android

An Open Handset Alliance Project

Persistance en bases de données

O.Legrand
G. Seront

Bases de données en local

- Utilisation de SQLite :
  - SGBD disponible sur tout mobile Android;
- Toute application peut créer ses propres BD;
- Par défaut :
  - BD privée et accessible uniquement par l'application qui l'a créée;
  - BD stockée en : /data/data/<nom_package>/databases
- Si *Content Provider* :
  - accès possible par les autres applications.

developer.android.com/guide/topics/data/data-storage.html#db

Bases de données en local

- Toute application peut utiliser les BD natives :
  - BD des contacts;
  - BD des fichiers audio et vidéo;
  - BD des préférences des utilisateurs;
  - …
- Permissions à ajouter au *Manifest File:*
  - Exemple :

```
<uses-permission  android:name="android.permission.READ_CONTACTS"/>
```

developer.android.com/guide/topics/data/data-storage.html#db

SQLite

- SQLite est :
  - un SGBD connu et présent sur de nombreux mobiles : lecteurs *MP3, iPhone, iPod,…*
  - open source;
  - compatible avec les standards;
  - léger (écrit en C, librairie incluse dans Android);
  - fiable;
  - faible typage des colonnes (plusieurs types possibles pour une même colonne)

www.sqlite.org

Création d'une BD

- Pour créer une BD :
  - étendez la classe abstraite *SQLiteOpenHelper;*
  - redéfinissez ses méthodes *onCreate() et onUpgrade();*
  - créez une instance de cette classe;
  - appelez les méthodes :
    - *getWritableDatabase();*
    - *getReadableDatabase();*

developer.android.com/guide/topics/data/data-storage.html#db

```
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);
    }
}
```

```
FeedReaderDbHelper mDbHelper = new FeedReaderDbHelper(getContext());
```

```
// 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);
```

http://developer.android.com/training/basics/data-storage/databases.html

Bonnes pratiques

- Externalisez les définitions de la DB dans une classe « 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";
        ...
```

http://developer.android.com/training/basics/data-storage/databases.html

Adapteur de BD

- Utilisez un adapteur de BD afin de limiter le couplage entre les composants applicatifs et la BD; (bonne pratique)
- Cet adapteur sert de couche d'abstraction :
  - il encapsule les interactions avec la BD;
  - il doit fournir des méthodes intuitives pour :
    - créer, ouvrir, monter en version la BD, la fermer;
    - consulter, ajouter, supprimer, modifier les données;

developer.android.com/guide/topics/data/data-storage.html#db

Adapteur de BD

- il peut aussi publier les noms des tables et des colonnes (constantes statiques).
- la classe qui étend *SQLiteOpenHelper* sera définie comme classe interne de l'adapteur.

developer.android.com/guide/topics/data/data-storage.html#db

```
public class MonDBAdapteur {
    public static final String NOM_BD = "MaDB.db";
    public static final int VERSION_BD = 1;
    public static final String NOM_TABLE = "tableVentes";
    public static final String COLONNE_ID = "_id";
    public static final String COLONNE_VENTES = "ventes";
    public static final int INDICE_COLONNE_ID = 0;
    public static final int INDICE_COLONNE_VENTES = 1;
    private static final String CREATION_TABLE = "create table " + NOM_TABLE
            + " ( " + COLONNE_ID + " integer primary key autoincrement, "
            + COLONNE_VENTES + " real);";
    private SQLiteDatabase db;
    private MonOpenHelper dbHelper;

    public MonDBAdapteur(Context context) {
        this.dbHelper = new MonOpenHelper(context);
    }
    public void open() throws SQLException {
        this.db = this.dbHelper.getWritableDatabase();
    }
    public void close() {
        this.db.close();
    }
    public long ajouter(float montantVente) {
    public boolean supprimer(double indice) {
    public boolean modifier(long indice, float nouvelleValeur) {
    public Cursor getToutesLesLignes() {
    public Cursor getLigne(int indice) {
    private static class MonOpenHelper extends SQLiteOpenHelper {
}
```

Interroger une BD

- Pour exécuter une requête : *db.query()*
- Renvoie un *Cursor* qui contient les résultats de la requête (=*ResultSet*)
- Exemples :

```
public Cursor getToutesLesLignes() {
    return db.query(NOM_TABLE, new String[] { COLONNE_ID, COLONNE_VENTES },
            null, null, null, null, null);
}

public Cursor getLigne(int indice) {
    String where = COLONNE_ID + "=" + indice;
    return db.query(NOM_TABLE, null, where, null, null, null, null);
}
```

developer.android.com/guide/topics/data/data-storage.html#db

Ecrire sur une BD

```
// 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);
```

developer.android.com/guide/topics/data/data-storage.html#db

BD : *Cursor*

- Quelques méthodes d'instance de *Cursor* :
  - *moveToFirst() : boolean;*
  - *moveToNext() : boolean;*
  - *moveToPrevious() : boolean;*
  - *getCount() : int;*
  - *getColumIndexOrThrow(nomColonne : String) : int;*
  - *getColumnName(indiceColonne : int) : String;*
  - *getColumnNames() : String[];*
  - *getPosition() : int;*

developer.android.com/reference/android/database/Cursor.html

Extraire les données du *Cursor*

- Se positionner sur la ligne :
  - *moveToFirst(), moveToNext(), …*
- Utilisez une méthode *get<type>* en passant l'indice de la colonne:
- Exemple :

```
// utiliser un curseur
Cursor cursor = dbAdapteur.getToutesLesLignes();
float totalVentes=0;
float moyenneVentes=0;
if ( cursor.moveToFirst()) {
    do {
        totalVentes += cursor.getFloat(MonDBAdapteur.INDICE_COLONNE_VENTES);
    } while( cursor.moveToNext() );
    moyenneVentes = totalVentes/cursor.getCount();
}
```

developer.android.com/guide/topics/data/data-storage.html#db

BD : Gestion du *Cursor*

- Pour les versions plus anciennes d'Android, la gestion du *Cursor* peut être prise en charge par l'activité, en appelant :
  - *startManagingCursor()* : démarre la gestion du curseur par l'activité;
  - *stopManagingCursor()* : demande à l'activité de stopper la gestion du curseur
  - *cursor.close()* : libère les ressources.
- Pour les versions plus récentes :
  - Utilisez les classes *CursorLoader* et *LoaderManager*

developer.android.com/reference/android/database/Cursor.html

Insérer une ligne

- Construisez une instance de *ContentValues;*
- Utilisez sa méthode *put()* pour affecter une valeur à chaque colonne;
- Appelez la méthode *insert()* sur la BD pour l'insérer dans la table;

```
public long ajouter(float montantVente) {
    long indice;
    // création de la nouvelle ligne à insérer
    ContentValues nouvelleLigne = new ContentValues();

    // assignation des valeurs à chaque colonne
    nouvelleLigne.put(COLONNE_VENTES, montantVente);

    // insertion de la ligne
    indice = this.db.insert(NOM_TABLE, null, nouvelleLigne);
    return indice;
}
```

Modifier une ligne

- Construisez une instance de *ContentValues;*
- Utilisez sa méthode *put()* pour affecter les nouvelles valeurs aux colonnes;
- Appelez la méthode *update()* sur la BD pour modifier la ligne spécifiée dans la table;

```
public boolean modifier(long indice, float nouvelleValeur) {
    // création d'un contentValue
    ContentValues ligneModifiée = new ContentValues();

    // assignation des valeurs à chaque colonne
    ligneModifiée.put(COLONNE_VENTES, nouvelleValeur);

    // clause where spécifiant la ligne à modifier
    String where = COLONNE_ID + "=" + indice;

    // modification de la ligne
    int indiceMod = this.db.update(NOM_TABLE, ligneModifiée, where, null);
    return indiceMod > 0;
}
```

Supprimer une ligne

- Appelez la méthode *delete()* sur la BD en spécifiant la table et l'indice de la ligne à supprimer;

```
public boolean supprimer(double indice) {
    String where = COLONNE_ID + "=" + indice;
    int indiceSup = this.db.delete(NOM_TABLE, where, null);
    return indiceSup > 0;
}
```

developer.android.com/guide/topics/data/data-storage.html#db