Лекция 5

Мобильная разработка

Пирская Любовь Владимировна,
к.т.н., доцент кафедры МОП ЭВМ
lpirskaya@sfedu.ru

База данных

SQlite

- Встраиваемая
- Реляционная
- Open source
- База данных - один файл
- Быстрая

БД в Android

- android.database.sqlite
- Schema и Contract
- SQLiteDatabase
- SQLiteOpenHelper
- _ID

Database Contract

```
public final class FeedReaderContract {
    // To prevent someone from accidentally instantiating the contract class,
    // give it an empty constructor.
    public FeedReaderContract() {}

    /* Inner class that defines the table contents */
    public static abstract class FeedEntry implements BaseColumns {
        public static final String TABLE_NAME = "entry";
        public static final String COLUMN_NAME_ENTRY_ID = "entryid";
        public static final String COLUMN_NAME_TITLE = "title";
        public static final String COLUMN_NAME_SUBTITLE = "subtitle";
        ...
    }
}
```

Use Contract

```
private static final String TEXT_TYPE = " TEXT";
private static final String COMMA_SEP = ",";
private static final String SQL_CREATE_ENTRIES =
    "CREATE TABLE " + FeedEntry.TABLE_NAME + " (" +
    FeedEntry._ID + " INTEGER PRIMARY KEY," +
    FeedEntry.COLUMN_NAME_ENTRY_ID + TEXT_TYPE + COMMA_SEP +
    FeedEntry.COLUMN_NAME_TITLE + TEXT_TYPE + COMMA_SEP +
    ... // Any other options for the CREATE command
    " )";

private static final String SQL_DELETE_ENTRIES =
    "DROP TABLE IF EXISTS " + FeedEntry.TABLE_NAME;
```

SQLiteOpenHelper

```
public class FeedReaderDbHelper extends SQLiteOpenHelper {
    // If you change the database schema, you must increment the database version.
    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "FeedReader.db";

    public FeedReaderDbHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(SQL_CREATE_ENTRIES);
    }
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // This database is only a cache for online data, so its upgrade policy is
        // to simply to discard the data and start over
        db.execSQL(SQL_DELETE_ENTRIES);
        onCreate(db);
    }
    public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        onUpgrade(db, oldVersion, newVersion);
    }
}
```

Insert

```
// Gets the data repository in write mode
SQLiteDatabase db = mDbHelper.getWritableDatabase();

// Create a new map of values, where column names are the keys
ContentValues values = new ContentValues();
values.put(FeedEntry.COLUMN_NAME_ENTRY_ID, id);
values.put(FeedEntry.COLUMN_NAME_TITLE, title);
values.put(FeedEntry.COLUMN_NAME_CONTENT, content);

// Insert the new row, returning the primary key value of the new row
long newRowId;
newRowId = db.insert(
         FeedEntry.TABLE_NAME,
         FeedEntry.COLUMN_NAME_NULLABLE,
         values);
```

Query

```
SQLiteDatabase db = mDbHelper.getReadableDatabase();

// Define a projection that specifies which columns from the database
// you will actually use after this query.
String[] projection = {
    FeedEntry._ID,
    FeedEntry.COLUMN_NAME_TITLE,
    FeedEntry.COLUMN_NAME_UPDATED,
    ...
    };

// How you want the results sorted in the resulting Cursor
String sortOrder =
    FeedEntry.COLUMN_NAME_UPDATED + " DESC";

Cursor c = db.query(
    FeedEntry.TABLE_NAME,  // The table to query
    projection,                                 // The columns to return
    selection,                                  // The columns for the WHERE clause
    selectionArgs,                              // The values for the WHERE clause
    null,                                       // don't group the rows
    null,                                       // don't filter by row groups
    sortOrder                                   // The sort order
    );
```

Update

```
SQLiteDatabase db = mDbHelper.getReadableDatabase();

// New value for one column
ContentValues values = new ContentValues();
values.put(FeedEntry.COLUMN_NAME_TITLE, title);

// Which row to update, based on the ID
String selection = FeedEntry.COLUMN_NAME_ENTRY_ID + " LIKE ?";
String[] selectionArgs = { String.valueOf(rowId) };

int count = db.update(
    FeedReaderDbHelper.FeedEntry.TABLE_NAME,
    values,
    selection,
    selectionArgs);
```

Cursors

- Virtual Table
- Набор строк
- Структура с данными
- moveToFirst()
- Названия столбцов
- Тип столбцов
- close()

Loaders

- Основная задача: асинхронная загрузка данных в активити или фрагмент
- Следят за источником данных
- Основные классы:
  - LoaderManager
  - LoaderCallbacks
  - Loader
  - CursorLoader
  - AsyncTaskLoader

LoaderCallbacks

```
public class SampleActivity extends Activity implements
LoaderManager.LoaderCallbacks<D> {

  public Loader<D> onCreateLoader(int id, Bundle args)
{ ... }

  public void onLoadFinished(Loader<D> loader, D
data) { ... }

  public void onLoaderReset(Loader<D> loader) { ... }

  /* ... */
}
```

Simple Loader Example

```
public class SampleListActivity extends ListActivity implements
LoaderManager.LoaderCallbacks<Cursor> {
  private static final String[] PROJECTION = new String[] { "_id", "text_column"
};
  private static final int LOADER_ID = 1;
  private LoaderManager.LoaderCallbacks<Cursor> mCallbacks;
  private SimpleCursorAdapter mAdapter;

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    String[] dataColumns = { "text_column" };
    int[] viewIDs = { R.id.text_view };

    mAdapter = new SimpleCursorAdapter(this, R.layout.list_item,
        null, dataColumns, viewIDs, 0);

    setListAdapter(mAdapter);
    mCallbacks = this;
    LoaderManager lm = getLoaderManager();
    lm.initLoader(LOADER_ID, null, mCallbacks);
  }
```

Simple Loader Example

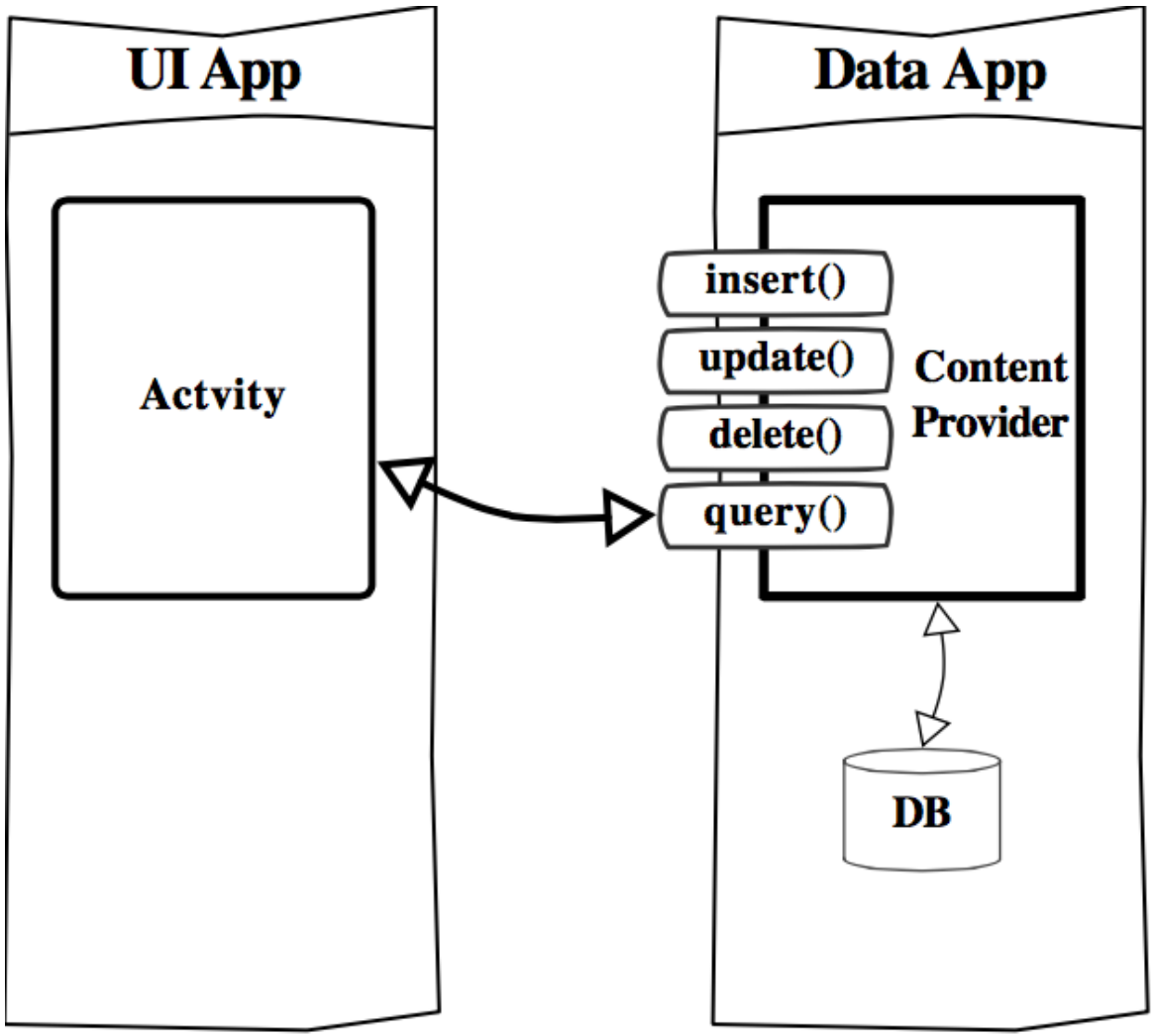
```
@Override
  public Loader<Cursor> onCreateLoader(int id, Bundle args) {
       return new CursorLoader(SampleListActivity.this,
CONTENT_URI,
        PROJECTION, null, null, null);
  }

  @Override
  public void onLoadFinished(Loader<Cursor> loader, Cursor
cursor) {
       switch (loader.getId()) {
          case LOADER_ID:
             mAdapter.swapCursor(cursor);
             break;
        }
  }

  @Override
  public void onLoaderReset(Loader<Cursor> loader) {
    mAdapter.swapCursor(null);
  }
```

ContentProvider

ContentProvider



ContentProvider

- Инкапсуляция данных
- Механизм доступа к данным
- Интерфейс – CRUD
- URI
- Batch Operations
- Permissions
- Представление данных в виде таблицы
- ContentResolver.query
- Системные провайдеры

URI

- Идентификация данных
- Идентификация провайдера: authority
- Путь до таблицы: path
- content://user_dictionary/words
- Uri singleUri = ContentUris.withAppendedId(UserDictionary.Words.CONTENT_URI, 4);

Реализация ContentProvider

- На примере списка городов
- Создать класс-наследник ContentProvider-а
- Реализовать onCreate для легковесной инициализации (БД, UriMatcher)

```
@Override
public boolean onCreate() {
    Log.d(TAG, "onCreate");

    mDatabaseHelper = new DatabaseHelper(getContext());
    mUriMatcher.addURI(CitiesContract.AUTHORITY, "cities", CITIES);
    mUriMatcher.addURI(CitiesContract.AUTHORITY, "capitals", CAPITALS);
    mUriMatcher.addURI(CitiesContract.AUTHORITY, "cities/#", CITY);
    mUriMatcher.addURI(CitiesContract.AUTHORITY, "image", IMAGE);

    return true;
}
```

Реализация ContentProvider

- Контракт

```
public static final String AUTHORITY = "ru.ilapin.recyclerviewandcontentprovider.provider";
public static final Uri AUTHORITY_URI = Uri.parse("content://" + AUTHORITY);

private CitiesContract() {}

public static final class Cities {

    private Cities() {}

    public static final String CONTENT_ITEM_TYPE = "vnd.android.cursor.item/city";
    public static final String CONTENT_TYPE = "vnd.android.cursor.dir/city";

    public static final Uri CONTENT_URI = Uri.withAppendedPath(AUTHORITY_URI, "cities");
    public static final Uri CAPITALS_CONTENT_URI = Uri.withAppendedPath(AUTHORITY_URI, "capitals");
    public static final Uri IMAGE_URI = Uri.withAppendedPath(AUTHORITY_URI, "image");

    public static final String _ID = "_ID";
    public static final String NAME = "name";
    public static final String CAPITAL = "capital";
}
```

Реализация ContentProvider

- Реализовать query для получения данных

```
@Override
public Cursor query(final Uri uri, final String[] projection, final String selection, final String[] selectionArgs, final String sortOrder) {
    Log.d(TAG, "query: " + uri);

    switch (mUriMatcher.match(uri)) {
        case CITIES: {
            Log.d(TAG, "cities URI match");
            final Cursor cursor = mDatabaseHelper.getReadableDatabase().query(
                    "City",
                    projection,
                    selection,
                    selectionArgs,
                    null,
                    null,
                    sortOrder
            );

            cursor.setNotificationUri(getContext().getContentResolver(), CitiesContract.Cities.CONTENT_URI);

            return cursor;
        }
```

Реализация ContentProvider

- Реализовать update для обновления данных, принимает Uri данных, данные, условия выборки для обновления, возвращает количество затронутых записей, notifyChange

```
@Override
public int update(final Uri uri, final ContentValues values, final String selection, final String[] selectionArgs) {
    Log.d(TAG, "update: " + uri);

    final int affectedRows = mDatabaseHelper.getWritableDatabase().update("City", values, selection, selectionArgs);

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

    return affectedRows;
}
```

Реализация ContentProvider

- Реализовать getType

```
@Override
public String getType(final Uri uri) {
    Log.d(TAG, "getType: " + uri);

    switch (mUriMatcher.match(uri)) {
        case IMAGE:
            return "image/jpeg";

        case CITY:
            return CitiesContract.Cities.CONTENT_ITEM_TYPE;

        default:
            return CitiesContract.Cities.CONTENT_TYPE;
    }
}
```

Реализация ContentProvider

- Реализовать getType

```
@Override
public String getType(final Uri uri) {
    Log.d(TAG, "getType: " + uri);

    switch (mUriMatcher.match(uri)) {
        case IMAGE:
            return "image/jpeg";

        case CITY:
            return CitiesContract.Cities.CONTENT_ITEM_TYPE;

        default:
            return CitiesContract.Cities.CONTENT_TYPE;
    }
}
```

Реализация ContentProvider

- Объявить провайдер в манифесте

```
<provider
    android:authorities="ru.ilapin.recyclerviewandcontentprovider.provider"
    android:name=".providers.CitiesContentProvider"
    android:icon="@drawable/ic_launcher"
    android:label="@string/provider_label"
    android:readPermission="ru.ilapin.recyclerviewandcontentprovider.READ_CITIES"
    android:exported="true"/>
```