Лабораторная работа №4

РАБОТА С АДРЕСНОЙ КНИГОЙ

**Цель работы**: получить навыки работы с адресной книгой мобильных устройств Android.

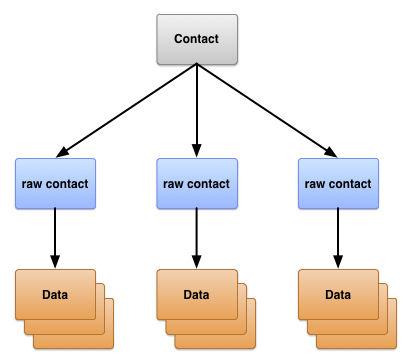
**ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Базовые операции по работе с адресной книгой (получение списка контактов, добавление контакта) были коротко описаны в лабораторной работе 2. Рассмотрим подробнее различные возможности работы с адресной книгой.

**Поставщик контактов**

*Поставщик контактов* представляет собой компонент Android, управляющий центральным репозиторием устройства, в котором хранятся пользовательские данные. Это источник данных, которые отображаются в приложении «Контакты» на мобильном устройстве.

Все контакты Android хранятся в базе данных SQLite. Поставщик контактов представляет собой поставщик контента Android. Он содержит в себе три типа данных о пользователе, каждый из которых указан в отдельной таблице, предоставляемой поставщиком, как показано на рисунке:



В качестве имен этих трех таблиц обычно используются названия соответствующих классов-контрактов. Эти классы определяют константы для URI контента, названий столбцов и значений в столбцах этих таблиц.

1. Таблица ContactsContract.Contacts. Строки в этой таблице содержат данные о разных пользователях, полученные путем агрегации строк необработанных контактов.
2. Таблица ContactsContract.RawContacts. Строки в этой таблице содержат сводные данные о пользователе, относящиеся к пользовательскому аккаунту и его типу.
3. Таблица ContactsContract.Data. Строки в этой таблице содержат сведения о необработанных контактах, такие как адреса эл. почты или номера телефонов.

Другие таблицы, представленные классами-контрактами в ContactsContract, представляют собой вспомогательные таблицы, которые поставщик контактов использует для управления своими операциями.

**Данные контактов**

*Необработанный контакт* представляет собой данные о пользователе, поступающие из одного аккаунта определенного типа. Поскольку в качестве источника данных о пользователе в поставщике контактов может выступать сразу несколько онлайн-служб, для одного и того же пользователя в поставщике контактов может существовать несколько необработанных контактов. Это также позволяет пользователю объединять пользовательские данные из нескольких аккаунтов одного и того же типа. Также у одного необработанного контакта может существовать несколько экземпляров одного и того же типа данных (например, адресов эл. почты или номеров телефонов).

Большая часть данных необработанного контакта не хранится в таблице ContactsContract.RawContacts. Вместо этого они хранятся в одной или нескольких строках в таблице ContactsContract.Data. В каждой строке данных имеется столбец Data.RAW\_CONTACT\_ID, в котором содержится значение android.provider.BaseColumns#\_ID RawContacts.\_ID его родительской строки ContactsContract.RawContacts.

Строки со сведениями об имени контакта, номере телефона, адресе эл. почты, почтовом адресе, фото и веб-сайте хранятся в таблице ContactsContract.Data. Для упрощения управления этими данными в таблице ContactsContract.Data предусмотрены столбцы с различными типами имен:

1. Описательные имена столбцов (например, RAW\_CONTACT\_ID, MIMETYPE, IS\_PRIMARY).
2. Универсальные имена столбцов (существует 15 общедоступных столбцов с универсальными именами (DATA1–DATA15) и четыре дополнительных столбца (SYNC1–SYNC4), которые используются только адаптерами синхронизации).
3. Имена столбцов по типам строк (для упрощения работы со столбцами определенного типа строк в поставщике контактов предусмотрены константы для имен столбцов по типам строк, которые определены в подклассах класса ContactsContract.CommonDataKinds).

Рассмотрим наиболее часто используемые классы имен столбцов по типам строк:

|  |  |  |
| --- | --- | --- |
| Подклассы класса  ContactsContract.CommonDataKinds | Тип данных, к которым предоставляется доступ | Примечания |
| .StructuredName | Данные об имени необработанного контакта, связанного с этой строкой данных. | У необработанного контакта имеется только одна строка такого типа. |
| .Photo | Основная фотография необработанного контакта, связанного с этой строкой данных. | У необработанного контакта имеется только одна строка такого типа. |
| .Email | Адрес эл. почты необработанного контакта, связанного с этой строкой данных. | У необработанного контакта может быть несколько адресов эл. почты. |
| .StructuredPostal | Почтовый адрес необработанного контакта, связанного с этой строкой данных. | У необработанного контакта может быть несколько почтовых адресов. |
| .GroupMembership | Идентификатор, с помощью которого необработанный контакт связан с одной из групп в поставщике контактов. | Группы представляют собой необязательный компонент для типа аккаунта и имени аккаунта. |

**Профиль пользователя**

В таблице ContactsContract.Contacts имеется всего одна строка, содержащая данные профиля для устройства пользователя. Константы для доступа к профилю пользователя представлены в классе ContactsContract.Profile. Для доступа к профилю пользователя требуются особые разрешения:

- android.Manifest.permission#READ\_PROFILE – на чтение данных;

- android.Manifest.permission#WRITE\_PROFILE – на запись.

Чтобы получить строку контакта с профилем пользователя, необходимо вызвать метод ContentResolver.query(), задать для URI контента значение CONTENT\_URI без указания критериев выборки. Этот URI контента также можно использовать в качестве основного URI для получения необработанных контактов или данных профиля:

mProjection = new String[]

{

Profile.\_ID,

Profile.DISPLAY\_NAME\_PRIMARY,

Profile.LOOKUP\_KEY,

Profile.PHOTO\_THUMBNAIL\_URI

};

mProfileCursor = getContentResolver().query( Profile.CONTENT\_URI, Projection, null, null, null);

**Просмотр адресной книги**

Перейдем к созданию простейшей телефонной книги. Напомним, что для доступа к контактам нам надо установить соответствующие разрешения в файле манифеста приложения:

<uses-permission android:name="android.permission.READ\_CONTACTS" />

Подготовим некоторую разметку интерфейса для отображения списка контактов, например, следующего вида:

\_\_\_\_\_\_\_\_\_\_\_\_\_<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>

Контакты будут отображаться в виде списка в элементе ListView. Перейдем в класс activity, где будем получать контакты и передавать их в интерфейс.

При работе с таблицами контактов 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.

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);

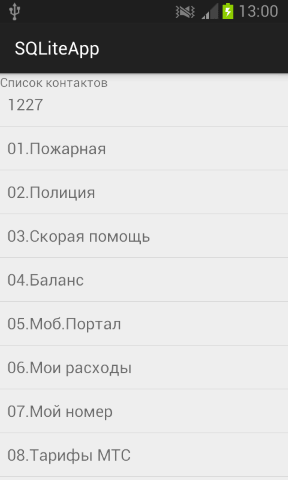
// устанавливаем для списка адаптер

contactsList.setAdapter(adapter);

}

}

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. При запуске нам отобразится список контактов:



**Добавление контакта**

Напомним, что для записи нового контакта в телефонную книгу необходимо добавить в файл манифеста разрешение:

<uses-permission android:name="android.permission.WRITE\_CONTACTS" />

Для добавления контакта создадим 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 Активности пропишем обработчик нажатия кнопки addContact с добавлением контакта. Нам надо добавить новый контакт в две таблицы. Вначале добавляем необходимые данные в таблицу raw contacts, затем – в data, как мы делали в лабораторной работе 2.

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();

}

Чтобы обратиться к Активности добавления контакта из главной (со списком всех контактов, открываемым при запуске приложения), мы можем на ней создать соответствующий пункт меню, по выбору которого будет происходить переход на страницу добавления нового контакта. Например, пункт меню в файле xml разметки главной activity можно прописать так:

<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>

Добавим обработчик для описанного пункта меню в коде главной активности:

@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);

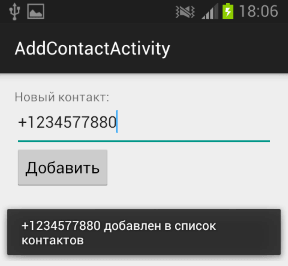
return true;

}

return super.onOptionsItemSelected(item);

}

Теперь вместе со списком контактов на стартовом экране будет отображаться ссылка на добавление новой записи. При переходе по ней увидим экран следующего вида:



**Удаление контакта**

Для удаления некоторого контакта по его имени и телефонному номеру можно использовать следующую запись:

public static void deleteContact(Context ctx, String phone, String name) {

Uri contactUri = Uri.withAppendedPath(PhoneLookup.CONTENT\_FILTER\_URI, Uri.encode(phone));

Cursor cur = ctx.getContentResolver().query(contactUri, null, null, null, null);

if (cur.moveToFirst()) {

do {

if (cur.getString(cur.getColumnIndex(PhoneLookup.DISPLAY\_NAME)) .equalsIgnoreCase(name)) {

String lookupKey = cur.getString( cur.getColumnIndex(

\_ ContactsContract.Contacts.LOOKUP\_KEY));

Uri uri = Uri.withAppendedPath(ContactsContract.Contacts

\_ .CONTENT\_LOOKUP\_URI, lookupKey);

ctx.getContentResolver().delete(uri, null, null);

}

} while (cur.moveToNext());

}

cur.close();

}

Тут мы ищем запись в телефонной книге по номеру телефона, после чего сравниваем имя контакта с удаляемым для каждой найденной записи (без учета регистра), при совпадении удаляем контакт с помощью метода delete. В качестве параметра метод принимает значение URI для контакта.

**Редактирование контакта с помощью намерений**

С помощью отправки намерения в приложение для работы с контактами, которое имеется на устройстве, можно получить доступ к поставщику контактов. Такой тип доступа позволяет пользователю выполнять следующие действия:

- выбор контактов из списка и их возврат в приложение для дальнейшей работы;

- изменение данных существующего контакта;

- вставка нового необработанного контакта для любых других аккаунтов;

- удаление контакта или его данных.

При использовании намерений для работы с контактами нет необходимости создавать собственный пользовательский интерфейс или код для доступа к поставщику. Также не нужно запрашивать разрешение на чтение или запись в поставщик. Приложение для работы с контактами, имеющееся на устройстве, может делегировать вам разрешение на чтение контакта, и поскольку вы вносите изменения в поставщик через другое приложение, вам не нужно разрешение на запись.

Пример использования намерения для редактирования контакта показан ниже:

// получаем URI контакта

mContact.str = phones.getString(phones.getColumnIndex(Phone.\_ID));

Uri mUri = ContentUris.withAppendedId( ContactsContract.Contacts.CONTENT\_URI,

Long.parseLong(mContact.str));

// создаем намерение редактирования данных

Intent intent = new Intent (Intent.ACTION\_EDIT);

intent.setDataAndType(mUri, Contacts.CONTENT\_ITEM\_TYPE);

intent.putExtra("finishActivityOnSaveCompleted", true);

intent.addFlag(intent.FLAG\_ACTIVITY\_CLEAR\_TOP);

mActivity.startActivityForResult(intent, EDIT\_CONTACT\_RESULT);

**Вопросы для самопроверки**

1. Что такое поставщик контакта? Для чего он нужен?
2. Какова структура таблиц поставщика контактов? В чем их назначение?
3. Что такое необработанный контакт?
4. Какие имена столбцов существуют в таблице ContactsContract.Data?
5. Какие классы имен столбцов по типам строк вам известны? Что вы о них знаете?
6. Какие разрешения могут понадобиться для работы с профилем пользователя?
7. Какие действия позволяет реализовать отправка намерений к поставщику контактов?

**Дополнительные материалы**

1. Поставщики контактов – [Электронный ресурс] – <https://developer.android.com/guide/topics/providers/contacts-provider.html> (дата обращения 25.02.2017).
2. Дейтел П. «Android для программистов: создаём приложения» – Питер, 2012 – 560 с.
3. Android Get Phone Contacts details – [Электронный ресурс] – <http://www.androidhub4you.com/2013/06/get-phone-contacts-details-in-android_6.html> (дата обращения 25.02.2017).
4. Working With Android Contacts – [Электронный ресурс] – <http://www.higherpass.com/android/tutorials/working-with-android-contacts/> (дата обращения 25.02.2017).
5. Android: Get Contact Details (ID, Name, Phone, Photo) – [Электронный ресурс] – <https://tausiq.wordpress.com/2012/08/23/android-get-contact-details-id-name-phone-photo/> (дата обращения 25.02.2017).

**Задание**

Создать приложение, которое добавляет, редактирует и удаляет контакт из адресной книги. При добавлении контакта необходимо указать фамилию, имя, номер мобильного телефона, E-mail, место работы, рабочий телефон, а также позволить пользователю выбрать для контакта изображение и рингтон.

**Варианты заданий**

1. Позволить пользователю группировать контакты по группам (например, Работа, Друзья, Семья и т.п.).
2. При создании контакта с номером телефона, уже имеющимся в телефонной книге, предлагать пользователю объединить контакты.
3. Позволить пользователю установить рингтон сразу нескольким выбранным пользователям.
4. Реализовать поиск контакта по месту работы.
5. Позволить пользователю установить изображение по умолчанию для новых контактов (если не выбрано конкретное изображение).
6. Позволить пользователю выбрать рингтон по умолчанию для новых контактов (если не задан рингтон при создании контакта).
7. Реализовать поиск контактов с пустыми полями (например, местом работы или рабочим телефоном).
8. Реализовать поиск контактов без изображения.
9. Реализовать поиск контактов с дублирующейся информацией (например, E-mail или рабочий телефон).
10. Позволить пользователю добавлять день рождения к контакту.