Content Provider

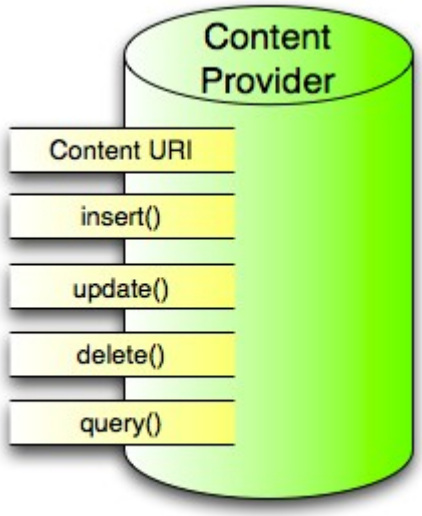
Table des matières

Qu’est-ce qu’un ContentProvider ?

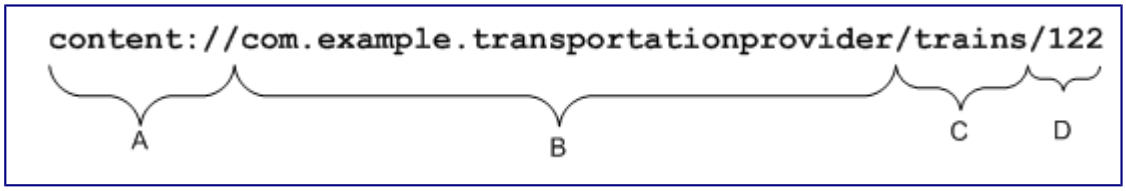
Un contentProvider sert à stocker et récupérer des données et ainsi les rendre accessibles à toutes les applications. C’est le moyen le plus connu de partager des données entres différentes applications. Par exemple, il existe un Content Provider gérant les Contacts d’un téléphone.

Android propose plusieurs ContentProviders basiques (audio, vidéo, images, informations sur les contacts…). Un contentProvider se compose d’une :

Uri
Méthodes (Insert, Update, Delete, Query).



Le chemin d’accès vers un **ContentProvider** se présente toujours sous forme d’une URI. Par exemple :



- **A** : Un préfixe standard, il sert à indiquer que les données sont contrôlées par un **ContentProvider**.
- **B** : L'autorité qui contrôle cette URI. Elle identifie le **ContentProvider** responsable de cette **URI**.
- **C** : Permet au **ContentProvider** de savoir quelle donnée est requêtée par l'url. Ce segment est optionnel. Un content provider peut exposer plusieurs données (ici les trains) mais on pourrait avoir par exemple les voitures, donc ce **ContentProvider** pourra gérer deux types de données.
- **D** : L'id de la donnée qu'on souhaite récupérer. (optionnel)

Comment créer un ContentProvider ?

Pour créer un ContentProvider, vous devez :

Mettre en place un systéme pour stocker vos données (les contents providers utilisent généralement le SQLite, la classe SQLiteOpenHelper vous facilite la création de votre base).

Etendre la classe ContentProvider.

Déclarer votre Content Provider dans le manifest (AndroidManifest.xml).

Un Content Provider doit surcharger les 6 méthodes suivantes :

query() : Cette méthode retourne un objet Cursor sur lequel vous pouvez itérer pour récupérer les données.

insert() : Cette méthode est utilisé pour rajouter des données à notre ContentProvider.

update() : Cette méthode est utilisé pour mettre à jour une données déjà existante dans notre Content Provider.

delete() : Cette méthode permet de supprimer une données du Content Provider.

getType() : Retourne le type MIME des données contenues dans le Content Provider.

Un Internet media type1, à l'origine appelé type MIME ou juste MIME ou encore Content-type2, est un identifiant de format de données sur internet en deux parties.

Un type MIME est composé d'au moins deux parties : un type et un sous-type et d'un ou plusieurs autres champs au besoin. Par exemple, les sous-types du type text ont un champ optionnel charset indiquant le codage des caractères

onCreate() : Appeler afin d'initialiser le Content Provider

Exemple

Nous allons étudier un exemple complet de création et l'utilisation d'un ContentProvider.
Notre provider s'appellera CoursProvider et permettra de stocker des cours.

Pour commencer, nous allons créer un projet :

Nom du projet : *com.example.tpcontentprovider*

SDK version : API niveau 15
Activité : `MainActivity`

Notre ContentProvider servira à gérer une liste de cours, chaque cours possédant un id, un nom et une description.

Pour commencer, nous allons créer la classe qui contiendra le nom des colonnes de notre table SQLite.

```
public class SharedInformation {

	public SharedInformation() {
	}

	public static final class Cours implements BaseColumns {

		private Cours() {}

		public static final String COURS_ID = "COURS";
		public static final String COURS_NAME = "COURS_NAME";
		public static final String COURS_DESC = "COURS_DESC";
	}
}
```

Cette classe est très simple, elle contient simplement l'identifiant des 3 colonnes. Elle doit absolument implémenter ***BaseColumns***.

Maintenant, nous allons créer une classe qui gérera notre base de données. Elle doit hériter de la classe ***SQLiteOpenHelper***, pour nous faciliter la création de notre base de données.
Cette classe se nommera : `CoursDatabaseHelper`

```
public class CoursDatabaseHelper extends SQLiteOpenHelper {

	// URI de notre content provider, elle sera utilisé pour accéder au
ContentProvider
	public static final Uri CONTENT_URI = Uri
		.parse("content://com.example.tpcontentprovider.CoursProvider");
```

```
	// Nom de notre base de données
		public static final String CONTENT_PROVIDER_DB_NAME = "cours.db";
	// Version de notre base de données
		public static final int CONTENT_PROVIDER_DB_VERSION = 1;
	// Nom de la table de notre base
		public static final String CONTENT_PROVIDER_TABLE_NAME = "cours";
	// Le Mime de notre content provider, la premiére partie est toujours
identique
		public static final String CONTENT_PROVIDER_MIME =
"vnd.android.cursor.item/vnd.com.example.tpcontentprovider.cours";



		// Création à partir du Context, du Nom de la table et du numéro de
version
		CoursDatabaseHelper(Context context) {

	super(context,CoursDatabaseHelper.CONTENT_PROVIDER_DB_NAME, null,
CoursDatabaseHelper.CONTENT_PROVIDER_DB_VERSION);
		}

		// Création des tables
		@Override
		public void onCreate(SQLiteDatabase db) {
			db.execSQL("CREATE TABLE " +
CoursDatabaseHelper.CONTENT_PROVIDER_TABLE_NAME + " (" + Cours.COURS_ID + "
INTEGER PRIMARY KEY AUTOINCREMENT," + Cours.COURS_NAME + " VARCHAR(255)," +
Cours.COURS_DESC + " VARCHAR(255)" + ");");
		}

		// Cette méthode sert à gérer la montée de version de notre base
		@Override
		public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
			db.execSQL("DROP TABLE IF EXISTS " +
CoursDatabaseHelper.CONTENT_PROVIDER_TABLE_NAME);
			onCreate(db);
		}

}
```

Cette classe comprend :

- La déclaration de l'uri de notre **ContentProvider**, vous remarquez bien le format « ***content://*** »
- Le nom de notre base de données
- La version de la base
- Le nom de notre table
- Le Mime correspondant à notre ContentProvider
- Une méthode ***onCreate***, qui sert à créer nos tables. Il s'agit d'une Requête SQL classique.
- Une méthode ***onUpgrade*** qui permet de monter de version dans une application.

Maintenant nous allons créer notre classe CoursProvider qui va hériter de ContentProvider.

Récupérer la classe CoursProvider.java

https://www.dropbox.com/s/xfux6fkhw8k4f0a/CoursProvider.java?dl=0

Question 1 °) Décrire la fonction de chaque méthode.

Création d'une classe de test

Avant de créer une fenêtre de test, il faut modifier le manifest et ajouter cette ligne :

```
<provider android:name=".CoursProvider"
            android:authorities="com.example.tpcontentprovider.CoursProvider" />
```

Nous allons modifier *notre activity* pour tester notre **CoursProvider**

Mofidier la méthode onCreate comme cela:

```
@Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            insertRecords();
            displayContentProvider();
      }
```

Créer la méthode `insertRecords()`

```
private void insertRecords() {
            ContentValues contact = new ContentValues();
            contact.put(Cours.COURS_NAME, "Android");
            contact.put(Cours.COURS_DESC, "Introduction à la programmation sous
Android");
            getContentResolver().insert(CoursDatabaseHelper.CONTENT_URI,
contact);

            contact.clear();
            contact.put(Cours.COURS_NAME, "Java");
            contact.put(Cours.COURS_DESC, "Introduction à la programmation
```

```
Java");
            getContentResolver().insert(CoursDatabaseHelper.CONTENT_URI,
contact);

            contact.clear();
            contact.put(Cours.COURS_NAME, "Iphone");
            contact.put(Cours.COURS_DESC, "Introduction à l'objectif C");
            getContentResolver().insert(CoursDatabaseHelper.CONTENT_URI,
contact);
      }
```

A quoi sert l'objet ContentValues ? Que doit on mettre dedans ?

Créer la méthode `displayContentProvider()`

```
private void displayContentProvider() {
            String columns[] = new String[] { Cours.COURS_ID, Cours.COURS_NAME,
Cours.COURS_DESC };
            Uri mContacts = CoursDatabaseHelper.CONTENT_URI;
            Cursor cur = managedQuery(mContacts, columns, null, null, null);

            Toast.makeText(MainActivity.this, cur.getCount() + "",
                        Toast.LENGTH_LONG).show();

            if (cur.moveToFirst()) {
                  String name = null;
                  do {
                        name = cur.getString(cur.getColumnIndex(Cours.COURS_ID))
+ " " +

      cur.getString(cur.getColumnIndex(Cours.COURS_NAME)) + " " +

      cur.getString(cur.getColumnIndex(Cours.COURS_DESC));
                        Toast.makeText(this, name + " ",
Toast.LENGTH_LONG).show();
                  } while (cur.moveToNext());
            }

      }
```

Que fait cette méthode ?
Que prend en paramètre la méthode managedQuery ? Faite le lien avec une requête SQL ?
Que retourne la méthode `cur.getCount()` ?
Que retourne la méthode `cur.getString(cur.getColumnIndex(Cours.COURS_ID))` ?

Application Tierce

Créer une autre application « AcessCoursProvider » récupérant le contenu du CoursProvider et l'affichant dans un Toast à la manière de **displayContentProvider ?**

Pouvez vous lire le contenu du CoursProvider ?

Modifier l'application tpcontentprovider afin qu'une autre application ne puisse pas lire le contenu du CoursProvider ? Modifier les permissions en lecture du CoursProvider dans le manifest de l'application

Source

http://developer.android.com/guide/topics/providers/content-providers.html
http://www.tutos-android.com/contentprovider-android
http://developer.android.com/training/contacts-provider/index.html