Persistance de donnée

SQLITE

Méthode de persistance de données

Préférences partagées (SharedPreferences)

- Sauvegarder les paramètres et l'état de l'application sous la forme de clé/valeurs dans un fichier de préférences partagé

SQLite

- SQLite est intégré dans le système android,
- L'API de gestion des bases de données SQLite est définie dans le package **android.database.sqlite**

Fichiers

- InputFileStream et OutputFileStream

SQLite

SQLite est un SGBDR embarqué et open source.

Caratéristiques

- SQLite est implémenté sous la forme d'une bibliothèque écrite en C.
- Utilise un typage dynamique
- Types d'affinités
  - TEXT
  - NUMERIC: la donnée est enregistrée comme un entier ou un réel si c'est possible (sinon elle sera enregistrée comme TEXT)
  - INTEGER: (entier, REAL,TEXT)
  - REAL
  - NONE: la donnée est enregistrée sans conversion.
  - INTEGER PRIMARY KEY: correspond à ROWID adresse de l'enregistrement dans la table.

Une base de données est privée et n'est accessible que par l'application qui l'a créée.

Le contrat

```
public final class DbContrat {

public static final String dbRestos = "dbrestos";
public static final int DB_VERSION = 1;

public static abstract class Restaurant implements
BaseColumns {

public static final String T_NOM = "restaurant";
public static final String C_NOM = "nom";
public static final String C_ADRESSE = "adresse";
public static final String SQL_CREATE = "create table
" + T_NOM + "("
+ _ID + "INTEGER PRIMARY KEY," + C_NOM + "TEXT ," +
C_ADRESSE
+ " TEXT )";
public static final String SQL_DROP = "DROP TABLE IF
EXISTS " + T_NOM;}}
```

Un contrat est une classe qui contient des constantes pour les noms des tables et des colonnes, ce qui permet d'utiliser les mêmes constantes dans toutes les classes du même package.

Structure de la classe contrat

- Les constantes globales sont définies comme des champs de la classe.
- Les constantes de chaque table seront définies dans une classe imbriquée.
- Les classes imbriquées implémentent l'interface BaseColumns pour hériter d'un champ clé primaire nommé _ID et utilisé par certaines classes du framework Android.

Création de la base de données: SQLiteOpenHelper

```
public class DbRestos extends SQLiteOpenHelper
{

public DbRestos(Context context) {
super(context, DbContrat.dbRestos, null,
DbContrat.DB_VERSION);
}

@Override
public void onCreate(SQLiteDatabase db) {
db.execSQL(DbContrat.Restaurant.SQL_CREATE);
}

@Override
public void onUpgrade(SQLiteDatabase db, int
oldVersion, int newVersion) {
db.execSQL(DbContrat.Restaurant.SQL_DROP);
onCreate(db);
}
}
```

La classe abstraite SQLiteHelper automatise les tâches de création et de mise à jour du schéma d'une base de données.

- onCreate est appelée si la base de données n'existe pas.
- onUpgrade est appelée si la version de la base de données est supérieure à celle de la base de données existante.
- onDowngrade appelée si la version de la base de données est inférieure à celle de la base de données existante.

La base de données est créée dans le dossier /data/data/nom_package/databases

Utilisation de la base de données

Créer une instance de la classe qui implémente SQLiteOpenHelper

- `DbRestos dbrestos=new DbRestos(this);` passer le contexte en paramètre au constructeur, this dans le cas où l'instatiation est effectuée dans l'activité.

Obtenir une instance de la base de données

- En mode lecture/écriture
  - `SQLiteDatabase db =dbrestos.getWritableDatabase();`
- En mode lecture
  - `SQLiteDatabase db =dbrestos.getReadableDatabase();`

Ajout d'un enregistrement

```
ContentValues valeurs = new ContentValues();
valeurs.put(DbContrat.Restaurant.C_NOM, restaurant.getNom());
valeurs.put(DbContrat.Restaurant.C_ADRESSE,restaurant.getAdresse());
// 2eme param nom d'une colonne dans laquelle la méthode
// peut insérer une valeur nulle.
long res = db.insert(DbContrat.Restaurant.T_NOM, null, valeurs);
```

Supprimer un enregistrement

```
String selection = DbContrat.Restaurant._ID + " LIKE ?";
// déifinir la valeur de l'iD
String[] selectionArgs = { String.valueOf(1) };
// Exécuter la suppression
db.delete(DbContrat.Restaurant.T_NOM, selection, selectionArgs);
```

Modifier un enregistrement

```
// Définir les nouvelles valeurs dans un objet de type ContentValues
ContentValues valeurs = new ContentValues();
valeurs.put(DbContrat.Restaurant.C_NOM, "Nouveau Nom");
// Sélection de la ligne à modifer selon son id
String selection = DbContrat.Restaurant._ID + " LIKE ?";
String[] selectionArgs = { String.valueOf(1) };
db.update(DbContrat.Restaurant.T_NOM, valeurs, selection, selectionArgs);
```

Remarque:

- les opérations d'insertion, modification et suppression peuvent être aussi effectuées à l'aide de la méthode db.execSQL(String SQL).

Lecture des données

```
String [] projection ={
DbContrat.Restaurant.C_NOM,
DbContrat.Restaurant.C_ADRESSE
};

SQLiteCursor curseur=(SQLiteCursor) db.query(
    DbContrat.Restaurant.T_NOM,  //nom de la table
    projection,//un tableau contenant les noms des
colonnes à retourner
    //null correspond à toutes les colonnes
    null,// la clause where qui peut contenir des jokers ?
    null,// un tableau de valeurs qui remplaçent les ?
    null, // group by
    null, // clause having
    null //le tri
    );
Toast message;
while(curseur.moveToNext()) {

message=Toast.makeText(this, curseur.getInt(0)+":"
+curseur.getString(1)+ curseur.getString(2),
Toast.LENGTH_LONG);
message.show();
}
```

Pour exécuter une requête de type select utiliser db.query ou bien db.rawQuery, Exemple:

- `Cursor curseur=db.rawQuery("select * from restaurant", null);`
- le premier paramètre est une instruction Select
- Le second peut contenir un tableau de valeurs qui remplaçent les “?” de la clause where.

Méthodes d’un curseur:

- moveToFirst(), moveToNext(), moveToPrevious(), moveToLast(), move(int offset), moveToPosition(int position)
- getColumnIndex (String nom_colonne), getCount()

ListActivity

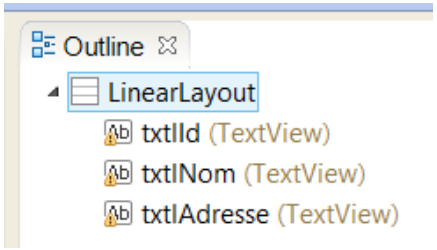
```
public class Liste extends ListActivity {

private SimpleCursorAdapter sca;
private SQLiteCursor curseur;


@Override
protected void onCreate(Bundle savedInstanceState) {
..
setContentView(R.layout.activity_liste_restaurants);
DbRestos dbrestos = new DbRestos(this);
SQLiteDatabase db = dbrestos.getReadableDatabase();
curseur = (SQLiteCursor) db.rawQuery("select * from restaurant", null);
// Colonnes à lier
String[] colonnes = { DbContrat.Restaurant._ID,
DbContrat.Restaurant.C_NOM, DbContrat.Restaurant.C_ADRESSE };
// les vues à lier
int[] vues = { R.id.txtlId, R.id.txtlNom, R.id.txtlAdresse };

sca = new SimpleCursorAdapter(this, R.layout.liste_element, curseur,
colonnes, vues, 0);

this.setListAdapter(sca);
}
…

}
```

Le layout activity_liste_restaurants doit contenu une ListView dont l'id est :

`android:id="@android:id/android:list"`

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".Liste" >

    <ListView
        android:id="@android:id/android:list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="27dp" >
    </ListView>

</RelativeLayout>
```

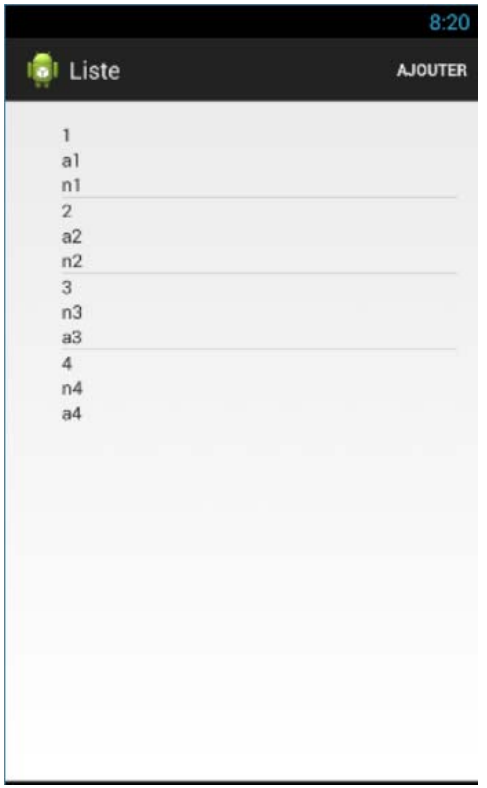
Le layout liste_element.xml




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/txtlId"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView" />

    <TextView
        android:id="@+id/txtlNom"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView" />

    <TextView
        android:id="@+id/txtlAdresse"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView" />

</LinearLayout>
```

Navigation: Activité ListeRestaurants

```
public class ListeRestaurants extends ListActivity {
// Afficher l'activité AjoutRestaurant
@Override
public boolean onOptionsItemSelected(MenuItem item) {
Intent i = new Intent(this, AjouterRestaurant.class);
startActivityForResult(i, 1);
return super.onOptionsItemSelected(item);
}

// Retour de l'activité AjouterRestaurant
@Override
protected void onActivityResult(int requestCode, int
resultCode, Intent data) {
// Rafraîchir les données à partir de la base de données
curseur.requery();
// Mettre à jour les éléments de la liste à partir du curseur
sca.notifyDataSetChanged();
}
…
}
```



Navigation: Activité AjouterRestaurant

```
//Retour vers l'activité ListeRestaurant
public boolean onOptionsItemSelected(MenuItem item) {
Intent i=new Intent();
setResult(1);
finish();
return super.onOptionsItemSelected(item);
}
```