内容提供器

2018年4月28日 11:36

内容提供器（content Provider）主要用于在不同的应用程序之间实现数据共享的功能，它提供了一套完整的机制，允许一个程序访问另一个程序中的数据，同时还能保证被访问数据的安全性。
——Android实现跨程序共享数据的标准方式

1、运行时权限
当操作涉及到用户设备的安全性时，就需要在AndroidManifest.xml中加入权限声明，否则程序会崩溃

权限组名	权限名
CALENDAR	READ_CALENDAR WRITE_CALENDAR
CAMERA	CAMERA
CONTACTS	READ_CONTACTS WRITE_CONTACTS GET_ACCOUNTS
LOCATION	ACCESS_FINE_LOCATION ACCESS_COARSE_LOCATION
MICROPHONE	RECORD_AUDIO
PHONE	READ_PHONE_STATE CALL_PHONE READ_CALL_LOG WRITE_CALL_LOG ADD_VOICEMAIL USE_SIP PROCESS_OUTGOING_CALLS
SENSORS	BODY_SENSORS
SMS	SEND_SMS RECEIVE_SMS READ_SMS RECEIVE_WAP_PUSH RECEIVE_MMS
STORAGE	READ_EXTERNAL_STORAGE WRITE_EXTERNAL_STORAGE

```
public void onClick(View v) {
        //ContextCompat.checkSelfPermission（）判断用户是不是已经授权
        //没有授权的话，则ActivityCompat.requestPermissions（）向用户申请授权
        if (ContextCompat.checkSelfPermission(MainActivity.this,Manifest.permission.CALL_PHONE) !=
PackageManager.PERMISSION_GRANTED){
          ActivityCompat.requestPermissions(MainActivity.this,new String[]{Manifest.permission.CALL_PHONE},1);
        }else {
          call();
```

2、访问其他程序的数据
内容提供器：**使用现有的内容提供器来读取和操作相应程序中的数据**
（获取到该应用程序的内容URI，然后借助ContentResolver进行CRUD操作）
创建自己的内容提供器给程序的数据提供外部访问接口
要访问内容提供器中共享的数据，要借助**contentResolver**类（通过Context中的getcontentResolver（）方法获取该类的实例）
contentResolver不接收表名参数，而是使用Uri参数代替（内容URI：给内容提供器中的数据建立了唯一标识符：authority（对不同的应用程序做区分，避免冲突，则采用程序包名的方式命名）和path（对同一应用程序中不同的表做区分，添加在authority后面））
内容URI标准格式：content：//com.example.app.provider/table1 >>>>>>>>>>表示调用方期望访问的是com.example.app这个应用的table1表中的数据

内容URI字符串 >>>解析成Uri对象（ Uri.parse()方法 **）**

```
Uri uri = Uri.parse("content：//com.example.app.provider/table1")
//得到cursor对象后，读取数据
```

```
Cursor cursor = getContentResolver().query(uri,projection,selection,selectionArgs,sortOrder); //查询表中数据
if(cursor != null){
while(cursor.moveToNext()){
String column1 = cursor.getString(cursor.getColumnIndex("column1"));
int column2 = cursor, getInt(cursor.getColumnIndex("column2"));
}
cursor.close();
}
```

//添加一条数据

```
ContentValues values = new ContentValues();
values.put("column1","text");
values.put("column2",1);
getContentResolver().insert(uri,values);
```

//更新数据

```
ContentValues values = new ContentValues();
values.put("column1","");
getContentResolver().update(uri,values,"column1 = 1 ? and column2 = ?",new String[] {"text","1"});
```

//删除数据

```
getContentResolver().delete(uri,"column2 = ?", new String[]{"1"});
```

3、创建内容提供器

> 新建一个类去继承contentProvider的方式来创建一个内容提供器

> contentProvider类中有6个抽象方法

//初始化内容提供器的时候调用（完成对数据库的创建和升级等操作）注意！！！只有当存在ContentResolver尝试访问程序中的数据时，内容提供器才会被初始化

```
@Override
public boolean onCreate(){
    return  false ;
}
```

//从内容提供器中查询数据（uri参数来确定查询哪张表，projection参数用于确定查询哪些列，selection和selectionArgs参数用于约束查询哪些行，sortOrder参数用于对结果进行排序，查询的结果存放在Cursor对象中返回）

```
@Override
public Cursor query (@NonNull Uri uri, @Nullable String[] projection ,@Nullable selection , @Nullable String[]
selectionArgs , @Nullable String sortOrder) {
    return null;
}
```

//根据传入的内容URI来返回相应的MIME类型

```
@Override
public String getType(@NonNull Uri uri){
    return null;
}
```

//向内容提供器中添加一条数据（使用uri参数来确定要添加到的表，待添加的数据保存在values参数中。添加完成后，返回一个用于表示这条新纪录的URI）

```
@Nullable
@Override
public Uri insert(@NonNull Uri uri, @Nullable ContentValues values){
    return null;
}
```

//从内容提供器中删除数据（使用uri参数来确定删除哪一张表中的数据，selection和selectionArgs参数用于约束删除哪些行，被删除的行数将作为返回值返回）

```
@Override
public int delete(@NonNull Uri uri,@Nullable String selection , @Nullable String[] selectionArgs){
    return 0;
}
```

//更新内容提供器中已有的数据（使用uri参数来确定更新哪一张表中的数据，新数据保存在values参数中，selection和selectionArgs参数用于约束更新哪些行，受影响的行数将作为返回值返回）

```
@Override
public int update (@NonNull Uri uri, @Nullable ContentValues values, @Nullable String selection, @Nullable String[]
selectionArgs){
    return 0;
}
```

content：//com.example.app.provider/table1/1 >>>>>>>>>>表示调用方期望访问的是com.example.app这个应用的table1表中id为1的数据

- *：表示匹配任意长度的任意字符
- #：表示匹配任意长度的数字
- UriMatcher实现匹配内容URI<addURI(authority，path，一个自定义代码)方法><match（）判断调用方访问的是哪张表>

getType（）方法，是所有的内容提供器都必须提供的方法，用于获取Uri对象所对应的MIME类型。

- 必须以vnd开头
- 如果内容URI以路径结尾，则后接android.cursor.dir/，如果内容URI以id结尾，则后接android.cursor.item/
- 最后接上vnd.<authority>.<patch>