ContentProviders 用于保存和获取数据，并使其对所有应用程序可见。（应用程序之间共享数据）

**根据官方文档：**

Content providers负责管理结构化数据的访问，Cotent Providers封装数据并提供一套定义数据安全的机制。Content Providers是一套不同进行间数据访问的接口。Content Providers为数据跨进程访问提供了一套安全的访问机制，对数据组织和安全访问提供了可靠的保证。

那什么地方才适合用Content Providers呢？怎么使用Content Providers呢？

1. 什么情况下需要使用Content Providers？
2. 你想为其它应用程序提供复杂数据或者文件。
3. 你想允许用户从你的应用程序中拷贝复杂数据到其它应用中。
4. 你想使用搜索框架提供自定义的查询建议功能。
5. 怎么使用？

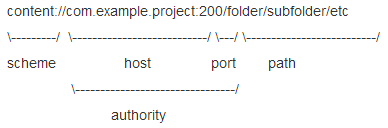
官方文档：当你想要通过Content Providers访问数据时，在应用程序的上下文（Context）中使用ContentResolver对象为客户端（client）与Provider进行交互。ContentResolver对象通过实现**抽象类ContentProvide**r的一个实例来访问Provider。Provider对象从客户端(client)接收数据请求，执行请求操作并且返回请求结果。



Content Providers 通过URI（统一资源标识符）来访问数据，URI可以理解为访问数据的唯一地址，URI有authority和数据地址构成，关于authority可以理解成网站地址中的主机地址，而数据地址可以理解成某一个页面的子地址，二者共同构成了一个完整的访问地址。

对于anthority的命名，官方有规范性要求：用程序所在的包名+provider来命名，例如：包名：com.lhqj.<appname>，那么authority的名称为：com.lhqj.<appname>.provider。

2.1、URI的格式



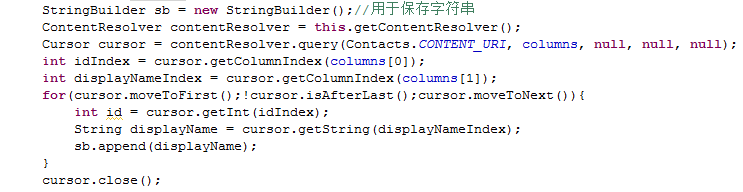
content://<authority>/<path>/<id>，path就是数据的路径，比如是一张表，id就是该表中之间为id的一行，也可以理解为一个实体对象。

对于id，Android提供了一系列的辅助类（在android.provider包下），里面包含了很多以类变量形式给出的查询字符串。特别是ContentUris.withAppendedId()和Uri.withAppendedPath()方法，能够轻松的将ID加到URI中。例如：content://contacts/people/45等价于：Uri person = ContentUris.withAppendedId(People.CONTENT\_URI,  45);

2.2、Android中可以分为2种ContentProvider，一种是系统提供一些常见的URI(比如：联系人URI：ContactsContract.Contacts.CONTENT\_URI 、联系人的电话的Uri：ContactsContract.CommonDataKinds.Phone.CONTENT\_URI 等)。另外一种是自定义ContentProvider，根据用户的需求查询特定的数据结构。

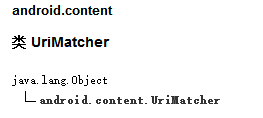
2.2.1、查询手机联系人

通过getContentResolver()获取ContentResolver对象，用系统管理联系人的URI查询手机联系人，返回一个Cursor对象。



2.2.2、自定义ContentProvider

**1）涉及URI时需要用到的工具类：UriMatcher类和ContentUris类。**



**UriMatcher类**用于匹配Uri，UriMatcher类用法如下：

1. 定义UriMatcher对象：

 构造函数：创建URI树的根节点

a)       Public UriMatcher()：默认根节点编码为-1；

b)       Public UriMatcher(int code):code参数指定根节点的编码。

UriMatcher matcher = new UriMatcher(UriMatcher.NO\_MATCH)；

1. 注册完需要匹配的Uri：Public void addURI(String authority,String path,int code)

URI可以是精确的字符串，uri中带有\*表示可匹配任意text，#表示只能匹配数字。

Authority：用于匹配的域名；

Path：匹配路径，\*表示text的占位符，#表示使用数字的占位符；

Code：当使用匹配成功后返回code，值需要大于0，否则抛出IllegalArgument异常。

此方法将authority按照”/”进行拆分，然后将拆分后的每一部分保存到UriMatcher类型的ArrayList中；在添加的时候会判断当前authority是否已经添加过，若已加则break；若未添加过，则判断是否含有”#”则将其标识成1代表域名后面跟随的是数字;”\*”标识成2,代表域名后面跟随的是文本；0代表后面没有跟随数据；最后创建一个新的UriMatcher对象添加到集合中。

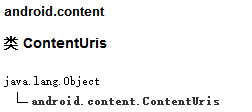
* matcher.addURI("com.bing.procvide.personprovider", "person", 1);//如果match()方法匹配<content://com.bing.procvide.personprovider/person>路径，返回匹配码为1。
* matcher.addURI("com.bing.provider.personprovider", "person/#", 2);//#号为通配，如果match()方法匹配<content://com.bing.provider.personprovider/person/230>路径，返回匹配码为2。

3、使用方法 Public int match(Uri uri);// Uri：指定需要匹配的url；返回值：在使用addURI时产生的code，若没有匹配则返回-1。

在match就是相当于遍历已经注册存放在ArrayList<UriMatcher>中的URI。

**具体操作为：**

使用uri.getPathSegments()获取uri中各段存入list中，若list size为0或uri的Authority为null则返回默认值（此默认值在new时指定，若为指定则为-1）；然后遍历ArrayList<UriMatcher>进行匹配uri。



**ContentUris类：**ContentUris类用于操作Uri路径后面的ID部分，他有2个比较实用的方法：

（1）withAppendedId(uri, id)用于为路径加上ID部分。

Uri uri = Uri.parse("[content://com.example.provider.personprovider/person](content://com.bing.provider.personprovider/person)")

Uri resultUri = ContentUris.withAppendedId(uri, 10);

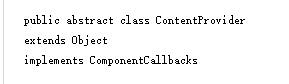
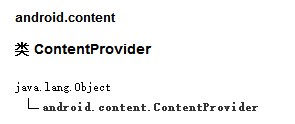
//生成后的resultUri为：[content://com.example.provider.personprovider/person/10](content://com.bing.provider.personprovider/person/10)

（2）parseId(uri)方法用于从路径中获取ID部分。

Uri uri = Uri.parse("[content://com.example.provider.personprovider/person/10](content://com.ljq.provider.personprovider/person/10)")

long personid = ContentUris.parseId(uri);//获取的结果为:10

**2）自定义ContentProvider共享数据，继承ContentProvider抽象类，实现它的方法。**



**（1）主要方法：**

1. public boolean onCreate()：该方法在ContentProvider创建后就会被调用，Android开机后，ContentProvider在其它应用第一次访问它时才会被创建。
2. public Uri insert(Uri uri, ContentValues values)：该方法用于供外部应用往ContentProvider添加数据。
3. public int delete(Uri uri, String selection, String[] selectionArgs)：该方法用于供外部应用从ContentProvider删除数据。
4. public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs)：该方法用于供外部应用更新ContentProvider中的数据。
5. public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)：该方法用于供外部应用从ContentProvider中获取数据。
6. public String getType(Uri uri)：该方法用于返回当前Url所代表数据的MIME类型。

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ContentValues\*\*\*\*\*\*\*\*\*\*\*/**

ContentValues：按键值对的方式存储。put(String key,XXX vaule); get(String key);

**（2）MIME类型：**全称Multipurpose Internet Mail Extensions，多功能Internet 邮件扩充服务。它是一种多用途网际邮件扩充协议，在1992年最早应用于电子邮件系统，但后来也应用到浏览器。MIME类型就是设定某种扩展名的文件用一种应用程序来打开的方式类型，当该扩展名文件被访问的时候，浏览器会自动使用指定应用程序来打开。多用于指定一些客户端自定义的文件名，以及一些媒体文件打开方式。（百度）

简单的来说就是自己定义一个类型，并与该类型设定一个对应的数据集。即，我们通过URI匹配之后得到的类型，去执行对应操作。

数据集的MIME类型字符串则应该以vnd.android.cursor.dir/开头

单一数据的MIME类型字符串应该以vnd.android.cursor.item/开头

**（3）ContentResolver操作ContentProvider中的数据**

1、当外部应用需要对ContentProvider中的数据进行添加、删除、修改和查询操作时，可以使用ContentResolver 类来完成，要获取ContentResolver 对象，可以使用Activity提供的getContentResolver()方法。

2、ContentResolver 类提供了与ContentProvider类相同签名的四个方法：

1) public Uri insert(Uri uri, ContentValues values)：该方法用于往ContentProvider添加数据。  
2) public int delete(Uri uri, String selection, String[] selectionArgs)：该方法用于从ContentProvider删除数据。  
3) public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs)：该方法用于更新ContentProvider中的数据。  
4) public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)：该方法用于从ContentProvider中获取数据。

这些方法的第一个参数为Uri，代表要操作的ContentProvider和对其中的什么数据进行操作。

**例如：**Uri.parse("content://com.ex.providers.personprovider/person/10")，那么将会对主机名为com.ex.providers.personprovider的ContentProvider进行操作，操作的数据为person表中id为10的记录。

3、如果ContentProvider的访问者需要知道ContentProvider中的数据变化，可以在ContentProvider发生数据变化时调用getContentResolver.notifyChange(uri,null)来通知注册此URI上的访问者，例如：

public class PersonContentProvider extends ContentProvider {

   public Uri insert(Uri uri, ContentValues values) {

      db.insert("person", "personid", values);

      getContext().getContentResolver().notifyChange(uri, null);

   }

}

**上述是通知发出，那访问者要得到该数据变化的通知，**则必须使用ContentObserver对数据（数据采用uri描述）进行监听，当监听到数据变化通知时，系统就会调用ContentObserver的onChange()方法获得通知,例如：

getContentResolver().registerContentObserver(Uri.parse("<content://com.ljq.providers.personprovider/person>"),true, new PersonObserver(new Handler()));**//对ContentResolver注册监听**

public class PersonObserver extends ContentObserver{

   public PersonObserver(Handler handler) {

      super(handler);

   }

   public void onChange(boolean selfChange) {

      //此处可以进行相应的业务处理

   }

}

3、综合例子