



# Android应用开发

主讲教师 胡鑫

[hithuxin@hit.edu.cn](mailto:hithuxin@hit.edu.cn)

## 第10章 ContentProvider

1. [ContentProvider](#)
2. [ContentResolver](#)
3. [MediaStore媒体库示例](#)
4. [自定义ContentProvider](#)

## 10.1 ContentProvider

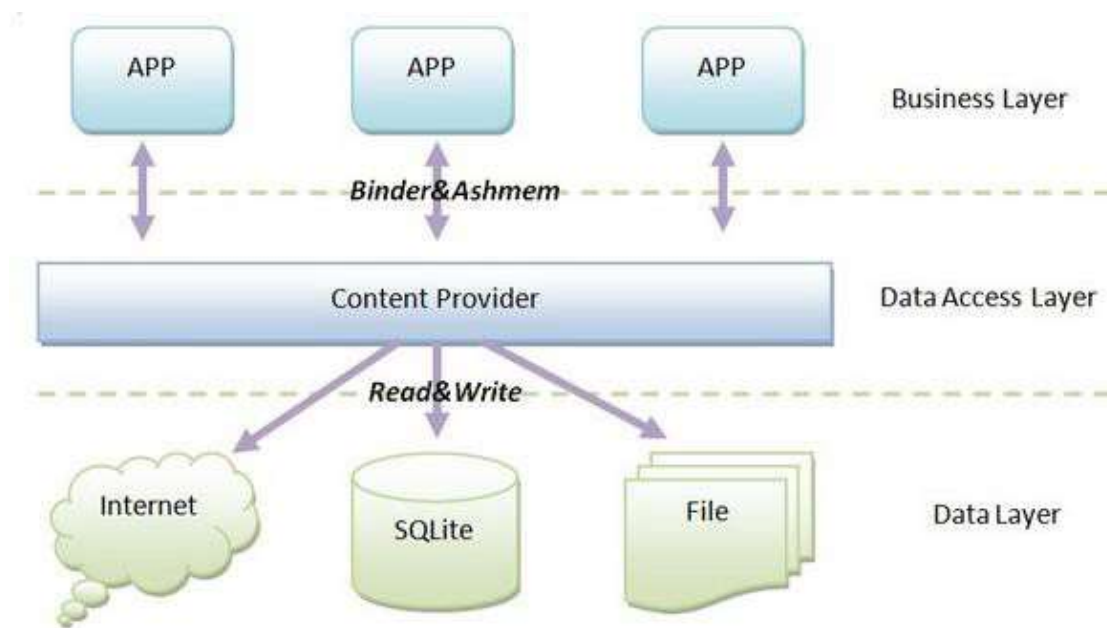
### ■ Why ContentProvider ?

- Android应用程序运行在不同的进程空间中，因此不同应用程序的数据是不能够直接访问的。
- ContentProvider提供了[应用程序之间共享数据](#)的方法。
- 一个应用程序可以通过ContentProvider将自己的数据暴露出去。

Android四大核心组件： Activity、BroadcastReceiver、Service、**ContentProvider**

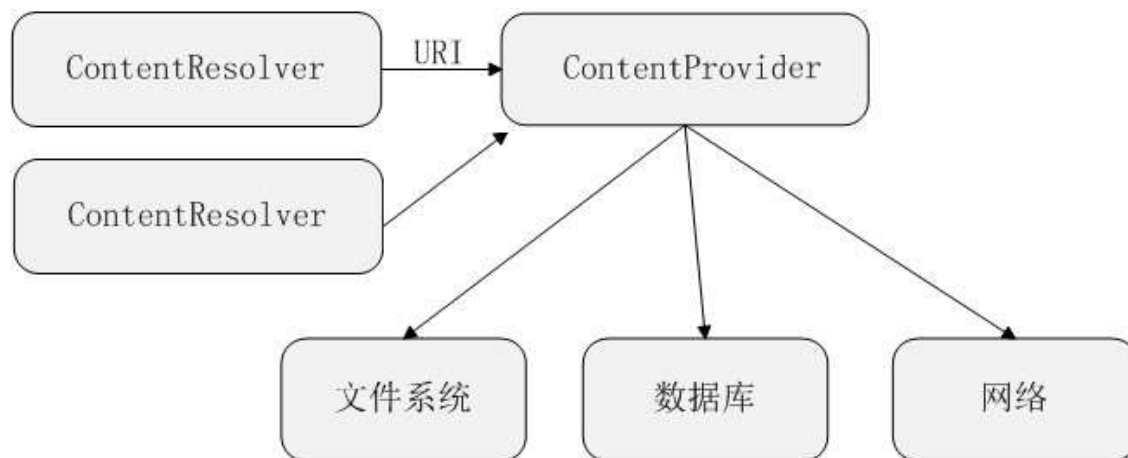
## ContentProvider

- 应用程序通过ContentProvider访问数据而不需要关心数据具体的存储及访问过程，这样既提高了数据的访问效率，同时也保护了数据。



## 10.2 ContentResolver


- 应用程序使用 **ContentResolver** 对象，利用**URI**，才能访问ContentProvider提供的**数据集**。



一个ContentProvider可以提供多个数据集，可为多个ContentResolver服务

## 关键1：什么是ContentProvider数据集

- ContentProvider可以形象地看作是"数据库"。
- ContentProvider数据集类似于数据库的"表"。
- ContentProvider数据集的每条记录都包含一个long型的字段\_ID，用来唯一标识每条记录，例如：



long型_ID	NAME	AGE	HEIGHT
1	Tom	21	1.81
2	Jim	22	1.78

备注：一个ContentProvider表并不要求必须有一个\_ID列,不过如果要把查询的数据放在ListView中显示（使用SimpleCursorAdapter填充）,则必须有\_ID列

## 关键2：什么是URI

■ URI：统一资源标识符（Uniform Resource Identifier），用来标识资源的逻辑位置(定义远程或本地的可用资源)

■ 每一个Content Provider 都对外提供一个能够唯一标识自己数据集的公开URI；如果一个Content Provider管理多个数据集，则应给每个数据集分配一个独立的URI。

■ URI字符串基本格式：

可用Uri.parse(串)转换为Uri

■ content://<authority>/<data\_path>/<id>

固定前缀

ContentProvider名称(唯一标识)  
(近似理解为数据库名)

数据集名称  
(近似理解为表)

数据编号，用来匹配数据集中\_ID字段的值  
(用来唯一确定数据集中的一条记录) 如果请求的数据并不只限于一条数据，则<id>可以省略

## URI示例

ContentProvider名称      数据集名称

- `content://contacts/people/` 表示全部联系人信息的URI
- `content://contacts/people/1` 表示ID=1的联系人信息的URI

ContentProvider名称      数据集名称

推荐写法

- `content://com.android.contacts/contacts` 原生写法
- `ContactsContract.Contacts.CONTENT_URI` Uri常量写法

由于URI通常比较长，而且容易写错，所以，在Android系统中定义了一些辅助类和常量来代替这些长字符串。



## 系统的一些ContentProvider

都位于android.provider包下

- Browser: 存储如浏览器的信息。
- CallLog: 存储通话记录等信息。
- Contacts: 存储联系人等信息。
- MediaStore: 存储媒体文件的信息。
- Settings: 存储设备的设置和首选项信息。
- ...

## 关键3： ContentResolver基本用法

(1) 如何创建ContentResolver：

```
ContentResolver resolver = getContentResolver();
```

(2) 如何增删改查：(查询是重点)

```
resolver.query()、insert()、update()、delete()
```

## ContentResolver四大操作说明

- `query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)`: 通过Uri进行查询, 返回一个Cursor。
- `insert(Uri url, ContentValues values)`: 将一组数据插入到Uri 指定的地方, 返回最新添加那个记录的Uri。
- `update(Uri uri, ContentValues values, String where, String[] selectionArgs)`: 更新Uri指定位置的数据, 返回更新的行数。
- `delete(Uri url, String where, String[] selectionArgs)`: 删除指定Uri并且符合一定条件的数据, 返回删除的行数。

## 参数含义 -- 以查询为例

- 查询方法：通过Uri进行查询，返回一个Cursor。

resolver.query( Uri uri,

uri: 查询的数据集

String[] projection,

projection: 返回数据集的字段  
(null表示所有字段)

String selection,

selection: 查询条件(null表示无条件)

String[] selectionArgs,

selectionArgs: 查询条件的值  
(null表示无查询条件值)

String sortOrder );

sortOrder: 排序方式  
(null表示按默认排序)

例如：

```
ContentResolver resolver = getContentResolver();
```

```
Cursor cursor = resolver.query(
```

```
    ContactsContract.Contacts.CONTENT_URI,
```

```
    new String[]{"_id", "display_name", "has_phone_number"},
```

```
    "display_name like ?",
```

?为占位符

```
    new String[]{"%zz%"},
```

```
    "display_name asc");
```

ASC: 升序(默认)

DESC: 降序

Contacts数据集  
的字段

基本含义：  
`select _id, display_name, has_phone_number from Contacts数据集 where display_name like %zz% orderby display_name asc`

## 示例：显示手机联系人及其电话

添加读取联系人的权限：

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

```
String str="联系人列表:\n";
TextView tv=(TextView)findViewById(R.id.textView1);
//查询所有联系人(联系人数据集)
ContentResolver resolver = getContentResolver();
Cursor cursor = resolver.query(null, null, null, null);
while(cursor.moveToNext()){
    //取得联系人的名字索引
    int nameIndex = cursor.getColumnIndex("display_name");
    String name = cursor.getString(nameIndex);
    str += (name+":\n");
    //取得联系人的ID索引值
    String contactId = cursor.getString(cursor.getColumnIndex("_id"));
```

//查询该联系人的所有电话号码(电话数据集)

```
Cursor phone = resolver.query(
```

```
    ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
```

```
    null, "contact_id = ?", new String[]{ contactId }, null);
```

//一个人可能有几个号码

```
while( phone.moveToNext() ){
```

```
    String strPhoneNumber = phone.getString(phone.getColumnIndex("data1"));
```

```
    str += (strPhoneNumber+"\n");
```

```
}
```

```
str += "\n";
```

```
phone.close();
```

```
}
```

```
cursor.close();
```

```
tv.setText(str);
```

"data1"字段存放  
的是电话号码

## 补充：Android6.0以后添加权限请求代码

```
FATAL EXCEPTION: main
Process: com.example.testsqlite, PID: 18239
java.lang.SecurityException: Permission Denial: opening provider com.android.providers.contacts.ContactsProvider2 from ProcessR
    at android.os.Parcel.readException(Parcel.java:1620)
    at android.os.Parcel.readException(Parcel.java:1573)
```

```
if (ActivityCompat.checkSelfPermission(
    MainActivity.this, Manifest.permission.READ_CONTACTS) !=
    PackageManager.PERMISSION_GRANTED ) {

    ActivityCompat.requestPermissions(MainActivity.this,
        new String[] { Manifest.permission.READ_CONTACTS }, 123);

    return;
}
```

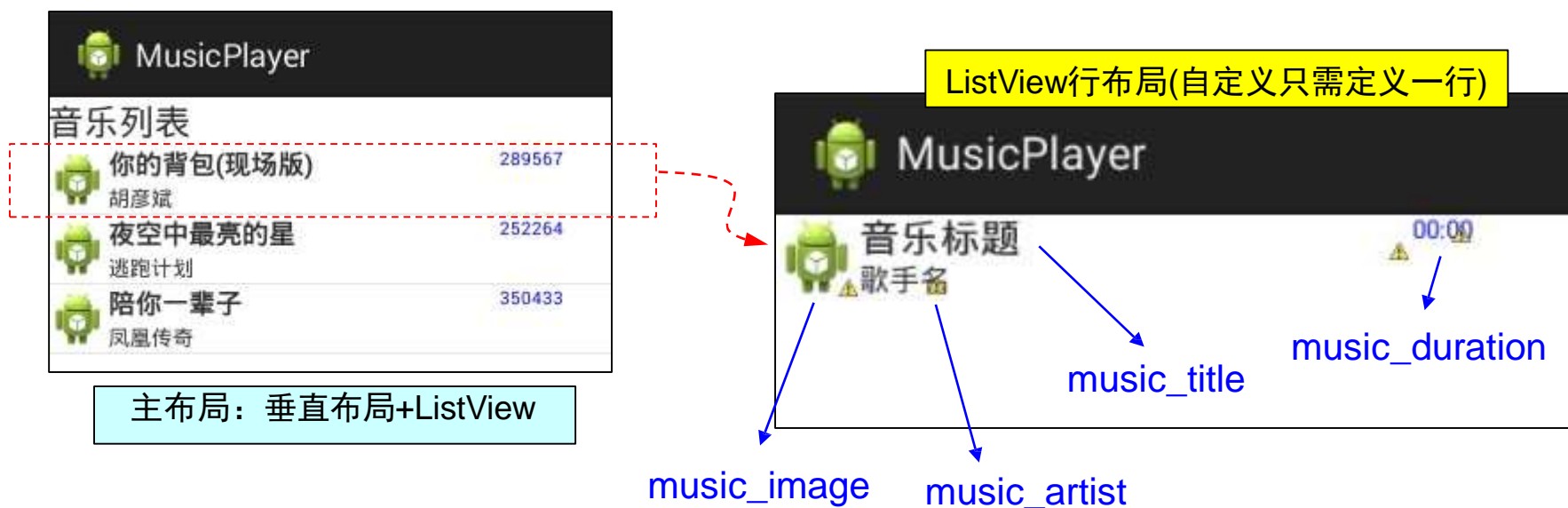


## 运行结果：



## 10.3 MediaStore媒体库示例

URI: `MediaStore.Audio.Media.EXTERNAL_CONTENT_URI`



底下还两个隐藏2个TextView的: `music_id`、`music_url`

说明:

- (1) `ContentResolver`含有`_id`字段, 所有视图中也需绑定一个, 但可以不显示
- (2) `music_url`可不显示, 后续要获取该值来播放音乐

添加读取外部存储器权限

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

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <ImageView
            android:id="@+id/music_image"
            android:layout_width="34dp"
            android:layout_height="match_parent"
            android:src="@drawable/ic_launcher" />

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="vertical" >
            <TextView
                android:id="@+id/music_title"
                android:layout_width="220dp"
                android:layout_height="wrap_content"
                android:text="音乐标题"
                android:textSize="15dp"
                android:textStyle="bold" />
            <TextView
                android:id="@+id/music_artist"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textSize="12dp"
                android:text="歌手名" />
            </LinearLayout>
        </LinearLayout>
```

```
<TextView
    android:id="@+id/music_duration"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="10dp"
    android:text="00:00" />
</LinearLayout>

<TextView
    android:id="@+id/music_id"
    android:layout_width="wrap_content"
    android:layout_height="0dp"
    android:visibility="invisible"
    android:text="TextView" />

<TextView
    android:id="@+id/music_url"
    android:layout_width="wrap_content"
    android:layout_height="0dp"
    android:visibility="invisible"
    android:text="TextView" />
</LinearLayout>
```


## 使用ContentResolver+SimpleCursorAdapter

```
Cursor cursor = getContentResolver().query(  
    MediaStore.Audio.Media.EXTERNAL_CONTENT_URI, null, null, null,  
    MediaStore.Audio.Media.TITLE);    //按标题排序
```

```
SimpleCursorAdapter adapter = new SimpleCursorAdapter(  
    this, R.layout.list_row, cursor, from, to);
```



使用自定义布局



from: 查询结果字段 to: 视图控件id 数据将自动绑定到对应显示控件上

from、to的定义见后页

## from 和 to 的定义

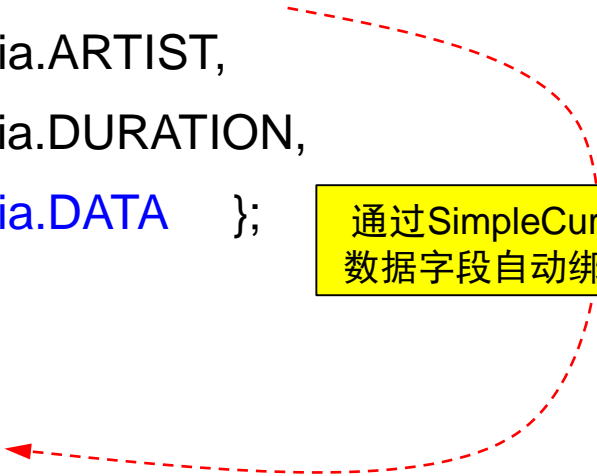
```
String[] from = { MediaStore.Audio.Media._ID,  
MediaStore.Audio.Media.TITLE,  
MediaStore.Audio.Media.ARTIST,  
MediaStore.Audio.Media.DURATION,  
MediaStore.Audio.Media.DATA };
```

from: 查询结果字段

```
int[] to = { R.id.music_id,  
R.id.music_title,  
R.id.music_artist,  
R.id.music_duration,  
R.id.music_url };
```

to: 视图控件id

通过SimpleCursorAdapter可将  
数据字段自动绑定到显示控件上



## ListView代码:

```
ListView li=(ListView)findViewById(R.id.listView1);
li.setAdapter(adapter);

li.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view,int position, long id) {
        TextView music_url=(TextView)view.findViewById(R.id.music_url);
        try{
            myPlayer.reset();
            myPlayer.setDataSource(music_url.getText().toString());
            myPlayer.prepare();
            myPlayer.start();
        }catch (Exception e){ }
    }
});
```

关键点  
(view参数获得自定义布局容器)

成员定义: MediaPlayer myPlayer=new MediaPlayer();

```
@Override
```

```
protected void onDestroy() {
```

```
    super.onDestroy();
```

```
    if(myPlayer!=null ){
```

```
        myPlayer.stop();
```

```
        myPlayer.release();
```

```
    }
```

```
}
```

Activity退出后结束播放

## 10.4 自定义ContentProvider

- [ContentProvider提供端](#) （ContentProviderDemo项目）
- [ContentProvider使用端](#) （TestContentProvider项目）



TestContentProvider			
查询	添加	删除所有	修改
1	wust1		21
2	wust2		22
3	wust3		23
4	wust4		24
5	wust5		25



## 1. ContentProvider提供端

### ■ 创建ContentProvider主要步骤：

(1) 数据库准备：使用SQLiteOpenHelper创建数据库和表

(2) ContentProvider类：

① 新建ContentProvider (authority值：Provider的唯一标识)

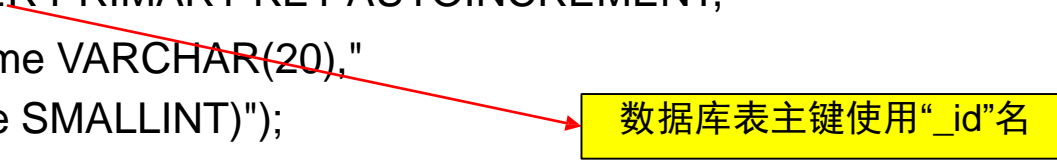
② 定义UriMacther (用于匹配当前操作是哪个数据集URI)

③ 实现ContentProvider的6个方法：

onCreate、getType、query、insert、delete、update

## (1) 数据库准备 -- SQLiteOpenHelper类

```
public class DBHelper extends SQLiteOpenHelper {  
    public DBHelper(Context context, String name, CursorFactory factory, int version) {  
        super(context, name, factory, version);  
    }  
    @Override  
    public void onCreate(SQLiteDatabase db) {  
        db.execSQL("CREATE TABLE IF NOT EXISTS person  
        (" + "_id INTEGER PRIMARY KEY AUTOINCREMENT,"  
            + "name VARCHAR(20),"  
            + "age SMALLINT)");  
    }  
    @Override  
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
        db.execSQL("DROP TABLE IF EXISTS person");  
        onCreate(db);  
    }  
}
```



数据库表主键使用“\_id”名

## (1) 数据库准备 (续) -- 数据模型类Person

```
public class Person {  
    private int _id;  
    private String name;  
    private int age;  
  
    public Person() { }  
    public Person(String name, int age) {  
        this.name = name; this.age = age;  
    }  
    public int getId() { return _id; }  
    public void setId(int _id) { this._id = _id; }  
    public String getName() { return name; }  
    public void setName(String name) { this.name = name; }  
    public int getAge() { return age; }  
    public void setAge(int age) { this.age = age; }  
}
```

## (1) 数据库准备 (续) -- 添加几条测试数据



## 布局文件

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin" android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin" tools:context="com.example.contentproviderdemo.MainActivity" >

    <Button android:id="@+id/button1"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_alignParentLeft="true" android:layout_alignParentTop="true"
        android:text="person表添加5条记录" />

    <Button android:id="@+id/button2"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/button1" android:layout_alignRight="@+id/button1"
        android:layout_below="@+id/button1" android:text="清空person表" />

    <Button android:id="@+id/button3"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/button2" android:layout_alignRight="@+id/button2"
        android:layout_below="@+id/button2" android:text="查询person表" />

    <TextView android:id="@+id/textView1"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/button3" android:layout_below="@+id/button3"
        android:layout_marginLeft="16dp" android:layout_marginTop="17dp"
        android:text="@string/hello_world" />
</RelativeLayout>
```

//先创建数据库+表，并添加5条记录

```
Button btn1=(Button)findViewById(R.id.button1);
```

```
btn1.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```
    public void onClick(View v) {
```

```
        DBHelper helper=new DBHelper(getApplicationContext(), "test.db", null, 1);
```

```
        SQLiteDatabase db=helper.getWritableDatabase();
```

```
        for(int i=1;i<6;i++){
```

```
            Person person = new Person();
```

```
            person.setName("wust"+i);
```

```
            person.setAge(20+i);
```

```
            db.execSQL("INSERT INTO person VALUES (NULL, ?, ?)",
```

```
                        new Object[ ] { person.getName(), person.getAge() } );
```

```
        }
```

```
        db.close();
```

```
        Toast.makeText(getApplicationContext(), "记录添加成功", Toast.LENGTH_SHORT)
```

```
            .show();
```

```
    }
```

```
});
```

清空person表

```
//清空person表
Button btn2=(Button)findViewById(R.id.button2);
btn2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        DBHelper helper=new DBHelper(getApplicationContext(), "test.db", null, 1);
        SQLiteDatabase db=helper.getWritableDatabase();
        db.execSQL("delete from person");
        db.close();
        Toast.makeText(getApplicationContext(), "记录全部删除", Toast.LENGTH_SHORT)
            .show();
    }
});
```

//查询person表记录

```
Button btn3=(Button)findViewById(R.id.button3);
```

```
btn3.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```
    public void onClick(View v) {
```

```
        DBHelper helper=new DBHelper(getApplicationContext(), "test.db", null, 1);
```

```
        SQLiteDatabase db=helper.getWritableDatabase();
```

```
        Cursor cursor = db.rawQuery( "SELECT * FROM person", null );
```

```
        TextView tv=(TextView)findViewById(R.id.textView1);    //简单用TextView显示数据
```

```
        tv.setText("查询到"+cursor.getCount()+"条记录");
```

```
        while (cursor.moveToNext()) {
```

```
            int id = cursor.getInt(cursor.getColumnIndex("_id") );
```

```
            String name = cursor.getString(cursor.getColumnIndex("name") );
```

```
            int age = cursor.getInt(cursor.getColumnIndex("age") );
```

```
            tv.setText(tv.getText()+"\n"+"_id="+id+",name="+name+",age="+age);
```

```
        }
```

```
        cursor.close();
```

```
        db.close();
```

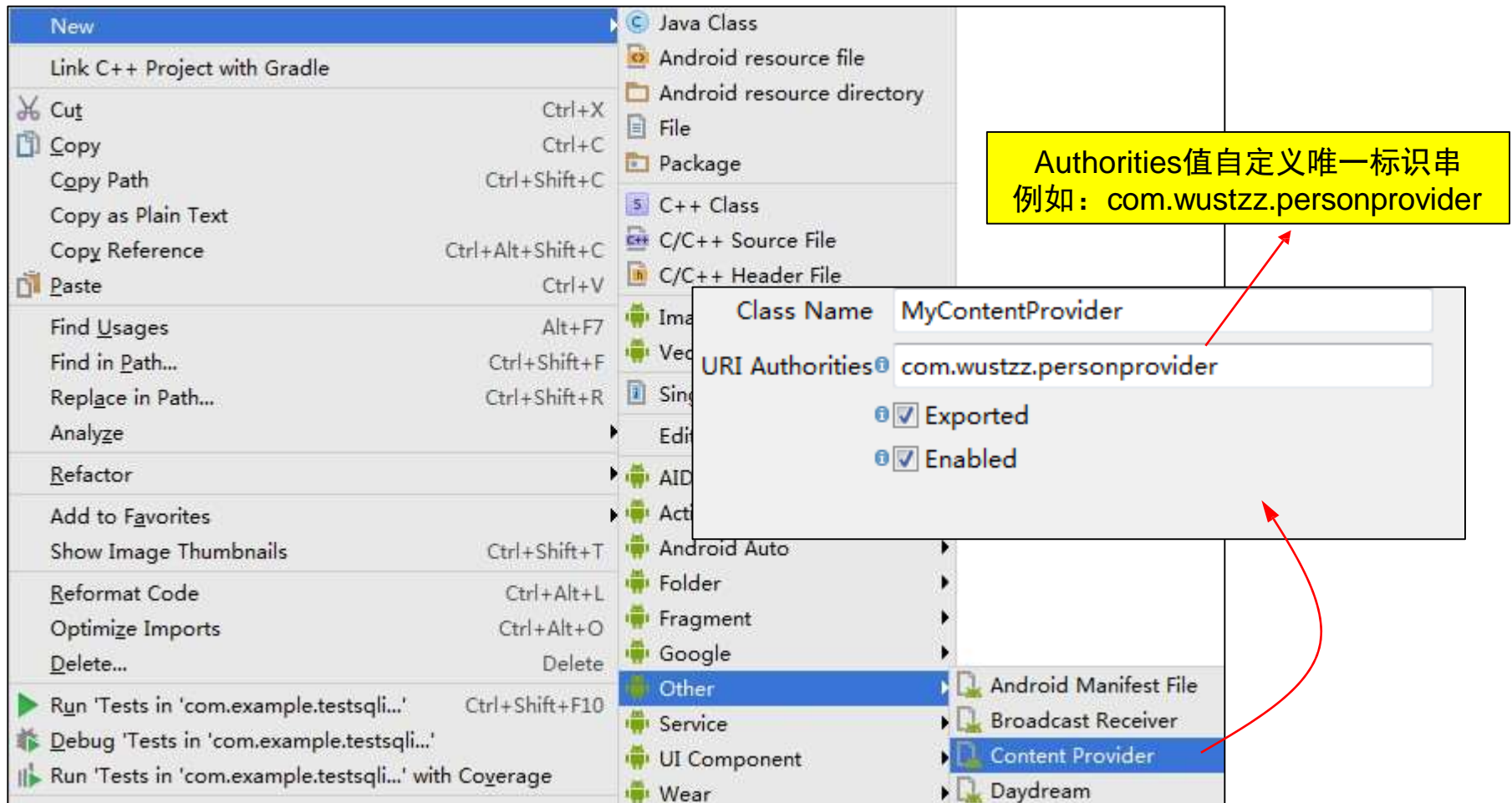
```
    }
```

```
});
```

[【返回】](#)



## ① 新建ContentProvider类（如MyContentProvider）



## ① 新建ContentProvider类（续）

### ■ 自动在AndroidManifest.xml注册：

```
<provider  
    android:name=".MyContentProvider"  
    android:authorities="com.wustzz.personprovider"  
    android:enabled="true"  
    android:exported="true" >  
  
</provider>
```

关键数据：URI的authority

必须是唯一标识串

## MyContentProvider代码框架

```
public class MyContentProvider extends ContentProvider {  
    public MyContentProvider() {  
        //要实现以下6个方法  
        public boolean onCreate() {  
        public String getType(Uri uri) {  
        public Cursor query(Uri uri, String[] projection, String selection,  
                             String[] selectionArgs, String sortOrder) {  
        public Uri insert(Uri uri, ContentValues values) {  
        public int delete(Uri uri, String selection, String[] selectionArgs) {  
        public int update(Uri uri, ContentValues values, String selection,  
                           String[] selectionArgs) {  
    }  
}
```

要实现的6个方法

【[返回](#)】

## ② 定义UriMacther

### ■ 在MyContentProvider中定义UriMacther:

```
private DBHelper helper; //后面代码会用到
private static final UriMatcher MATCHER = new UriMatcher(UriMatcher.NO_MATCH);
private static final int PERSONS = 1; //复数
private static final int PERSON = 2; //单数
static {
    MATCHER.addURI("com.wustzz.personprovider", "person", PERSONS);
    MATCHER.addURI("com.wustzz.personprovider", "person/#", PERSON);
}
```

URI的authority一定要与创建的保持一致

URI的path(理解为数据集名)

标识码, 用于识别是复数还是单数操作

将会生成两个URI:

com.wustzz.personprovider/person 复数形式

com.wustzz.personprovider/person/id值 单数形式

【[返回](#)】

### ③ MyContentProvider6个方法的实现

onCreate方法

```
@Override  
public boolean onCreate() {  
    helper=new DBHelper(this.getContext(), "test.db", null, 1);  
    return true;  
}
```

## getType方法

@Override

```
public String getType(Uri uri) {
```

```
    switch ( MATCHER.match(uri) ) {
```

```
        case PERSONS:
```

```
            return "vnd.android.cursor.dir/person";
```

```
        case PERSON:
```

```
            return "vnd.android.cursor.item/person";
```

```
    default:
```

```
        throw new IllegalArgumentException("Unkwon Uri:" + uri.toString());
```

```
    }
```

```
}
```

说明：自定义的 MIME 类型的主类型必须是  
vnd.android.cursor.dir 代表多行数据  
或vnd.android.cursor.item 代表单行数据


@Override

```
public Cursor query(Uri uri, String[] projection, String selection,
                    String[] selectionArgs, String sortOrder) {
    SQLiteDatabase db = helper.getReadableDatabase();
    switch ( MATCHER.match(uri) ) {
        case PERSONS:
            return db.query("person", projection, selection, selectionArgs, null, null, sortOrder);
        case PERSON:
            long id = ContentUris.parseId(uri);    //使用ContentUris类的parse方法获得_id值
            String where = "_id=" + id;    //注： 创建person表时已经用的是"_id"
            if (selection != null && !"".equals(selection)) {
                where = selection + " and " + where;    //将_id参数补充到条件中
            }
            return db.query("person", projection, where, selectionArgs, null, null, sortOrder);
        default:
            throw new IllegalArgumentException("Unkwon Uri:" + uri.toString());
    }
}
```



@Override

```
public Uri insert(Uri uri, ContentValues values) {  
    SQLiteDatabase db = helper.getWritableDatabase();  
    switch ( MATCHER.match(uri) ) {  
        case PERSONS:  
            long rowid = db.insert("person", null, values);  
            Uri insertUri = ContentUris.withAppendedId(uri, rowid); // 得到新增记录的Uri  
            return insertUri;  
        default:  
            throw new IllegalArgumentException("Unkwon Uri:" + uri.toString());  
    }  
}
```



底层利用SQLiteDatabase的insert()方法添加数据




@Override

```
public int delete(Uri uri, String selection, String[] selectionArgs) {  
    SQLiteDatabase db = helper.getWritableDatabase();  
    int count = 0;  
    switch ( MATCHER.match(uri) ) {  
        case PERSONS:  
            count = db.delete("person", selection, selectionArgs);  
            return count;  
        case PERSON:  
            long id = ContentUris.parseId(uri);  
            String where = "_id=" + id;  
            if (selection != null && !"".equals(selection)) {  
                where = selection + " and " + where;  
            }  
            count = db.delete("person", where, selectionArgs);  
            return count;  
        default:  
            throw new IllegalArgumentException("Unkwon Uri:" + uri.toString());  
    }  
}
```

底层利用SQLiteDatabase的  
delete()方法删除数据

@Override

```
public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {  
    SQLiteDatabase db = helper.getWritableDatabase();  
    int count = 0;  
    switch (MATCHER.match(uri)) {  
        case PERSONS:  
            count = db.update("person", values, selection, selectionArgs);  
            return count;  
        case PERSON:  
            long id = ContentUris.parseId(uri);  
            String where = "_id=" + id;  
            if (selection != null && !"".equals(selection)) {  
                where = selection + " and " + where;  
            }  
            count = db.update("person", values, where, selectionArgs);  
            return count;  
        default:  
            throw new IllegalArgumentException("Unkwon Uri:" + uri.toString());  
    }  
}
```



底层利用SQLiteDatabase的  
update()方法更新数据

## 2. ContentProvider使用端

 TestContentProvider			
查询	添加	删除所有	修改
1	wust1	21	
2	wust2	22	
3	wust3	23	
4	wust4	24	
5	wust5	25	

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <LinearLayout    android:layout_width="match_parent"    android:layout_height="wrap_content" >
        <Button android:id="@+id/button1"
            android:layout_width="wrap_content"    android:layout_height="wrap_content"
            android:text="查询" />
        <Button android:id="@+id/button2"
            android:layout_width="wrap_content"    android:layout_height="wrap_content"
            android:text="添加" />
        <Button android:id="@+id/button3"
            android:layout_width="wrap_content"    android:layout_height="wrap_content"
            android:text="删除所有" />
        <Button android:id="@+id/button4"
            android:layout_width="wrap_content"    android:layout_height="wrap_content"
            android:text="修改" />
    </LinearLayout>
    <ListView    android:id="@+id/listView1"
        android:layout_width="match_parent"    android:layout_height="wrap_content" >
    </ListView>
</LinearLayout>
```

## ListView自定义布局 (listview.xml)

```
<?xml version="1.0" encoding="utf-8"?>
```

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

```
    android:layout_width="fill_parent"    android:layout_height="fill_parent"  
    android:orientation="horizontal" >
```

```
    <TextView    android:id="@+id/txtID"
```

```
        android:layout_width="100dip"        android:layout_height="wrap_content"  
        android:text="id"        android:textSize="20dp"        android:gravity="center"/>
```

```
    <TextView    android:id="@+id/txtName"
```

```
        android:layout_width="100dip"        android:layout_height="wrap_content"  
        android:text=" name"        android:textSize="20dp"        android:gravity="center"/>
```

```
    <TextView    android:id="@+id/txtAge"
```

```
        android:layout_width="fill_parent"        android:layout_height="wrap_content"  
        android:text=" age"        android:textSize="20dp"        android:gravity="center"/>
```

```
</LinearLayout>
```

```
//查询
Button btn1=(Button)findViewById(R.id.button1);
btn1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ListView listView=(ListView)findViewById(R.id.listView1);
        String[ ] from =new String[]{"_id", "name", "age"};
        int[ ] to =new int[] { R.id.txtID, R.id.txtName, R.id.txtAge };
        ContentResolver resolver = getContentResolver();
        Uri selectUri = Uri.parse("content://com.wustzz.personprovider/person");
        Cursor cursor=resolver.query(selectUri, new String[]{"_id","name","age"}, null, null, null);
        SimpleCursorAdapter adapter = new SimpleCursorAdapter(MainActivity.this,
                                                                    R.layout.listview, cursor, from , to);
        listView.setAdapter(adapter);
    }
});
```

//添加

```
Button btn2=(Button)findViewById(R.id.button2);
btn2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ContentResolver resolver = getContentResolver();
        Uri insertUri = Uri.parse("content://com.wustzz.personprovider/person");
        ContentValues values = new ContentValues();
        values.put("name", "QQQQ");
        values.put("age", 20);
        resolver.insert(insertUri, values);
        Toast.makeText(MainActivity.this, "添加完成", Toast.LENGTH_SHORT).show();
    }
});
```

删除所有

//删除所有

```
Button btn3=(Button)findViewById(R.id.button3);
btn3.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ContentResolver resolver = getContentResolver();
        Uri deleteUri = Uri.parse("content://com.wustzz.personprovider/person");
        int count=resolver.delete(deleteUri, null, null);
        Toast.makeText(MainActivity.this, "删除"+count+"条记录",
            Toast.LENGTH_SHORT).show();
    }
});
```



//修改

```
Button btn4=(Button)findViewById(R.id.button4);
btn4.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        ContentValues resolver = getContentResolver();
        Uri updateUri = Uri.parse("content://com.wustzz.personprovider/person");
        ContentValues cv = new ContentValues();
        cv.put("age", 10);
        int count=resolver.update(updateUri, cv, "name like ?",
                                new String[]{"wust%"});
        Toast.makeText(MainActivity.this, "删除"+count+"条记录",
                        Toast.LENGTH_SHORT).show();
    }
});
```

【完】