9-ЛЕКЦИЯ. КОНТЕНТ ПРОВАЙДЕРЫ В ПРИЛОЖЕНИЯХ АНДРОИД И ИХ ИСПОЛЬЗОВАНИЕ

9.1.Понятие контент-провайдера

Контент-провайдер управляет доступом к хранилищу данных. Для реализации провайдера в Android приложении должен быть создан набор классов в соответствии с манифестом приложения. Один из этих классов должен быть наследником класса ContentProvider, который обеспечивает интерфейс между контент-провайдером и другими приложениями.

Основное назначение этого компонента приложения заключается в предоставлении другим приложениям доступа к данным, однако ничто не мешает в приложении иметь активность, которая позволит пользователю запрашивать и изменять данные, находящиеся под управлением контент-провайдера.

В мобильных приложениях контент-провайдеры необходимы в следующих случаях:

- приложение предоставляет сложные данные или файлы другим приложениям;
- приложение позволяет пользователям копировать сложные данные в другие приложения;
- приложение предоставляет специальные варианты поиска, используя поисковую платформу (framework).

Если приложение требует использования контент-провайдера, необходимо выполнить несколько этапов для создания этого компонент.

Данные, с которыми работают контент-провайдеры, могут быть организованы двумя способами:

- Данные представлены файлом, например, фотографии, аудио или видео. В этом случае необходимо хранить данные в собственной области памяти приложения. В ответ на запрос от другого приложения, провайдер может возвращать ссылку на файл.
- Данные представлены некоторой структурой, например, таблица, массив. В этом случае необходимо хранить данные в табличной форме. Строка таблицы представляет собой некоторую сущность, например, сотрудник или товар. А столбец - некоторое свойство этой сущности, например, имя сотрудника или цена товара. В системе Android общий способ хранения подобных данных - база данных SQLite, но можно использовать любой способ постоянного хранения.

Создание класса-наследника от класса ContentProvider напрямую или через любого его потомка. При этом в реализации класса необходимо переопределить (т. е. написать свою реализацию) обязательные методы.

query()	- метод, извлекающий данные из провайдера, в качестве аргументов получает таблицу, строки и столбцы, а также порядок сортировки результата, возвращает объект типа Cursor.
insert()	- метод, добавляющий новую строку, в качестве аргументов получает таблицу, и значения элементов строки, возвращает URI добавленной строки.
update()	- метод, обновляющий существующие строки, в качестве аргументов получает таблицу, строки для обновления и новые

	значения элементов строк, возвращает количество обновленных строк.
delete()	- метод, удаляющий строки, в качестве аргументов принимает таблицу и строки для удаления, возвращает количество удаленных строк.
getType()	- метод, возвращающий String в формате MIME, который описывает тип данных, соответствующий URI.
onCreate()	- метод, вызываемый системой, сразу после создания провайдера, включает инициализацию провайдера. Стоит отметить, что провайдер не создается до тех пор, пока объект ContentResolver не попытается получить к нему доступ.

Созданный контент-провайдер управляет доступом к структурированным данным, выполняя обработку запросов от других приложений. Все запросы, в конечном итоге, вызывают объект ContentResolver, который в свою очередь вызывает подходящий метод объекта ContentProvider для получения доступа.

Все вышеперечисленные методы, кроме onCreate(), вызываются приложением-клиентом. И все эти методы имеют такую же сигнатуру, как одноименные методы класса ContentResolver.

Определение строки авторизации провайдера, URI для его строк и имен столбцов. Если от провайдера требуется управление намерениями, необходимо определить действия намерений, внешние данные и флаги. Также необходимо определить разрешения, которые необходимы приложениям для доступа к данным провайдера. Все эти значения необходимо определить как константы в отдельном классе, этот класс в последствии можно предоставить другим разработчикам.

9.2.Создание контент провайдера

Наше приложение может сохранять разнообразую информацию о пользователе, какие-то связанные данные в файлах или настройках. Однако ОС Android уже хранит ряд важной информации, связанной с пользователем, к которой имеем доступ и которую мы можем использовать. Это и списки контактов, и файлы сохраненных изображений и видеоматериалов, и какие-то отметки о звонках и т.д., то есть некоторый контент. А для доступа к этому контенту в ОС Android определены провайдеры контента (content provider)

В Android имеются следующие встроенные провайдеры, определенные в пакете android.content:

- AlarmClock: управление будильником
- Browser: история браузера и закладки
- CalendarContract: каледарь и информаци о событиях
- CallLog: информация о звонках
- ContactsContract: контакты
- MediaStore: медиа-файлы
- SearchRecentSuggestions: подсказки по поиску
- Settings: системные настройки
- UserDictionary: словарь слов, которые используются для быстрого набора
- VoicemailContract: записи голосовой почты

Контакты в Android обладают встроенным API, позволяющим производить чтения и изменение списка контактов. Все контакты хранятся в базе данных SQLite, однако они не

представляют единой таблицы. Для контактов отведено три таблицы, связанных отношением один-ко-многим: таблица для хранения информации о людях, таблица их телефонов и и таблица адресов их электоронных почт. Но благодаря имеющемуся в Android API мы можем абстрагироваться от связей между таблицами.

Итак, для доступа к контактам нам надо установить соответствующие разрешения в файле манифеста приложения:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.eugene.sqliteapp" >

  <uses-permission android:name="android.permission.READ_CONTACTS"
android:maxSdkVersion="21" />
  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <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>
    </activity>
  </application>

</manifest>
```

Для вывода списка контактов определим следующую разметку интерфейса:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

  <TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/contact_list">
  </TextView>

  <ListView
    android:id="@+id/contactList"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
  </ListView>

</LinearLayout>
```

Для вывода списка контактов воспользуемся элементом ListView. И в классе activity получим контакты:

```
package com.example.eugene.sqliteapp;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

import android.provider.ContactsContract;
import android.database.Cursor;
import android.widget.ListView;
import android.widget.ArrayAdapter;
import java.util.ArrayList;

public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ListView contactsList = (ListView) findViewById(R.id.contactList);
        ArrayList<String> contacts = new ArrayList<String>();

        Cursor contactsCursor = getContentResolver()
                .query(ContactsContract.Contacts.CONTENT_URI, null, null, null, null);
        while (contactsCursor.moveToNext()) {

            // получаем каждый контакт
            String contact = contactsCursor.getString(

contactsCursor.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME_PRIMARY));
            // добавляем контакт в список
            contacts.add(contact);
        }

        // создаем адаптер
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_1, contacts);
        // устанавливаем для списка адаптер
        contactsList.setAdapter(adapter);
    }
}
```

Все контакты и сопутствующий функционал хранятся в специальных базах данных SQLite. Но нам не надо напрямую работать с ними. Мы можем воспользоваться объектом класса Cursor. Чтобы его получить, сначала вызывается метод getContentResolver(), который возвращает объект ContentResolver. Затем по цепочке вызывается метод query(). В этот метод передается ряд параметров, первый из которых представляет URI - ресурс, который мы хотим получить. Для обращения к базе данных контактов используется константа ContactsContract.Contacts.CONTENT_URI

Метод contactsCursor.moveToNext() позволяет последовательно перемещаться по записям контактов, считывая по одному контакту через вызов contactsCursor.getString().

Все полученные контакты добавляются в список, который потом выводится в ListView.

Добавление контактов представляет собой запрос на изменение списка контактов, то есть его запись. Поэтому нам надо установить соответствующее разрешение в файле манифеста. Возьмем проект из прошлой темы и добавим в файл манифеста после корневого тега manifest следующую строку

<uses-permission android:name="android.permission.WRITE_CONTACTS" android:maxSdkVersion="21" />

Чтобы вносить изменения в список контактов, следует установить разрешение android.permission.WRITE_CONTACTS

Для добавления контакта добавим новую activity. Назовем ее AddContactActivity. Определим в ее разметке текстовое поле и кнопку для добавления контакта:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="@dimen/activity_horizontal_margin"
    android:orientation="vertical">

    <TextView android:text="Новый контакт:"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <EditText
        android:id="@+id/contactText"
        android:layout_width="match_parent"
        android:layout_height="45dip" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Добавить"
        android:onClick="addContact"/>
</LinearLayout>
```

В коде activity пропишем обработчик addContact с добавлением контакта:

```
package com.example.eugene.sqliteapp;

import android.provider.ContactsContract;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.content.ContentValues;
import android.content.ContentUris;
import android.provider.ContactsContract.RawContacts;
import android.provider.ContactsContract.Data;
import android.provider.ContactsContract.CommonDataKinds.StructuredName;
import android.widget.EditText;
import android.widget.Toast;
import android.net.Uri;
```

```
public class AddContactActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_contact);
    }

    public void addContact(View view) {

        ContentValues contactValues = new ContentValues();
        EditText contactText = (EditText) findViewById(R.id.contactText);
        String newContact = contactText.getText().toString();
        contactValues.put(RawContacts.ACCOUNT_NAME, newContact);
        contactValues.put(RawContacts.ACCOUNT_TYPE, newContact);
        Uri newUri = getContentResolver().insert(RawContacts.CONTENT_URI,
contactValues);
        long rawContactsId = ContentUris.parseId(newUri);
        contactValues.clear();
        contactValues.put(Data.RAW_CONTACT_ID, rawContactsId);
        contactValues.put(Data.MIMETYPE, StructuredName.CONTENT_ITEM_TYPE);
        contactValues.put(StructuredName.DISPLAY_NAME, newContact);
        getContentResolver().insert(Data.CONTENT_URI, contactValues);
        Toast.makeText(this, newContact + " добавлен в список контактов",
Toast.LENGTH_LONG).show();
    }
}
```

Весь код добавления находится в обработчике нажатия кнопки addContact. В Android контакты распределяются по трем таблицам: contacts, raw contacts и data. И нам надо добавить новый контакт в две последне таблицы. В таблицу contact в силу настроек мы добавить не можем, но это и не нужно.

Данные контакта представляют объект ContentValues, который состоит из ключей и их значений, то есть объект словаря. После его создания происходит добавление в него пары элементов:

```
contactValues.put(RawContacts.ACCOUNT_NAME, newContact);
contactValues.put(RawContacts.ACCOUNT_TYPE, newContact);
```

Здесь устанавливается название и тип контакта. В качестве ключей выставляются значения RawContacts.ACCOUNT_NAME и RawContacts.ACCOUNT_TYPE, а в качестве их значения - текст из текстового поля.

Далее этот объект добавляется в таблицу RawContacts с помощью метода insert():

```
Uri newUri = getContentResolver().insert(RawContacts.CONTENT_URI, contactValues);
```

Метод insert() возвращает URI - ссылку на добавленный объект в таблице, у которого мы можем получить id. Затем после очистки мы подготавливаем объект для доабвления в таблицу Data, вновь наполняя его данными:

```
contactValues.put(Data.RAW_CONTACT_ID, rawContactsId);
contactValues.put(Data.MIMETYPE, StructuredName.CONTENT_ITEM_TYPE);
```

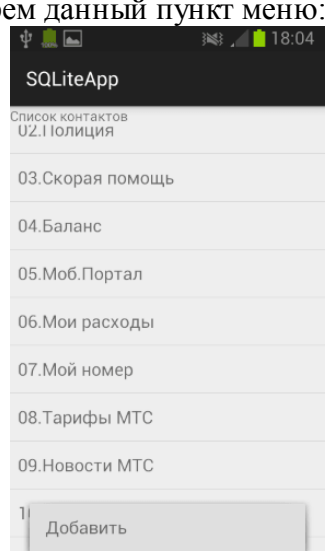
```
contactValues.put(StructuredName.DISPLAY_NAME, newContact);
```
И опять добавление производит метод insert():

```
getContentResolver().insert(Data.CONTENT_URI, contactValues);
```
Чтобы обратиться к этой activity из главной мы можем на главной создать соответствующий пункт меню, по выбору которого будет перекидывать на страницу добавления нового контакта. Например, пункт меню в файле ресурса меню главной activity:

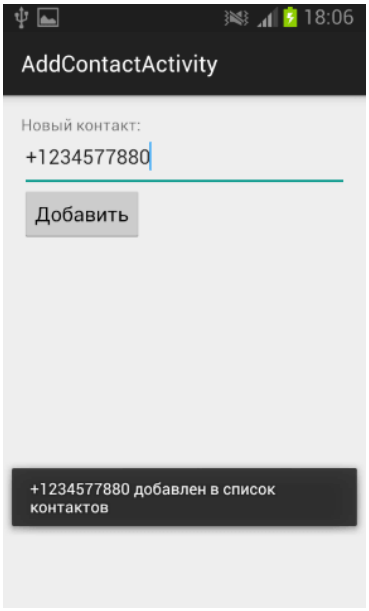
```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools" tools:context=".MainActivity">
    <item
        android:id="@+id/action_settings"
        android:title="Добавить"
        android:orderInCategory="100"
        app:showAsAction="never" />
</menu>
```
И обработка этого пункта в коде главной activity:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    if (id == R.id.action_settings) {
        Intent intent = new Intent(this, AddContactActivity.class);
        startActivity(intent);
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```
Запустим приложение и выберем данный пункт меню:



Перейдем на экран добавления и добавим новый контакт:



9.3.Использование созданных контент провайдеров

Для доступа к данным какого-либо контент-провайдера используется объект класса ContentResolver, который можно получить с помощью метода getContentResolver контекста приложения для связи с поставщиком в качестве клиента:

```
ContentResolver cr = getApplicationContext().getContentResolver();
```

Объект ContentResolver взаимодействует с объектом контент-провайдера, отправляя ему запросы клиента. Контент-провайдер обрабатывает запросы и возвращает результаты обработки.

Контент-провайдеры представляют свои данные потребителям в виде одной или нескольких таблиц подобно таблицам реляционных БД. Каждая строка при этом является отдельным «объектом» со свойствами, указанными в соответствующих именованных полях. Как правило, каждая строка имеет уникальный целочисленный индекс и именем «_id», который служит для однозначной идентификации требуемого объекта.

Контент-провайдеры, обычно предоставляют минимум два URI для работы с данными: один для запросов, требующих все данные сразу, а другой – для обращения к конкретной «строке». В последнем случае в конце URI добавляется / (который совпадает с индексом «_id»).

Запросы на получение данных похожи на запросы к БД, при этом используется метод query объекта ContentResolver. Ответ также приходит в виде курсора, «нацеленного» на результирующий набор данных (выбранные строки таблицы):

```
ContentResolver cr = getContentResolver();
// получить данные всех контактов
Cursor c = cr.query(ContactsContract.Contacts.CONTENT_URI, null, null,
null, null);
// получить все строки, где третье поле имеет конкретное
// значение и отсортировать по пятому полю
String where = KEY_COL3 + "=" + requiredValue;
String order = KEY_COL5;
Cursor someRows = cr
.query(MyProvider.CONTENT_URI, null, where, null, order);
```

Изменение данных:

```
Uri myRowUri = cr.insert(SampleProvider.CONTENT_URI, newRow);

// Массовая вставка:
ContentValues[] valueArray = new ContentValues[5];

// здесь заполняем массив

// делаем вставку
int count = cr.bulkInsert(MyProvider.CONTENT_URI, valueArray);
```

При вставке одного элемента метод insert возвращает URI вставленного элемента, а при массовой вставке возвращается количество вставленных элементов.

Пример удаления:

```
ContentResolver cr = getContentResolver();
// удаление конкретной строки
cr.delete(myRowUri, null, null);
// удаление нескольких строк
String where = "_id < 5";
cr.delete(MyProvider.CONTENT_URI, where, null);
```

Пример изменения:

```
ContentValues newValues = new ContentValues();
newValues.put(COLUMN_NAME, newValue);
String where = "_id < 5";
getContentResolver().update(MyProvider.CONTENT_URI, newValues,
where, null);
```

Контрольные вопросы:

1. Опишите понятие контент-провайдера.
2. Опишите создание контент провайдера.
3. Как используется созданные контент провайдеры ?
4. Какие методы используются для работы с контент провайдерами ?