ContentProvider 实例代码

当数据需要在应用程序间共享时，我们就可以利用 ContentProvider 为数据定义一个 URI。之后其他应用程序对数据进行查询或者修改时，只需要从当前上下文对象获得一个 ContentResolver(内容解析器)传入相应的 URI 就可以了。

contentProvider 和 Activity 一样是 Android 的组件，故使用前需要在 AndroidManifest.xml 中注册，必须放在主应用所在包或其子包下。

一、AndroidManifest.xml

```
<application android:icon="@drawable/icon"
android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <intent-filter>
                <data
android:mimeType="vnd.android.cursor.dir/person" />
            </intent-filter>
            <intent-filter>
                <data
android:mimeType="vnd.android.cursor.item/person" />
            </intent-filter>
        </activity>
        <!-- 配置内容提供者,android:authorities 为该内容提供者取名作为在本应用中的唯一标识 -->
```

```
        <provider android:name=".providers.PersonProvider"
android:authorities="cn.xyCompany.providers.personProvider"/>
    </application>
```

二、内容提供者

```
package cn.xy.cotentProvider.app.providers;
import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.util.Log;
import cn.xy.cotentProvider.service.DBOpeningHelper;

/**
 * contentProvider 作为一种组件必须放在应用所在包或其子包下，主要作用是对外共享数据
 * 测试步骤 1：将本项目先部署
 * 测试步骤 2：调用测试方法
 * @author xy
 *
 */
public class PersonProvider extends ContentProvider
{
    private DBOpeningHelper dbHelper;

    // 若不匹配采用 UriMatcher.NO_MATCH(-1)返回
```

```
    private static final UriMatcher MATCHER = new
UriMatcher(UriMatcher.NO_MATCH);

    // 匹配码
    private static final int CODE_NOPARAM = 1;
    private static final int CODE_PARAM = 2;

    static
    {
        // 对等待匹配的 URI 进行匹配操作，必须符合
cn.xyCompany.providers.personProvider/person 格式
        // 匹配返回 CODE_NOPARAM，不匹配返回-1
        MATCHER.addURI("cn.xyCompany.providers.personProvider",
"person", CODE_NOPARAM);

        // #表示数字 cn.xyCompany.providers.personProvider/person/10
        // 匹配返回 CODE_PARAM，不匹配返回-1
        MATCHER.addURI("cn.xyCompany.providers.personProvider",
"person/#", CODE_PARAM);
    }

    @Override
    public boolean onCreate()
    {
        dbHelper = new DBOpeningHelper(this.getContext());
        return true;
    }

    /**
```

```
     * 外部应用向本应用插入数据
     */
    @Override
    public Uri insert(Uri uri, ContentValues values)
    {
        SQLiteDatabase db = dbHelper.getWritableDatabase();
        switch (MATCHER.match(uri))
        {
            case CODE_NOPARAM:
                // 若主键值是自增长的 id 值则返回值为主键值，否则为行
号，但行号并不是 RecNo 列
                long id = db.insert("person", "name", values);
                Uri insertUri = ContentUris.withAppendedId(uri, id);
                return insertUri;
            default:
                throw new IllegalArgumentException("this is unkown
uri:" + uri);
        }
    }

    /**
     * 外部应用向本应用删除数据
     */
    @Override
    public int delete(Uri uri, String selection, String[]
selectionArgs)
    {
        SQLiteDatabase db = dbHelper.getWritableDatabase();
        switch (MATCHER.match(uri))
```

```
        {
            case CODE_NOPARAM:
                return db.delete("person", selection, selectionArgs);
// 删除所有记录
            case CODE_PARAM:
                long id = ContentUris.parseId(uri); // 取得跟在 URI 后
面的数字
                Log.i("provider", String.valueOf(id));
                String where = "id = " + id;
                if (null != selection
&& !"".equals(selection.trim()))
                {
                    where += " and " + selection;
                }
                return db.delete("person", where, selectionArgs);
            default:
                throw new IllegalArgumentException("this is unkown
uri:" + uri);
        }
    }

    /**
     * 外部应用向本应用更新数据
     */
    @Override
    public int update(Uri uri, ContentValues values, String
selection, String[] selectionArgs)
    {
        SQLiteDatabase db = dbHelper.getWritableDatabase();
```

```
        switch (MATCHER.match(uri))
        {
            case CODE_NOPARAM:
                return db.update("person",values,selection,
selectionArgs); // 更新所有记录
            case CODE_PARAM:
                long id = ContentUris.parseId(uri); // 取得跟在 URI 后
面的数字
                String where = "id = " + id;
                if (null != selection
&& !"".equals(selection.trim()))
                {
                    where += " and " + selection;
                }
                return
db.update("person",values,where,selectionArgs);
            default:
                throw new IllegalArgumentException("this is unkown
uri:" + uri);
        }
    }

    /**
     * 返回对应的内容类型
     * 如果返回集合的内容类型，必须以 vnd.android.cursor.dir 开头
     * 如果是单个元素，必须以 vnd.android.cursor.item 开头
     */
    @Override
    public String getType(Uri uri)
```

```
    {
        switch(MATCHER.match(uri))
        {
            case CODE_NOPARAM:
                return "vnd.android.cursor.dir/person";
            case CODE_PARAM:
                return "vnd.android.cursor.item/person";
            default:
                throw new IllegalArgumentException("this is unkown
uri:" + uri);
        }
    }

    @Override
    public Cursor query(Uri uri, String[] projection, String
selection, String[] selectionArgs, String sortOrder)
    {
        SQLiteDatabase db = dbHelper.getReadableDatabase();
        switch (MATCHER.match(uri))
        {
            case CODE_NOPARAM:
                return db.query("person", projection, selection,
selectionArgs, null, null, sortOrder);
            case CODE_PARAM:
                long id = ContentUris.parseId(uri); // 取得跟在URI后
面的数字
                String where = "id = " + id;
                if (null != selection
&& !"".equals(selection.trim()))
```

```
                {
                    where += " and " + selection;
                }
                return db.query("person", projection, where,
selectionArgs, null, null, sortOrder);
            default:
                throw new IllegalArgumentException("this is unkown
uri:" + uri);
        }
    }

}
```

三、测试代码

```
package cn.xy.test.test;
import android.content.ContentResolver;
import android.content.ContentValues;
import android.database.Cursor;
import android.net.Uri;
import android.test.AndroidTestCase;
import android.util.Log;

/**
 * 测试代码
 * @author xy
 *
 */
public class TestProviders extends AndroidTestCase
{
```

```
    // 在执行该测试方法时需要先将还有内容提供者的项目部署到 Android
中，否则无法找到内容提供者
    public void testInsert()
    {
        Uri uri =
Uri.parse("content://cn.xyCompany.providers.personProvider/person");
        ContentResolver resolver =
this.getContext().getContentResolver();
        ContentValues values = new ContentValues();
        values.put("name", "xy");
        values.put("phone", "111");
        resolver.insert(uri, values); // 内部调用内容提供者的 insert
方法
    }

    // 不带 id 参数的删除
    public void testDelete1()
    {
        Uri uri =
Uri.parse("content://cn.xyCompany.providers.personProvider/person");
        ContentResolver resolver =
this.getContext().getContentResolver();
        int rowAffect = resolver.delete(uri, null, null);
        Log.i("rowAffect", String.valueOf(rowAffect));
    }

    // 带参数的删除，通过 URI 传递了 id 至 contentProvider 并可追加其他
条件
    public void testDelete2()
```

```
    {
        Uri uri =
Uri.parse("content://cn.xyCompany.providers.personProvider/person/18"
);
        ContentResolver resolver =
this.getContext().getContentResolver();
        int rowAffect = resolver.delete(uri, "name = ?", new String[]
{ "XY2" }); // 在 provider 中手动进行了拼装
        Log.i("rowAffect", String.valueOf(rowAffect));
    }

    public void testUpdate()
    {
        Uri uri =
Uri.parse("content://cn.xyCompany.providers.personProvider/person/19"
);
        ContentResolver resolver =
this.getContext().getContentResolver();
        ContentValues values = new ContentValues();
        values.put("name", "newxy");
        values.put("phone", "new111");
        int rowAffect = resolver.update(uri, values, null, null);
        Log.i("rowAffect", String.valueOf(rowAffect));
    }

    public void testQuery()
    {
```

```
        Uri uri =
Uri.parse("content://cn.xyCompany.providers.personProvider/person/19"
);
        ContentResolver resolver =
this.getContext().getContentResolver();
        Cursor cursor = resolver.query(uri, new
String[]{"id","name","phone"}, null, null, "id asc");
        if(cursor.moveToFirst())
        {
            Log.i("query",
cursor.getString(cursor.getColumnIndex("name")));
        }
        cursor.close();
    }
}
```