河北师范大学软件学院
Software College of Hebei Normal University

Android 基础开发

第九章
ContentProvider

教学目标

- 掌握Android中ContentProvider的使用

目录

数据共享

- Android中存在多个应用程序，应用程序间的数据如何共享呢？

联系人数据

数据共享

- 为了在应用程序之间共享数据，Android提供了ContentProvider，这是一种不同应用之间共享数据的标准API：
  - 当应用希望提供数据时，就提供ContentProvider。
  - 其它应用通过ContentResolver来操作。
- 注意：
  - ContentProvider需要在AndroidManifest.xml中注册。
  - 一旦应用提供CP，不论应用启动与否，都可被操作。

数据共享标准

- ContentProvider是应用间共享数据的标准，它以Uri的形式对外提供数据，其他应用程序使用ContentResolver根据Uri去访问指定数据。
- ContentProvider是单例模式的，多个ContentResolver请求数据时，是委托给同一个CP对象来操作的。

ContentProvider简介

- 系统内置的ContentProvider，如联系人信息。
- 自定义ContentProvider。
  - 定义自己的类继承ContentProvider类。

```
boolean onCreate()
Uri insert(uri, values)
int delete(uri, selection, selectionargs)
int update(uri, values, selection, selectionargs)
Cursor query(uri,projection,selection,selectionargs,sortorder)
String getType(uri)
```

  - 向Android系统注册ContentProvider。

```
<provider android:name=" " android:authorities=" "/>
```

Uri简介

- Uri的结构和网站URL命名规则类似：
- URL举例：
  - http://www.baidu.com/index.php
- Uri举例：content://org.edu.provider/words
  - content://:这部分是Android固定的
  - org.edu.provider：这部分是ContentProvider中的authority
  - words：资源部分，根据资源不同这部分不同

Uri简介

- Uri可以表达的功能有很多.
  - content://org.edu.provider/words/2
  - content://org.edu.provider/words/2/name
  - content://org.edu.provider/words
- 通过Uri提供的静态方法parse()来实现将字符串转换为Uri。
  - Uri uri = Uri.parse("content://org.edu.p../words")

操作Uri的工具类

- CP不对Uri做判断，Uri工具类可协助CP操作Uri。
- UriMatcher工具类
  - UriMatcher(UriMatcher.NO_MATCH)：构造方法。
  - void addURI(String authority,String path,int code)：用于向UriMatcher对象注册Uri。
  - Int match(Uri uri)：根据注册的Uri判断指定Uri对应的标识码，找不到则返回-1。

操作Uri的工具类

- UriMatcher工具类

```
UriMatcher matcher = new UriMatcher(UriMatcher.NO_MATCH);
// 用于注册两个Uri
matcher.addURI("org.providers.prods", "words", 1);
matcher.addURI("org.providers.prods", "word/#", 2);
matcher.match(Uri.parse("content://org.providers.prods/words")); // 返回1
matcher.match(Uri.parse("content://org.providers.prods/word/5")); // 返回2
```

操作Uri的工具类

- CP不对Uri做判断，Uri工具类可协助CP操作Uri
- ContentUris工具类
  - withAppendedId(uri,id)：为路径uri加上ID部分
  - parseId(uri)：从指定Uri解析出所包含的ID值

```
Uri uri
     = Uri.parse("content://org.providers.prods/word");
Uri resultUri = ContentUris.withAppendedId(uri,5);
// 返回resultUri为：content://org.providers.prods/word/5
Long id = ContentUris.parseId(resultUri);
// 返回5
```

ContentResolver简介

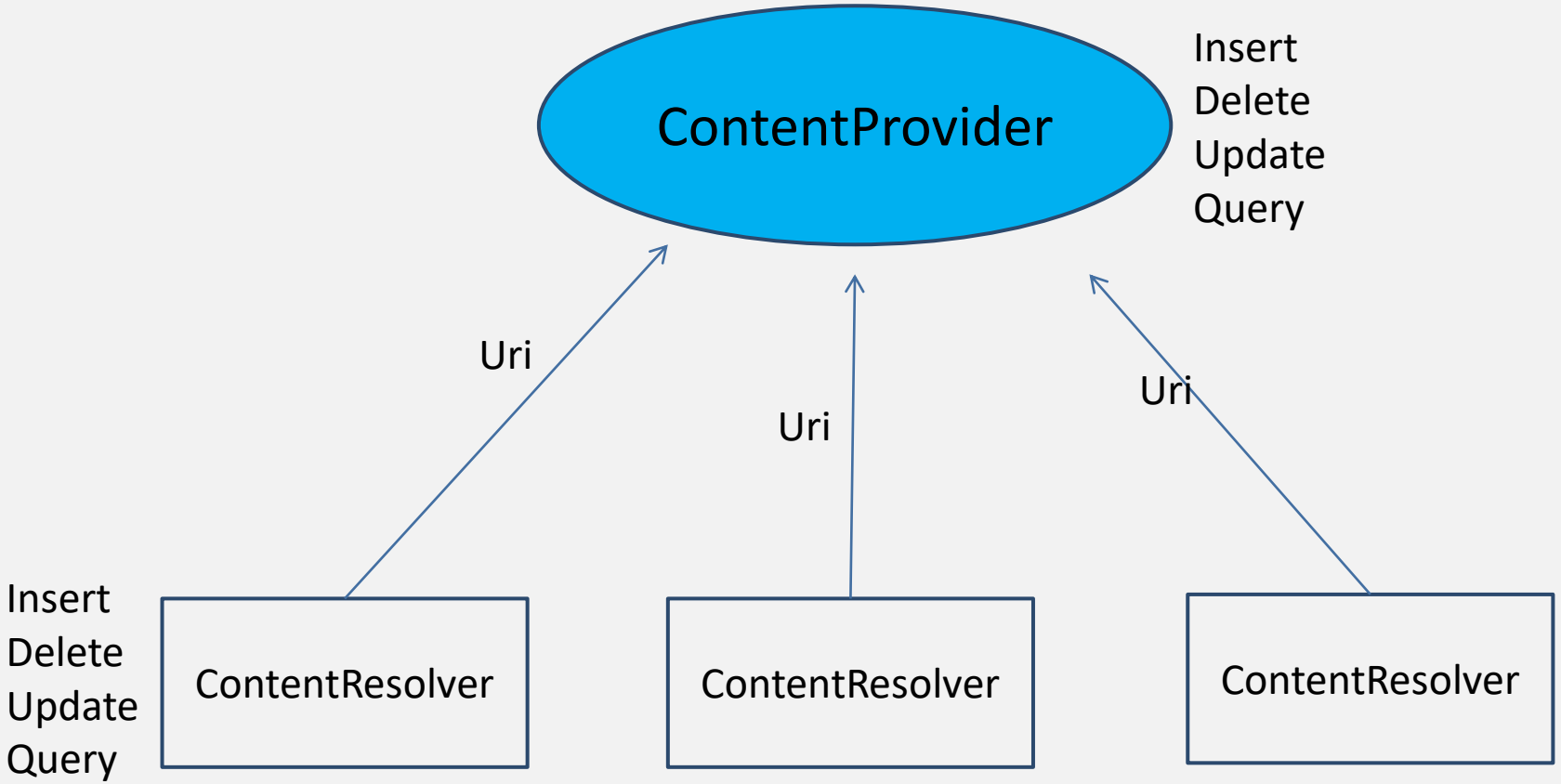
- 应用通过ContentResolver操作CP暴漏的数据。
- 获得ContentResolver的方法：

```
context.getContentResolver()
```

- 通过ContentResolver操作数据：

```
insert(uri, values)
delete(uri , where , selectionargs)
update(uri, values , where ,selectionargs)
query(uri,projection, selection ,selectionargs, order)
```

Android数据共享结构



目录

获得系统的CP的数据

- Android系统中会给应用提供一些开放的数据，采用CP的形式。
- 通过ContentResolver访问系统CP的步骤：
  - 调用Activity的ContentResolver获取CR对象。
  - 调用CR中的insert，delete，update，query方法获得系统CP提供的数据。
- 注意：
  - 要获得系统CP提供的数据，必须要了解该CP的Uri。

获得系统的CP的数据

- CR常用API：

```
// 向Uri对应的ContentProvider中插入values对应的数据
insert(Uri uri,ContentValues values)
// 删除Uri对应的ContentProvider中where条件匹配的数据
delete(Uri uri, String where, String[] selectionArgs)
// 更新Uri对应的ContentProvider中where条件匹配的数据
update(Uri uri, ContentValues values, String where,
        String[] selectionArgs)
// 查询Uri对应的ContentProvider中where条件匹配的数据
query(Uri uri, String[] projection, String selection,
      String[] selectionArgs, String sortOrder)
```

获得系统的CP的数据

- 编码中用到的常量
  - Data中的常量
    - Data._ID： "_id"
    - Data.DISPLAY_NAME： “display_name”
    - Data.DATA1： “data1”
    - Data.DATA2： “data2”
    - Data.RAW_CONTACT_ID： “raw_contact_id”
    - Data.MIMETYPE： “mimetype”
  - MIMETYPE（稍后解释）

使用CP管理联系人

- Android系统中Contacts应用来管理联系人，同时提供相应CP。
- 此CP支持的Uri如下：
  - ContactsContract.Contacts.CONTENT_URI：管理联系人的Uri。
  - ContactsContract.CommonDataKinds.Phone.CONTENT_URI：管理联系人的电话的Uri。
  - ContactsContract.CommonDataKinds.Email.CONTENT_URI：管理联系人的Email的Uri。

联系人信息数据库信息

- 通讯录存放位置
  - /data/data/com.android.providers.contacts/databases/contacts2.db
- 表结构1：（共3张表）

mimetypes表

RecNo	_id	mimetype
Click here to define a filter		
1	1	vnd.android.cursor.item/email_v2
2	2	vnd.android.cursor.item/im
3	3	vnd.android.cursor.item/postal-address_v2
4	4	vnd.android.cursor.item/photo
5	5	vnd.android.cursor.item/phone_v2
6	6	vnd.android.cursor.item/name
7	7	vnd.android.cursor.item/organization
8	8	vnd.android.cursor.item/nickname
9	9	vnd.android.cursor.item/group_membership

联系人信息数据库信息

- 通讯录数据的MIMETYPE
  - 电话：vnd.android.cursor.item/phone_v2
  - 姓名：vnd.android.cursor.item/name
  - 邮件：vnd.android.cursor.item/email_v2
  - 通信地址：vnd.android.cursor.item/postal-address_v2
  - 组织：vnd.android.cursor.item/organization
  - 照片：vnd.android.cursor.item/photo

联系人信息数据库信息

- 表结构2：（共3张表）

data表

RecNo	_id	package_id	mimetype_id	raw_contact_id	is_primary	is_super_primary	data_version	data1	data2	data3	data4	data5	data6	data7	data8	data9	da
Click here to define a filter																	
1	1	<null>	6	1	0	0	1	xdong xia	xdong	xia	<null>	<null>	<null>	<null>	<null>	<null>	1
2	2	<null>	3	1	0	0	2	Street City, Region 1234 6	1	<null>	Street	<null>	<null>	City	Region	123456	<n
3	3	<null>	5	1	0	0	0	12345678	2	<null>	87654321	<null>	<null>	<null>	<null>	<null>	<n
4	4	<null>	7	1	0	0	1	Company	1	<null>	Developer	<null>	<null>	<null>	<null>	<null>	<n
5	5	<null>	1	1	0	0	0	xiardong@xiardong.com	2	<null>	<null>	<null>	<null>	<null>	<null>	<null>	<n

- data：存放具体的数据
  - raw_contact_id属性用来连接raw_contacts表，每条记录表示一个具体数据；我们主要的数据（email、phone等)都存放在data表；
  - data1属性存放总数据；
  - data2属性：
    - -如果此记录存放姓名，则data2存放名；
    - -如果此记录存放电话，则data2存放类型，比如手机、家电；
    - -如果此记录存放组织，则data2存放类型，比如公司、其他；
    - -如果此记录存放地址，则data2存放类型，比如住宅，单位等

联系人信息数据库信息

- 表结构3：（共3张表）

raw_contacts表

_id	account_id	sourceid	raw_contact...	version	dirty	deleted	contact_id	aggregation_...	aggregation_...	custom_ringt...	send_to_voic...	times_contac...	last_time_co...	starred	display_name	display_nam...
2	1		0	12	1	1		3	1		0	0		0	name	name
6	1		0	6	1	0	6	0	0		0	0		0	name	name

– raw_contacts：存放联系人的ID
  - _id属性为主键，声明为autoincrement，即不需要手动设置，其他属性也不需要手动设置就有默认值；
– display_name：存放联系人的姓名

使用CP管理联系人

```
Cursor cur = getContentResolver().query(
    ContactsContract.Contacts.CONTENT_URI,
    null, null,null, null);
do{
    String str = cur.getString(cur.getColumnIndex(
            ContactsContract.Contacts.DISPLAY_NAME));
    Log.e("Name", str);
} while (cur.moveToNext());
```

注意：需要在AndroidManifest.xml文件中添加读取联系人信息的权限

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

使用CP管理多媒体内容

- Android提供了Camera程序来支持拍照、视频，用户的多媒体信息都存放在固定的位置，并且提供了CP。
- 此CP支持的Uri如下：
  - MediaStore.Audio.Media.EXTERNAL_CONTENT_URI
  - MediaStore.Audio.Media.INTERNAL_CONTENT_URI
  - MediaStore.Images.Media.EXTERNAL_CONTENT_URI
  - MediaStore.Images.Media.INTERNAL_CONTENT_URI
  - MediaStore.Video.Media.EXTERNAL_CONTENT_URI
  - MediaStore.Video.Media.INTERNAL_CONTENT_URI
  - Content://sms/outbox 发送箱短信URI
  - Content://sms/sent 收信箱短信URI
  - Content://sms/draft 草稿箱短信URI

目录

创建ContentProvider的子类

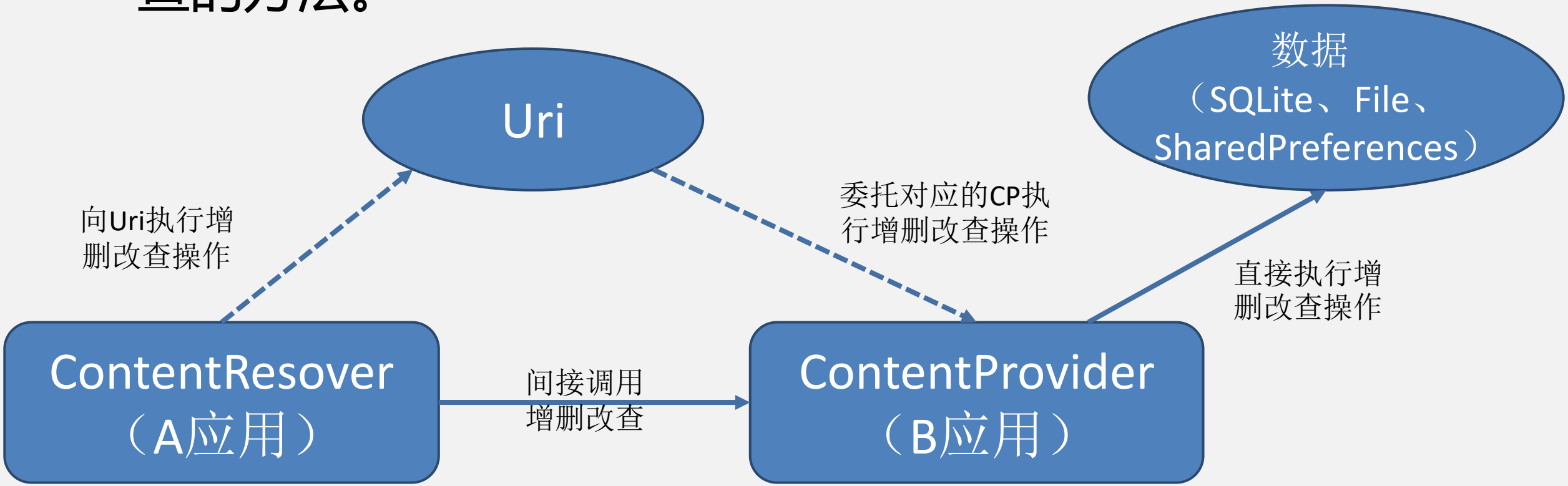
- 自定义一个类，继承ContentProvider
- 实现onCreate()、getType()方法
- 实现增、删、改、查的方法
- Uri的处理
  - 借助UriMatcher工具类

自定义ContentProvider

- 创建ContentProvider的步骤：
  - 自定义一个类，继承ContentProvider，实现增、删、改、查的方法。
- 配置ContentProvider
  - 在AndroidManifest.xml中注册该ContentProvider。
- 使用ContentResolver调用方法
  - 获取ContentResolver对象
  - 通过固定的URI进行数据操作

STEP1：创建ContentProvider

- 自定义一个类，继承ContentProvider，实现增、删、改、查的方法。



STEP1：创建ContentProvider

- 自定义一个类，继承ContentProvider，实现增、删、改、查的方法。
  - getType(Uri, uri)：如果uri操作的是多条数据，则MIME类型以vnd.android.cursor.dir/开头；如果uri操作的是单条数据，则MIME类型以vnd.android.cursor.item/开头。
  - query()、update()、delete()、insert()直接操作数据，实现其方法。

STEP2：配置ContentProvider

- 在AndroidManifest.xml中注册该ContentProvider

```
<provider
    android:name=".DictProvider"
    android:authorities=
            "com.example.providers.dictprovider"
    android:exported="true" />
```

  - name： CP的实现类的类名
  - authorities：指定CP对应的Uri，相当于分配了一个域名
  - exported：是否允许该CP被其他应用调用

STEP3：使用ContentResolver调用方法

- 获取ContentResolver对象
- 通过固定的Uri进行数据操作

```
// 获取系统的ContentResolver对象
ContentResolver resolver = getContentResolver();
// 借助ContentValues添加数据，数据封装在ContentValues对象中
ContentValues values = new ContentValues();
values.put(key,value);// 参数：(“字段名”，字段值);
values.put(key,value);
resolver.insert(uri, values);
Resolver.update(…);
```

CP工具类

- 为了确定ContentProvider匹配的Uri，使用UriMatcher工具类：
  - void addURI(authority, path , code)
  - int match(uri)

CP工具类

```
UriMatcher sMatcher = new UriMatcher(UriMatcher.NO_MATCH);
sMatcher.addURI("com.lww.provider.personprovider", "person", 1);
sMatcher.addURI("com.lww.provider.personprovider", "person/#",
                2);
switch (sMatcher.match(Uri.parse(
    "content://com.ljq.provider.personprovider/person/10"))) {
    case 1
       break;
    case 2
       break;
    default: // 不匹配
       break;
}
```

CP工具类

- 为了操作Uri，Android使用ContentUris工具类：
  - withAppendedId(uri,id)
  - parseId(uri)

```
Uri uri = Uri.parse("content://com.
lww.provider.personprovider/person")
Uri resultUri = ContentUris.withAppendedId(uri, 10);
// 生成后的Uri为：
content://com.ljq.provider.personprovider/person/10
```

```
Uri uri =
Uri.parse("content://com.lww.provider.personprovider/person/10")
long personid = ContentUris.parseId(uri); // 获取的结果为:10
```

目录

监听ContentProvider的数据

- 通过ContentObserver进行CP数据修改的监听。

联系人数据

监听ContentProvider的数据

- 如果CP的数据进行了修改，程序会调用如下代码：
  - getContext().getContentResolver().notifyChange()
  - 作用：通知所有Uri的监听者，CP数据发生改变
- 利用ContentObserver基类，监听数据改变
  - 重写onChange(Boolean selfChange)

监听ContentProvider的数据

- 实现监听CP的步骤如下：
  - 自定义类继承ContentObserver类，重写onChange方法，也就是当CP数据改变后的回调方法
  - 通过ContentResolver向Uri注册ContentObserver监听器
  - registerContentObserver(uri,notifyForDescendents,observer)
    - uri：该监听器所监听的CP的Uri
    - notifyForDescendents：子Uri数据改变时是否触发
    - observer：监听器实例

内容回顾

1. 数据共享
2. 系统中的ContentProvider
3. 自定义ContentProvider
4. 监听ContentProvider的数据

Thank you!