
id name value
Insert
Delete
Query
1|cupcake|15


8:52 AM
TestSqlite
DataBase:/data/data/com.android.datastorageapp/databases/testsqlite.db
id name value
Insert
Delete
donut
Query
1|cupcake|15
2|donut|16
3|eclair|20


8:52 AM
TestSqlite
DataBase:/data/data/com.android.datastorageapp/databases/testsqlite.db
id name value
Insert
Delete
Query
1|cupcake|15
3|eclair|20

打开数据库:

```
# sqlite3 /data/data/com.android.datastorageapp/databases/testsqlite.db
SQLite version 3.5.9
Enter ".help" for instructions
```

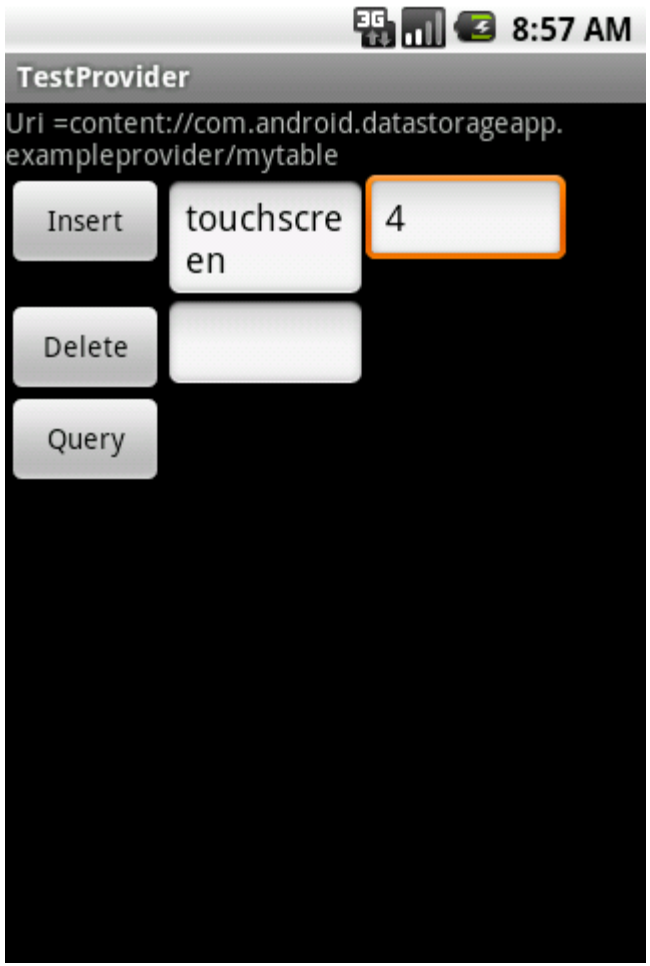
查看数据库中内容:

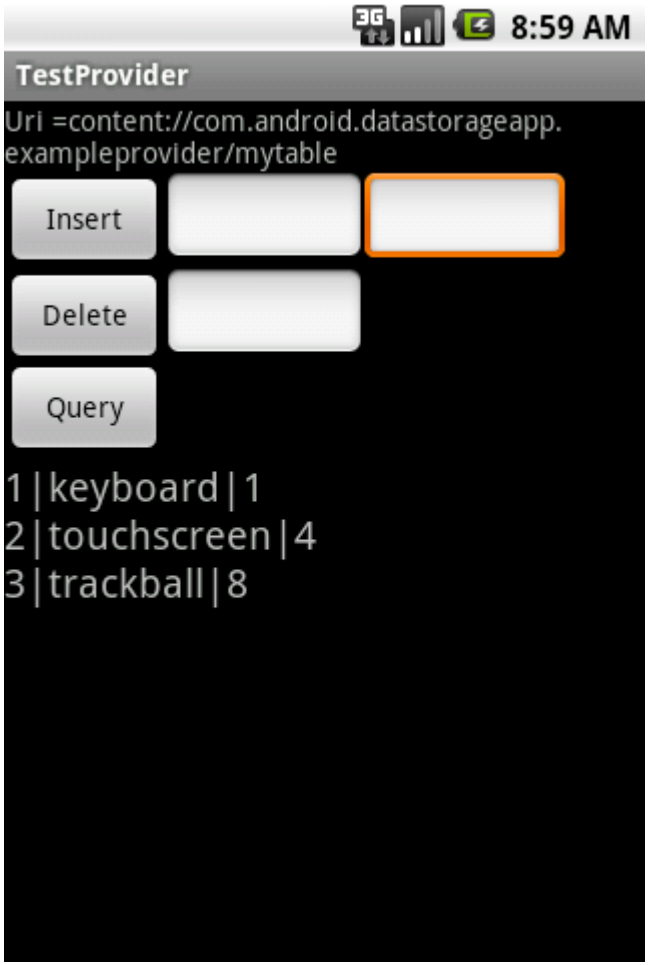
```
sqlite> .table
android_metadata  mytable

sqlite> .schema android_metadata
CREATE TABLE android_metadata (locale TEXT);
sqlite> .schema mytable
CREATE TABLE mytable (id INTEGER PRIMARY KEY,name TEXT,value INTEGER);

sqlite> select * from mytable;
1|cupcake|15
3|eclair|20
```

TestProvider


8:57 AM
TestProvider
Uri =content://com.android.datastorageapp.exampleprovider/mytable
Insert
touchscreen
4
Delete
Query


8:59 AM
TestProvider
Uri =content://com.android.datastorageapp.exampleprovider/mytable
Insert
Delete
Query
1|keyboard|1
2|touchscreen|4
3|trackball|8

点击事件的 Log：

```
V/TestProvider(  236): onClick()
V/TestProvider(  236): insert to database
V/ExampleProvider(  236): newUri =content://com.android.datastorageapp.exampleprovider/mytable/1
V/TestProvider(  236): Uri=content://com.android.datastorageapp.exampleprovider/mytable/1
```

打开和查看数据库中内容：

```
# sqlite3 /data/data/com.android.datastorageapp/databases/testprovider.db

sqlite> .schema mytable
CREATE TABLE mytable (id INTEGER PRIMARY KEY,name TEXT,value INTEGER);


sqlite> select * from mytable;
1|keyboard|1
2|touchscreen|4
3|trackball|8
```