PROGRAMMATION DE COMPOSANTS MOBILES (ANDROID)

WIESLAW ZIELONKA

SQLiteOpenHelper

SQLiteOpenHelper

SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version)

getReadableDatabase() : SQLiteDatabase

getWritableDatabase() : SQLiteDatabase

onCreate(SQLiteDatabase base) : void

onUpgrade(SQLiteDatabase base, int oldVersion, int newVersion) : void

SQLiteDatabase

insert(String table, String nullColumnHack, ContentValues values) : long

query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy) : Cursor

rawQuery(String sql, String[] selectionArgs) : Cursor

execSQL (String sql) : void

delete (String table, String whereClause, String[] whereArgs) : int

update (String table, ContentValues values, String whereClause, String[] whereArgs): int

replace(String table, String nullColumnHack, ContentValues initialValues) : long

ContentValues

put(String key, String value) : void

put(String key, Integer value): void

put(String key, Double value): void

.....

getAsString(String key) : String

getAsInteger(String key): Integer

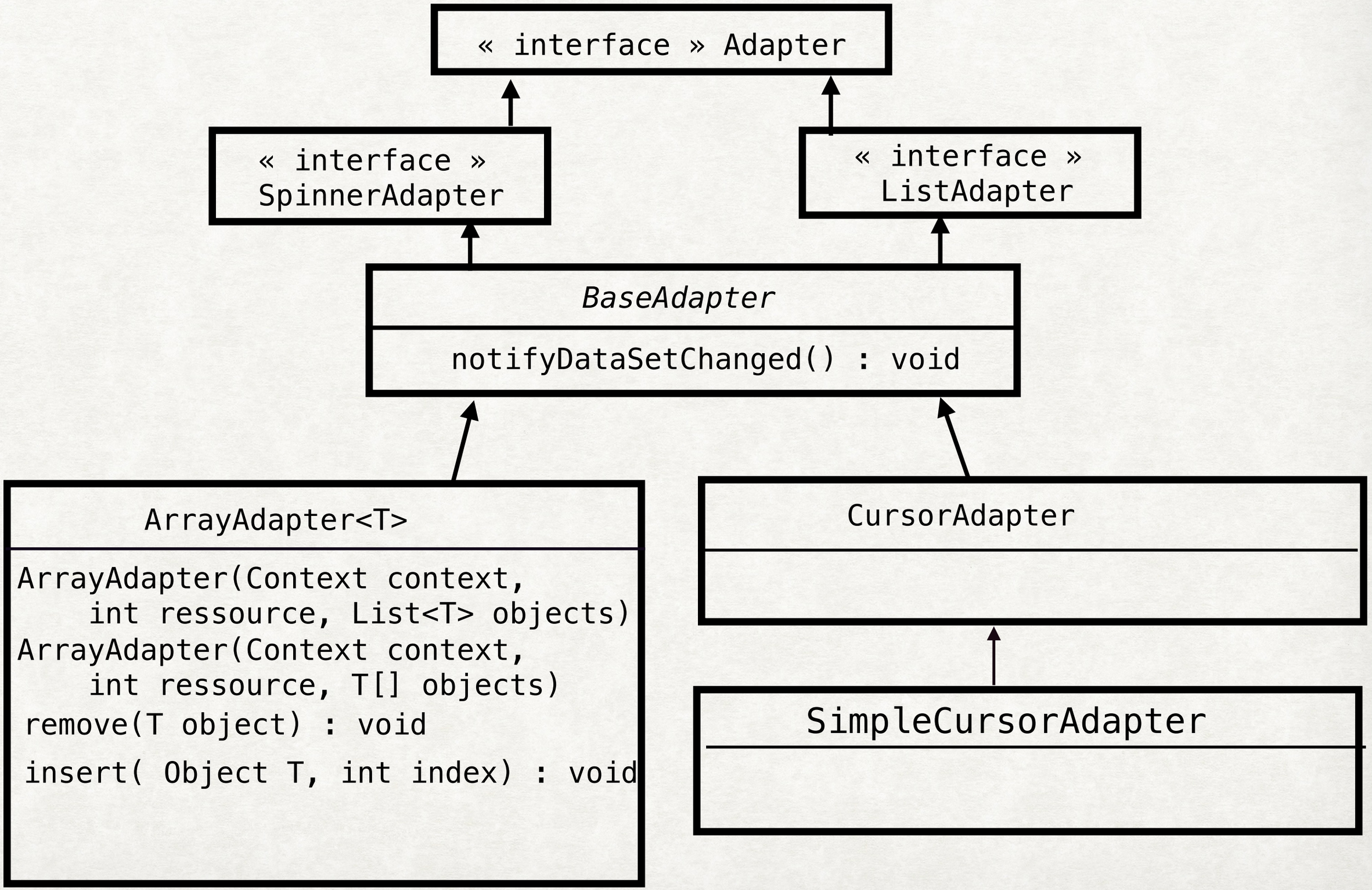
getAsDouble(String key): Double

.....

Comment afficher les résultats de SELECT?

Méthde simples : utiliser ListView et CursorAdpter

les adapters



SimpleCursorAdapter

SimpleCursorAdapter
`SimpleCursorAdapter(Context context, int layout, Cursor c, String[] from, int[] to)` deprecated
`SimpleCursorAdapter(Context context, int layout, Cursor c, String[] from, int[] to, int flags)`

- c -- cursor qui contient le résultat de la requête

- layout -- identifiant d'un layout qui contient un TextView. Utiliser pour spécifier le format de la ligne à afficher. Cela peut être un layout déjà existant ou un layout défini dans l'application.
- from -- le tableau de noms de colonnes à afficher
- to -- les tableau d'identifiants des View du layout, une View par colonne à afficher

SimpleCursorAdapter

```
ListAdapter adapter = new SimpleCursorAdapter(this,
        R.layout.list_item_layout,
        cursor,
        new String[]{MySQLiteHelper.NUMERO, MySQLiteHelper.RUE,
                MySQLiteHelper.ZIP, MySQLiteHelper.VILLE},
        new int[]{R.id.numero, R.id.rue, R.id.zip, R.id.ville}, 0);
```

les noms de colonnes

list_item_layout.xml (tous les attributs sauf id supprimés)

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
       android:orientation="vertical">

    <TextView
        android:id="@+id/numero"
    />

    <TextView
        android:id="@+id/rue"
    />

    <TextView
        android:id="@+id/zip"
    />

    <TextView
        android:id="@+id/ville"
    />


</RelativeLayout>
```

SimpleCursorAdapter

```
ListAdapter adapter = new SimpleCursorAdapter(this,
        R.layout.list_item_layout,
        cursor,
        new String[]{MySQLiteHelper.NUMERO, MySQLiteHelper.RUE,
                MySQLiteHelper.ZIP, MySQLiteHelper.VILLE},
        new int[]{R.id.numero, R.id.rue, R.id.zip, R.id.ville},
0);
```

Il est nécessaire que le cursor utilisé par SimpleCursorAdapter possède la colonne

_id integer

dont les valeurs uniques servent à identifier les lignes.

Sans colonne _id SimpleCursorAdaptor ne marche pas !

Il y a deux possibilités pour obtenir le cursor avec la colonne _id.

SimpleCursorAdapter et la colonne _id
1ère solution

Mettre dans chaque table la colonne `_id` comme la clé primaire :

```
_id integer primary key auto increment
```

et dans la requête SELECT sélectionner toujours cette colonne :

```
SQLiteDatabase bd = sqliteOpenHelper.getReadableDatabase();
String[] columns = new String[]{"_id", VILLE, ZIP, RUE, NUMERO};

/* ce cursor peut être utilisé avec SimpleCursorAdapter */
Cursor cursor = bd.query(ADRESY, columns, null, null, null, null, null);
```

Attention : dans la définition de `_id` vous ne pouvez pas remplacer "integer" par "int", cela ne marchera plus.

SimpleCursorAdapter et la colonne _id
2ème solution

Chaque table créé par SQLite possède déjà la colonne

```
rowid integer primary key
```

Au lieu d'ajouter une colonne `_id` il suffit de faire un alias sur cette colline dans la requête SELECT :

```
SQLiteDatabase bd = getReadableDatabase();
String[] columns = new String[]{"rowid as _id", VILLE, ZIP, RUE,
NUMERO};
Cursor cursor =  bd.query(ADRESY, columns, null, null, null, null,
null);

/* cursor obtenu peut de cette façon peut être utilisé avec SimpleCursorAdapter */
```

Est-ce qu'on est obligé d'utiliser SimpleCursorAdapter pour afficher les résultats de SELECT?

Non, utilisez RecyclerView à la place de ListView.

Les avantages :

- pas de contraintes artificielles et bizarres concernant la clé _id dans les tables,
- vous maitrisez totalement ce qui s'affiche dans RecyclerView grâce à votre propre RecyclerView.Adapter,
- pas de problème pour afficher les résultas d'un SELECT sur plusieurs tables.

Inconvénient :

- il faut écrire son propre RecyclerView.Adapter.

ContentProvider

- Activity
- ContentProvider
- Service
- BroadcastReceiver

ContentProvider

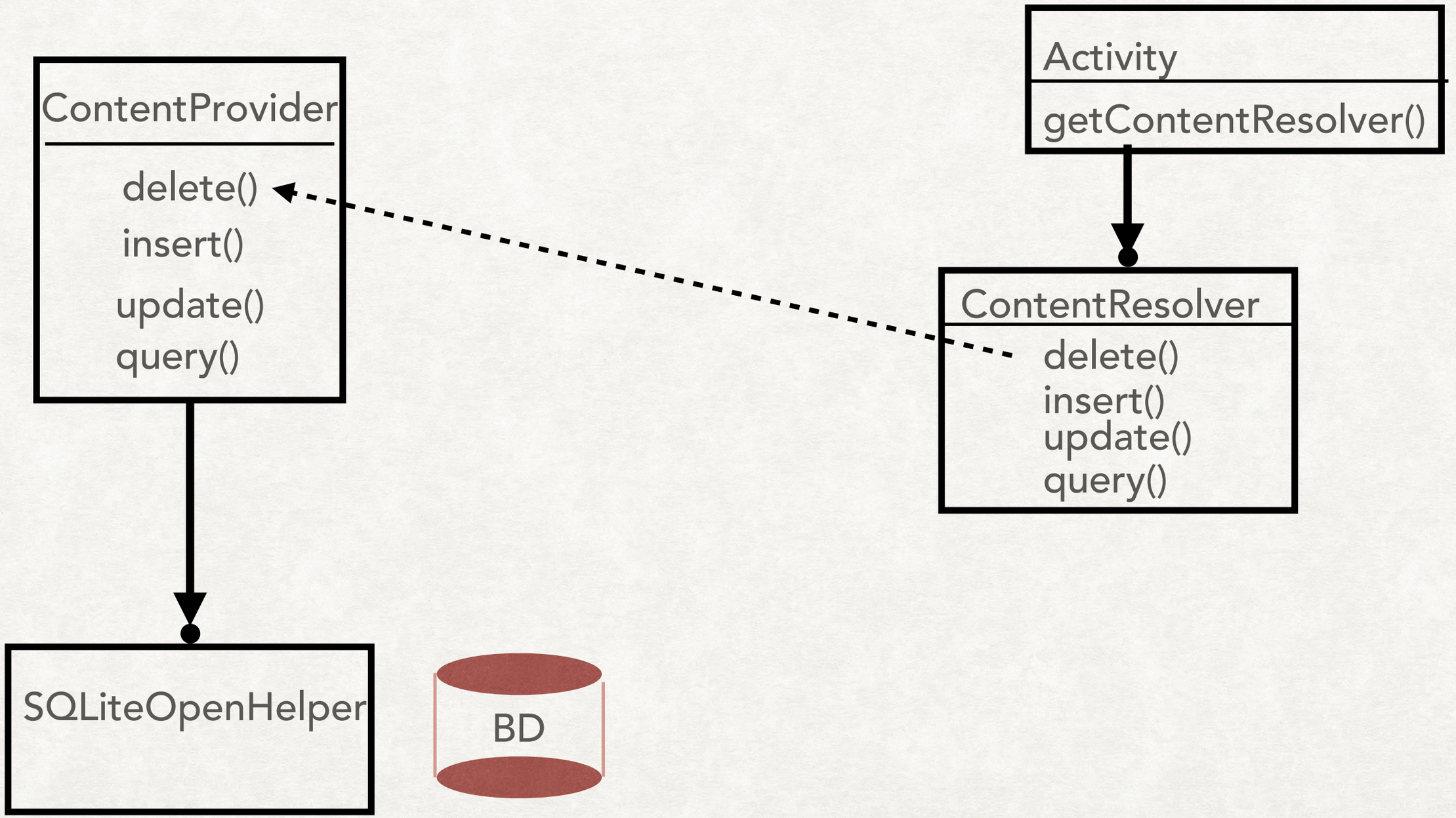
ContentProvider - composant qui gère les accès aux bases de données.

Le même ContentProvider peut être utilisé par plusieurs applications.

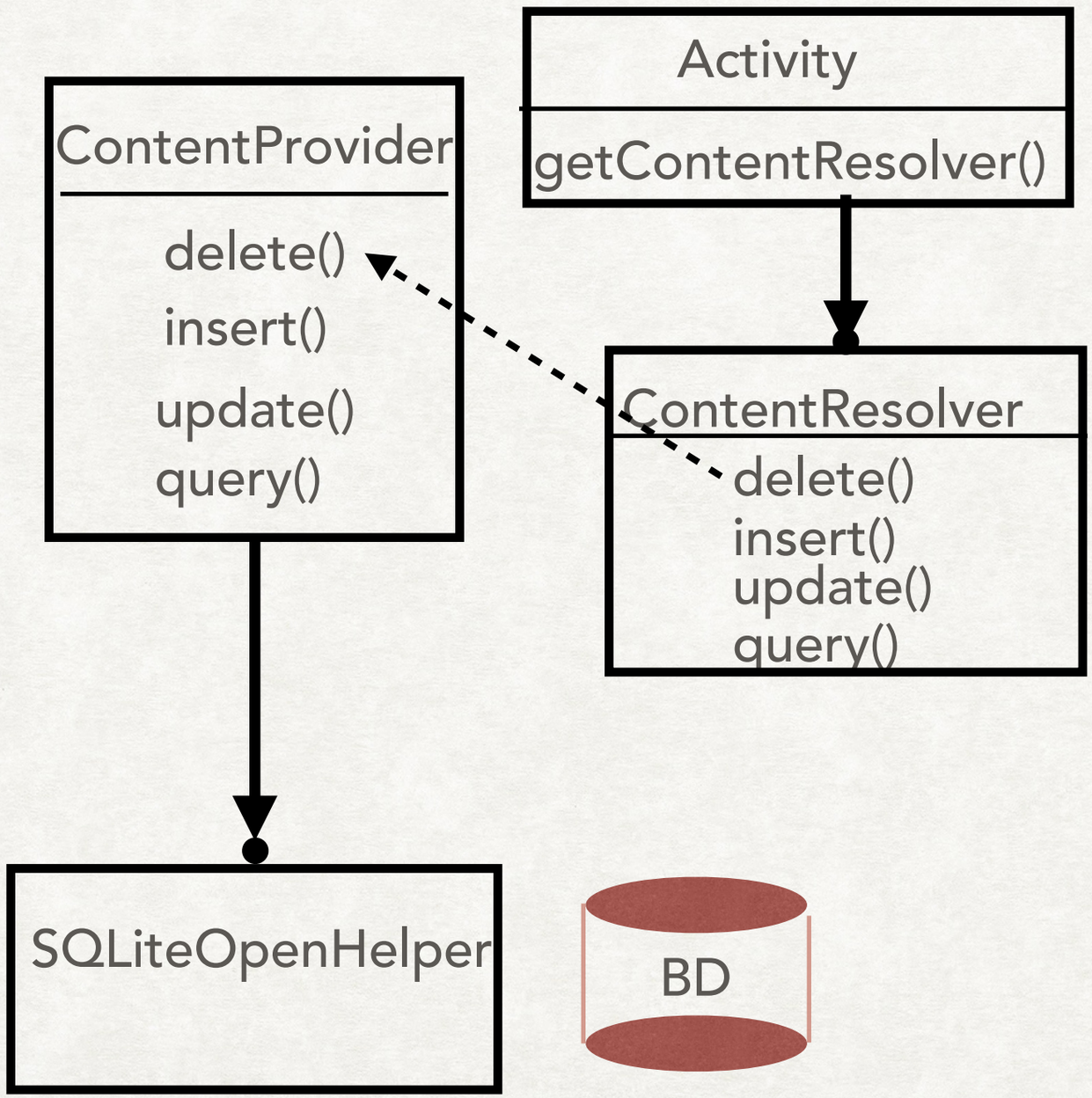
ContentProvider permet de restreindre la liberté de l'accès à la BD en imposant certaines formes de requêtes et pas d'autres.

Ne jamais accéder à une BD directement depuis Activity, toujours passer par un ContentProvider même pour une BD locale à l'application.

Architecture de l'accès à des bases de données (insert, update, delete)



Architecture de l'accès à des bases de données (insert, update, delete)



Problèmes:

1) il y a un seul ContentResolver par Activity et il doit "parler" avec tous les ContentProviders avec lesquels l'activité interagit

2) il y une seule méthode delete/insert/update/query et elle doit être capable d'exécuter toutes les opérations delete etc. sur toutes les tables.

3) query() peut bloquer l'interface graphique de Activity

ContentProvider et ContentResolver

Le problème 3) — solution :

dans l'activité il ne ne faut jamais faire query() directement sur ContentResolver

Alors comment faire les requêtes ?

Voir la suite du cours.

Vous pouvez faire update()/delete()/insert() en utilisant directement ContentResolver dans le thread de l'interface graphique.

Authority de ContentProvider

Chaque ContentProvider doit posséder une sorte d'identifiant unique qui le permet d'identifier, on l’appelle "authority"

Vous devez déclarer ContentProvider dans le fichier AndroidManifest de votre application (comme pour les activités).

AndroidManifest.xml

```
<provider
    android:name=".BookContentProvider"
    android:authorities="fr.irif.zielonka.bookcontentprovider"
    android:enabled="true"
    android:exported="false" />
```

"fr.irif.zielonka.bookcontentprovider" –– authority de mon ContentProvider

ContentProvider

Faire une classe dérivée de ContentProvider et implémenter les méthodes.

ContentProvider

```
boolean onCreate ()
Cursor query (Uri uri,
              String[] projection,
              String where,
              String[] whereArgs,
              String sortOrder)

Uri insert (Uri uri, ContentValues values)
int update (Uri uri,
              ContentValues values, String where,
              String[] whereArgs)

int delete (Uri uri, String where, String[] whereArgs)
```

ContentResolver

ContentResolver

```
final Cursor query (Uri uri,
                    String[] projection,
                    String where,
                    String[] whereArgs,
                    String sortOrder)

final Uri insert (Uri uri, ContentValues values)
final int update (Uri uri,
                  ContentValues values, String selection,
                  String[] selectionArgs)

final int delete (Uri uri, String where, String[] whereArgs)
```

On obtient un ContentResolver dans l'activité grâce à la méthode

ContentResolver getContentResolver()

Uri - Uniform Resource Identifier

Qu'est-ce qu'on code dans Uri?

Nos Uri auront toujours la forme suivante :

content://authority/path/id

- `content` – le schema, toujours le même
- *`authority`* – autorité, une chaîne de caractères sans '/' qui identifie de façon unique le `ContentProvider` (l'identifiant de `ContentProvider` déclaré dans `AndroidManifest` de celui-ci).
- *`path`* – *zéro* ou plusieurs segments séparés par '/' servent à passer des données, le nom de la table, etc. à `ContentProvider`
- *`id`* – identifiant numérique, utilisé pour identifier une ligne d'une table (clé primaire)

path et id sont facultatifs.

Preparer Uri dans ContentResolver

Pour construire plus facilement les Uri android nous fournit plusieurs classes:

`android.net.Uri` - la classe pour stocker Uri

`Uri.Builder` - pour construire un Uri:

```
private static String authority = "fr.irif.zielonka.bookcontentprovider";
Uri.builder builder = new Uri.Builder();
builder.scheme("content").authority(authority).appendPath("book_table");
Uri uri = builder.build();
uri = resolver.insert(uri,values);
```

`ContentUris` – pour ajouter un id à la fin de Uri

builder = ContentUris.appendId(builder, 5);

```
int n = resolver.delete(uri, null, null);
```

Decoder le contenu de Uri dans ContentProvider grâce à UriMatcher

Les actions à faire, par exemple dans query(), dépendent de l'information codé dans Uri.

Il y aura toujours aiguillage à faire en fonction du contenu de Uri.

```
String u = uri.toString();

if(u.equals("content//….    ")){

    insert dans la table A

}else if( u.equals("content//…   ")){

 insert dans la table B

}
```

etc.

TERRIBLE !

Pour faciliter cet aiguillage on utilise les méthodes de la classe `UriMatcher`

UriMatcher

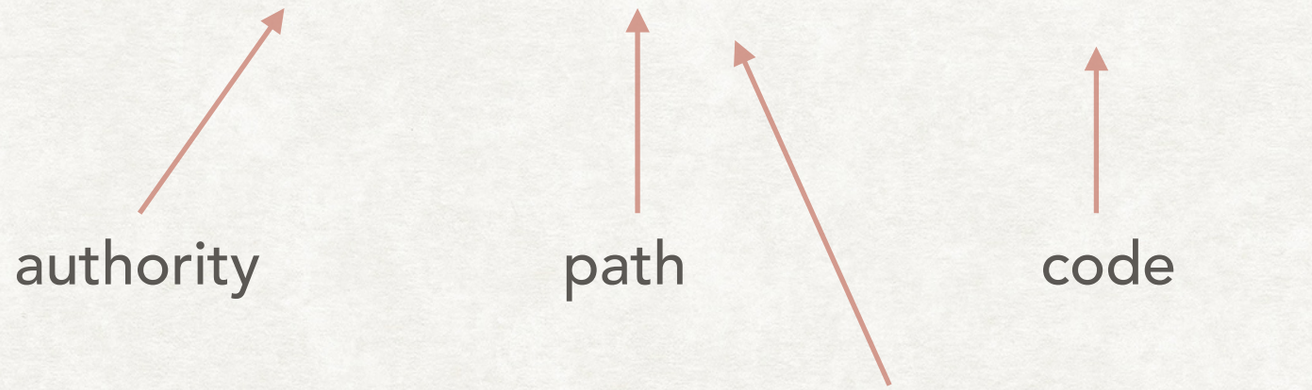
(utilisé dans ContentProvider)

```
private static String authority = "fr.irif.zielonka.bookcontentprovider";

private static final int AUTHOR = 1;
private static final int BOOK = 2;
private static final int AUTHOR_TITLE = 3;
private static final int ONE_BOOK = 4;

private static final UriMatcher matcher = new UriMatcher(UriMatcher.NO_MATCH);
```

```
static {
    matcher.addURI(authority, "author_table", AUTHOR);
    matcher.addURI(authority, "book_table",   BOOK);
    matcher.addURI(authority, "author_title", AUTHOR_TITLE);
    matcher.addURI(authority, "book/#",       ONE_BOOK);
   }
```



Utiliser UriMatcher dans insert() dans ContentProvider

```
public Uri insert(Uri uri, ContentValues values) {
    SQLiteDatabase db = helper.getWritableDatabase();
    int code = matcher.match(uri);
    String path ;

    switch (code) {
        case AUTHOR:
            id = db.insert("author_table", null, values);
            path = "author_table";
            break;
        case BOOK:
            id = db.insert("book_table", null, values);
            path = "book_table";
            break;
        default:
            throw new UnsupportedOperationException("Not yet
implemented");
    }
    Uri.Builder builder = (new Uri.Builder())
                              .authority(authority)
                              .appendPath(path);
    /* retourner Uri dans ContentResolver */
    return ContentUris.appendId(builder, id).build();
}
```

trouver le code de cet Uri

faire switch sur le code

UriMatcher query dans ContentProvider)

```
public Cursor query(Uri uri, String[] projection, String selection,
                    String[] selectionArgs, String sortOrder) {
    SQLiteDatabase db = helper.getReadableDatabase();
    int code = matcher.match(uri);
    Cursor cursor;
    switch (code) {
        case AUTHOR:
            cursor = db.query("author_table", projection, selection,
                    selectionArgs, null, null, sortOrder);
            break;
        case BOOK:
            cursor = db.query("book_table", projection, selection,
                    selectionArgs, null, null, sortOrder);
            break;
        case AUTHOR_TITLE:
            cursor = db.rawQuery("SELECT book_table._id as _id, nom, title " +
                    " FROM author_table, title_table " +
                    " where author_table._id = book_table.author and " +
                    " author_table.nom = % ", selectionArgs);
            break;
        case ONE_BOOK:
            long id = ContentUris.parseId(uri);
            cursor = db.query("book_table", new String[]{"_id", "title"},
                    "author = " + id , null, null, null,null);
            break;
        default:
            Log.d("Uri provider =", uri.toString());
            throw new UnsupportedOperationException("Not yet implemented");
    }
    return cursor;
}
```

onCreate() dans ContentProvider

```
private LivresSQLiteOpenHelper helper;

public boolean onCreate() {
    helper = LivresSQLite.getInstance(getContext());
    return true;
}
```

Construire SQLiteOpenHelper pour ContentProvider

```
public class LivresSQLiteOpenHelper extends SQLiteOpenHelper {
    private static int VERSION = 1;
    private static LivresSQLiteOpenHelper instance;
    private static String BOOKSBD = "BooksBD";

    private String author_table = "create table author_table( " +
            " nom varchar(30) not null," +
            " _id integer primary key "       +
    ");";
    private String book_table = "create table book_table (" +
            " title varchar(50) not null,"                    +
            " author int references author_table ,"           +
            " _id integer primary key "       +
    ");";

/* singleton – pour avoir un seul exemplaire de SQLiteOpenHelper */
    public static LivresSQLiteOpenHelper getInstance(Context context){
        if( instance == null ){
            instance = new LivresSQLite(context);
        }
        return instance;
    }

    private LivresSQLiteOpenHelper(Context context) {
        super(context, BOOKSBD, null, VERSION);
    }
```

Construire SQLiteOpenHelper pour ContentProvider

```
/* creation de la BD */
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(author_table);
        db.execSQL(book_table);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        if (newVersion > oldVersion) {
            db.execSQL("drop table if exists book_table ;");
            db.execSQL("drop table if exists author_table ;");

            onCreate(db);
        }
    }
}
```

Insérer un livre - le code dans Activity

```
String n = nom.getText().toString();
String t = titre.getText().toString();
ContentValues values = new ContentValues();
values.put("nom",n);

ContentResolver resolver = getContentResolver();
Uri.Builder builder = new Uri.Builder();
builder.scheme("content").authority(authority).appendPath("author_table");
Uri uri = builder.build();
uri = resolver.insert(uri,values);

long id = ContentUris.parseId(uri);
values = new ContentValues();
values.put("title",t);
values.put("author",id);

builder = new Uri.Builder();
builder.scheme("content").authority(authority).appendPath("book_table");
uri = builder.build();
uri = resolver.insert(uri,values);

nom.getText().clear();
titre.getText().clear();
```