Пример создания ContentProvider для работы с базой SQLite часть 1



Согласно официальным руководствам от команды Android SDK, ContentProvider необходим лишь в случаях, когда вашему приложению необходимо предоставлять свои данные для других сторонних приложений, или, к примеру, реализовать поиск среди данных вашего приложения с использованием search suggestions (подсказки при вводе слов для поиска).

Тем не менее, для отображения массивов данных, приходится часто использовать CursorAdapter совместно с CursorLoader. Данная связка облегчает работу с различными списками и таблицами, потому что автоматически наблюдает за изменениями в базе sqlite и обновляет данные, когда это необходимо.

В этой статье я расскажу, как написать удобный ContentProvider для взаимодействия с базой sqlite и что для этого необходимо. Первым делом нужно определиться со структурой базы данных. В качестве примера, возьмем две таблицы: ученики и учебные классы. Таблицы будут иметь следующие столбцы:

Students

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| \_id | first\_name | second\_name | average\_score | fk\_class\_id |

Classes

|  |  |  |
| --- | --- | --- |
| \_id | class\_number | class\_letter |

fk\_class\_id будет являться внешним ключом к классу Classes и хранить значение Classes.\_id

Теперь, когда мы знаем схему нашей базы sqlite, можно создать так называемый Contract Class — класс, который будет хранить все имена столбцов, проекции для запросов и остальные стандартные (для наших таблиц) значения. Сам класс выглядит так:

public final class ContractClass {

public static final String AUTHORITY = "org.nerdgrl.examples.contentproviderexample.provider.ContractClass";

private ContractClass() {}

public static final class Students implements BaseColumns {...}

public static final class Classes implements BaseColumns {...}

}

где AUTHORITY — это уникальное имя приложения, которое будут использовать другие приложения, если вы решите предоставить для них свои внутренние данные из БД. Чтобы не было конфликтов с другими провайдерами, рекомендуют использовать доменное имя в качестве префикса.

Класс BaseColumns представляет собой список необходимых для работы с базой sqlite столбцов, а именно содержит константы \_ID и \_COUNT. Хотя ContentProvider не требует чтобы первичный ключ назывался \_ID, для работы с ListView, к примеру, ключ \_ID необходим. Поэтому проще использовать класс BaseColumns, где оба эти столбца уже есть и вы о них не забудете.

Теперь разберемся с классом Students:

public static final class Students implements BaseColumns {

private Students() {}

public static final String TABLE\_NAME ="students";

private static final String SCHEME = "content://";

private static final String PATH\_STUDENTS = "/students";

private static final String PATH\_STUDENTS\_ID = "/students/";

public static final int STUDENTS\_ID\_PATH\_POSITION = 1;

public static final Uri CONTENT\_URI = Uri.parse(SCHEME + AUTHORITY + PATH\_STUDENTS);

public static final Uri CONTENT\_ID\_URI\_BASE = Uri.parse(SCHEME + AUTHORITY + PATH\_STUDENTS\_ID);

public static final String CONTENT\_TYPE = "vnd.android.cursor.dir/vnd.google.students";

public static final String CONTENT\_ITEM\_TYPE = "vnd.android.cursor.item/vnd.google.students";

public static final String DEFAULT\_SORT\_ORDER = "second\_name ASC";

public static final String COLUMN\_NAME\_FIRST\_NAME = "first\_name";

public static final String COLUMN\_NAME\_SECOND\_NAME = "second\_name";

public static final String COLUMN\_NAME\_AVERAGE\_SCORE = "average\_score";

public static final String COLUMN\_NAME\_FK\_CLASS\_ID = "fk\_class\_id";

public static final String[] DEFAULT\_PROJECTION = new String[] {

ContractClass.Students.\_ID,

ContractClass.Students.COLUMN\_NAME\_FIRST\_NAME,

ContractClass.Students.COLUMN\_NAME\_SECOND\_NAME,

ContractClass.Students.COLUMN\_NAME\_AVERAGE\_SCORE,

ContractClass.Students.COLUMN\_NAME\_FK\_CLASS\_ID

};

}

Мы видим приватный конструктор, который запрещает неправильное использование данного класса, ведь нам нужны только константы.

TABLE\_NAME — имя таблицы в базе sqlite.

SCHEME — схема данных, которую указывают, например, в <intent-filter><data scheme=…></intent-filter> . Название чувствительно к регистру, поэтому рекомендуется использовать только нижний.

PATH\_STUDENTS и PATH\_STUDENTS\_ID — используются для составления Content Uri. ContentProvider рассматривает Content Uri как путь к нужной таблице, и/или к конкретной записи.  
Content Uri для таблицы Students будет выглядеть следующим образом:  
content://org.nerdgrl.examples.contentproviderexample.provider.ContractClass/students (для всей директории)  
content://org.nerdgrl.examples.contentproviderexample.provider.ContractClass/students/3 (для записи под номером 3)

CONTENT\_TYPE и CONTENT\_ITEM\_TYPE — это MIME типы для хранящихся в нашей БД данных. Все пользовательские типы данных должны начинаться с vnd.android.cursor.dir/vnd для директорий, и с vnd.android.cursor.item/vnd для отдельных записей.

Далее мы прописываем название столбца для сортировки по нему (значение по умолчанию), и все имена столбцов таблицы.

DEFAULT\_PROJECTION — это набор столбцов, которые будут включены в результаты запроса.

Код класса Classes:

public static final class Classes implements BaseColumns {

private Classes() {}

public static final String TABLE\_NAME ="classes";

private static final String SCHEME = "content://";

private static final String PATH\_CLASSES = "/classes";

private static final String PATH\_CLASSES\_ID = "/classes/";

public static final int CLASSES\_ID\_PATH\_POSITION = 1;

public static final Uri CONTENT\_URI = Uri.parse(SCHEME + AUTHORITY + PATH\_CLASSES);

public static final Uri CONTENT\_ID\_URI\_BASE = Uri.parse(SCHEME + AUTHORITY + PATH\_CLASSES\_ID);

public static final String CONTENT\_TYPE = "vnd.android.cursor.dir/vnd.org.nerdgrl.classes";

public static final String CONTENT\_ITEM\_TYPE = "vnd.android.cursor.item/vnd.org.nerdgrl.classes";

public static final String DEFAULT\_SORT\_ORDER = "class\_number ASC";

public static final String COLUMN\_NAME\_CLASS\_NUMBER = "class\_number";

public static final String COLUMN\_NAME\_CLASS\_LETTER = "class\_letter";

public static final String[] DEFAULT\_PROJECTION = new String[] {

ContractClass.Classes.\_ID,

ContractClass.Classes.COLUMN\_NAME\_CLASS\_NUMBER,

ContractClass.Classes.COLUMN\_NAME\_CLASS\_LETTER

};

}

В [следующей](http://www.nerdgrl.org/ru/programming/contentprovider-2/)части я расскажу из чего состоит ContentProvider, и как его использовать для отображения данных.