Loaders and ContentProviders

Parallels

Profit from the Cloud™

Зачем нужны? Как использовать?

Что такое ContentProvider?

- ContentProvider -- компонент приложения, позволяющий получать доступ к данным в табличном виде и в виде файлов
- Доступ к ContentProvider-у не ограничен рамками приложения, в котором он реализован
- Доступ к данным осуществляется через Uri

Что такое Loader?

- Loader-ы -- набор классов и интерфейсов, облегчающих организацию асинхронной загрузки данных и сохранение данных при изменении конфигурации
- Доступ к возможностям Loader-ов можно получить из любой Activity или Fragment-а
- CursorLoader отслеживает изменения источника данных

Совместная работа ContentProvider-а и Loader-а

- CursorLoader перезапрашивает данные при правильном использовании notifiChange и setNotificationUri
- Метод setNotificationUri класса Cursor позволяет установить URI данных, изменение которых нужно отслеживать
- В методах insert, update, delete вызываем метод notifyChange класса ContentResolver с URI элемента или элементов, подвергшихся изменению
- При обновлении потомка, обновляются родители
- content://ru.ilapin.recyclerviewandcontentprovider.provider/cities/65 обновит content://ru.ilapin.recyclerviewandcontentprovider.provider/cities
- Работа осуществляется через ContentResolver

Системные ContentProvider-ы на примере ContactProvider-а 1

- Используя информацию из класса-контракта узнать URI данных ContactsContract.Contacts.CONTENT_URI
- Реализовать интерфейс LoaderManager.LoaderCallbacks
- В методе onCreateLoader инстанциировать Loader

```
@Override
public Loader<Cursor> onCreateLoader(int id, Bundle args) {
    Log.d(TAG, "onCreateLoader");
    return new CursorLoader(
            mActivity,
            ContactsContract.Contacts.CONTENT_URI,
            new String[]{
                    ContactsContract.Contacts.DISPLAY_NAME_PRIMARY
            },
            null,
            null,
            ContactsContract.Contacts.DISPLAY_NAME_PRIMARY + " ASC"
    );
}
```

Системные ContentProvider-ы на примере ContactProvider-а 2

- В методе onLoadFinished написать код обрабатывающий получение данных из Loader-а

```
@Override
public void onLoadFinished(Loader<Cursor> cursorLoader, Cursor cursor) {
    Log.d(TAG, "onLoadFinished");
    mContactsAdapter.setCursor(cursor);
}
```

- В методе onLoaderReset написать код обрабатывающий сброс Loader-а

```
@Override
public void onLoaderReset(Loader<Cursor> cursorLoader) {
    Log.d(TAG, "onLoaderReset");
    mContactsAdapter.setCursor(null);
}
```

Системные ContentProvider-ы на примере ContactProvider-а 3

- Вызвать getLoaderManager().initLoader(LOADER_ID, null, this) и передать в него идентификатор Loader-а (int-овое число), Bundle с аргументами (может быть null), и реализацию интерфейса

Реализация ContentProvider-а 1

- 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-а 2

- 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-a 3

- 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-а 4

- Реализовать insert для вставки данных, принимает Uri данных и данные, возвращает Uri сохранённых данных, getContext().getContentResolver().notifyChange. В нашем случае пустой.
- Реализовать delete для удаления данных, принимает Uri данных, условия выборки для удаления, возвращает количество затронутых записей, notifyChange. В нашем случае пустой.

Реализация ContentProvider-a 5

- 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-а 6

- 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-а 7

- Реализовать openAssetFile

```
@Override
public AssetFileDescriptor openAssetFile(final Uri uri, final String mode) throws FileNotFoundException {
    Log.d(TAG, "openFile: " + uri);

    if (mUriMatcher.match(uri) != IMAGE) {
        throw new FileNotFoundException(uri.toString());
    }

    try {
        return getContext().getAssets().openFd(IMAGE_FILENAME);
    } catch (final IOException e) {
        throw new RuntimeException(e);
    }
}
```

Реализация ContentProvider-а 8

- В ContentResolver-е можно регистрировать ContentObserver

```
mActivity.getContentResolver().registerContentObserver(
        CitiesContract.Cities.CONTENT_URI,
        true,
        new ContentObserver(new Handler(Looper.getMainLooper())) {
            @Override
            public void onChange(final boolean selfChange) {
                Toast.makeText(mActivity, getString(R.string.content_updated), Toast.LENGTH_SHORT).show();
            }
        }
);
```

Реализация ContentProvider-а 9

- 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"/>
```

Права доступа и получение файлов через ContentProvider-ы

- Доступ к ContentProvider-у можно получать не только из приложения, в котором он реализован, но и из других приложений android:exported=“true”
- Можно ограничивать доступ, указывая android:readPermission="ru.ilapin.recyclerviewandcontentprovider.READ_CITIES“
- Для получения файла нужно реализовать метод openAssetFile или подобный ему (openFile)
- В методе getType добавить MIME для файла

Реализация Loader-а на примере AsyncTaskLoader-а 1

- Создать класс-наследник AsyncTaskLoader-а
- Реализовать по крайней мере методы loadInBackground и onStartLoading

Реализация Loader-а на примере AsyncTaskLoader-а 2

- В loadInBackground размещается основная «тяжёлая» логика загрузки, метод будет выполняться не в главном потоке

```
@Override
public List<DataEntry> loadInBackground() {
    Log.d(TAG, "loadInBackground");

    try {
        Thread.sleep(10000);
    } catch (final InterruptedException e) {
        throw new RuntimeException(e);
    }

    mDataEntries = DataProvider.sDataEntries;

    return mDataEntries;
}
```

Реализация Loader-а на примере AsyncTaskLoader-а 3

- onStartLoading должен содержать минимальную логику по запуску загрузки (вызов forceLoad)

```
@Override
protected void onStartLoading() {
    Log.d(TAG, "onStartLoading");

    if (mDataEntries != null) {
        deliverResult(mDataEntries);
    } else {
        forceLoad();
    }
}
```

Спасибо за внимание!

- https://github.com/raynor73/ContentConsumer
- https://github.com/raynor73/CustomLoader
- https://github.com/raynor73/RecyclerViewAndContentProvider