http://guendouz.wordpress.com/2012/11/23/utilisation-de-la-base-de-donnees-sqlite-en-java/

[JAVA] Utilisation de la base de données SQLite en JAVA

Dans ce Tutorial je vous présentez comment utiliser une base de données embarqué dans un projet java , de la connexion au base de donnée a la manipulation des données (Projection,Ajout,Mise a Jour etc …) et en fin comment bien gérer l'accès au base de donnée (Ouverture & Fermeture)

Prérequis :

1- Java : des notions fondamentales sur le langage (Classes , Méthodes …)

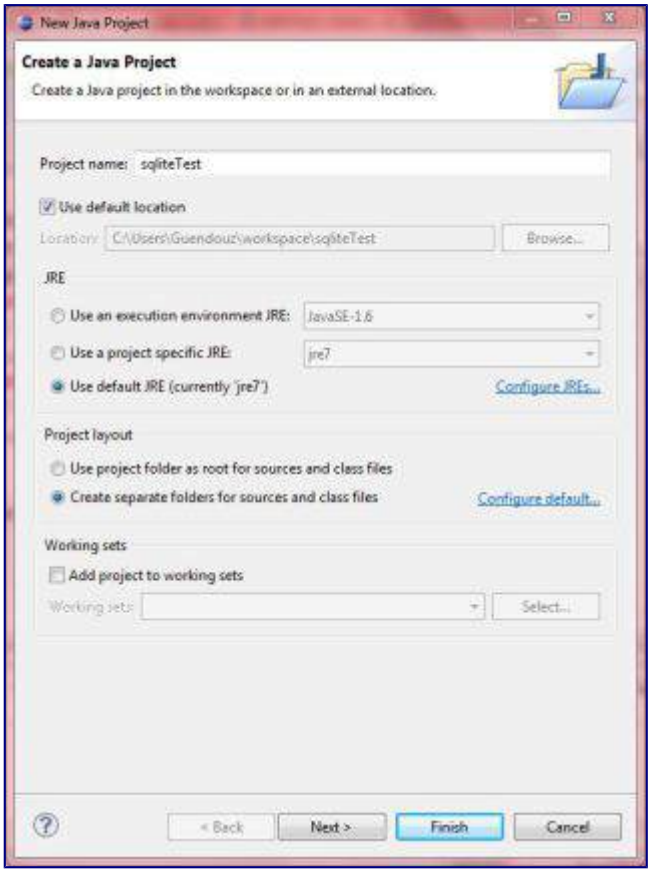
2- Bases de données : les Tables (Création , Ajout) syntaxe des Requêtes (SELECT , DELETE …)

3- SQLite JDBC : un driver JDBC pour accéder a une BD SQLite , on va utiliser ce Driver

4- une base de donnée SQLite (fichier .db) Telecharger ICI

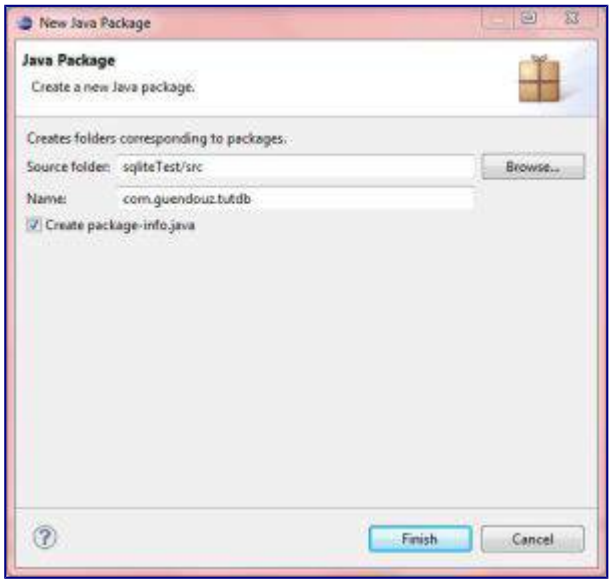
1 – Installation & Configuration :

ouvrir Eclipse et choisiser File -> New -> Java Project



après donner un nom au projet cliquer sur Finish

créer un nouveau package java et lui donner un nom comme **com.votre_nom.nom_application** (Ex : **com.guendouz.tutdb**)



Maintenant créer un dossier dans le Project sous le nom “lib” pour sauvegarder toutes les bibliothèques que l’on utilise (ex : driver jdbc) pour cela Clique droit sur l’icone du Project dans le “Package Explorer” -> New -> Folder

après télécharger et décompresser le **SQLite JDBC** copier le fichier dans le dossier “lib” de votre projet et l’ajouter a **Referenced Libraries** en cliquent sur le fichier jar -> Build Path -> Add to Build Path

2-Connexion à la base de données :

Pour connecter a une base de donnée en java ont doivent créer une Classe

```
1  package com.guendouz.tutdb;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.SQLException;
6  import java.sql.Statement;
7
8  public class Connexion {
9      private String DBPath = "Chemin aux base de donnée SQLite";
10     private Connection connection = null;
11     private Statement statement = null;
12
13     public Connexion(String dBPath) {
14         DBPath = dBPath;
15     }
16
17     public void connect() {
18         try {
19             Class.forName("org.sqlite.JDBC");
20             connection = DriverManager.getConnection("jdbc:sqlite:" +
21 DBPath);
22             statement = connection.createStatement();
23             System.out.println("Connexion a " + DBPath + " avec
24 succès");
25         } catch (ClassNotFoundException notFoundException) {
26             notFoundException.printStackTrace();
27             System.out.println("Erreur de connecxion");
28         } catch (SQLException sqlException) {
29
```

```
                sqlException.printStackTrace();
                System.out.println("Erreur de connecxion");
30          }
31      }
32
33      public void close() {
34          try {
35              connection.close();
36              statement.close();
37          } catch (SQLException e) {
38              e.printStackTrace();
39          }
40      }
    }
```

La Classe Main pour tester la connexion

```
1 package com.guendouz.tutdb;
2
3 public class Main {
4
5      public static void main(String[] args) {
6          Connexion connexion = new Connexion("Database.db");
7          connexion.connect();
8          connexion.close();
9      }
10
11 }
```

3- **Fouiller dans la base de données :**

avant que nous puisent lire les données dans la BD il faut q'ont ajoute une méthode "**query**" a la classe Connexion qui prend en paramètre la requête et retourne un Objet **ResultSet**

```
1 SELECT * FROM Book
```

la methode query :

```
1 public ResultSet query(String requet) {
2         ResultSet resultat = null;
3         try {
4             resultat = statement.executeQuery(requet);
5         } catch (SQLException e) {
6             e.printStackTrace();
7             System.out.println("Erreur dans la requet : " + requet);
8         }
9         return resultat;
10
11     }
```

Example 1 : afficher tout les titres des livres dans la Table Book

```
package com.guendouz.tutdb;

import java.sql.ResultSet;
import java.sql.SQLException;

public class Main {

    public static void main(String[] args) {
        Connexion connexion = new Connexion("Database.db");
        connexion.connect();
        ResultSet resultSet = connexion.query("SELECT * FROM BOOK");
        try {
            while (resultSet.next()) {
                System.out.println("Titre :
"+resultSet.getString("Title"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        connexion.close();
    }

}
```

Résultat après Exécution

```
Connexion a Database.db avec succès
Titre : Programmation JAVA
Titre : C++ par la pratique
Titre : L'art du développement Android
```

Example 2 : ajouter un livre a la base de donnée

pour ajouter un livre a notre base de donnée il suffit d'utiliser une autre Classe (**Book**) pour simplifier notre tache

la classe Book contient tous les attributs de la Table Book en plus des getters et les setters

```
package com.guendouz.tutdb;

import java.sql.Date;

public class Book {
    private String BookId;
    private String Title;
    private String SubTitle;
    private int Pages;
    private Date Published;
    private String Description ;
}
```

Astuce Eclipse :

pour générer automatiquement des getters et des setters des attributs cliquer sur Source -> Generate Getters and Setters -> Choisiser les Attributs

pour générer automatiquement un Constructeur a base des attributs cliquer sur Source -> Generate constructor using Fields -> Choiser les Attributs

et voila notre Classe Book après le traitement

```
1  package com.guendouz.tutdb;
2
3  import java.sql.Date;
4
5  public class Book {
6      private String BookId;
7      private String Title;
8      private String SubTitle;
9      private int Pages;
10     private Date Published;
11
12     public Book(String bookId, String title, String subTitle, int
13 pages,
14             Date published, String description) {
15         super();
16         BookId = bookId;
17         Title = title;
18         SubTitle = subTitle;
19         Pages = pages;
20         Published = published;
21         Description = description;
22     }
23
24     public String getBookId() {
25         return BookId;
26     }
27     public void setBookId(String bookId) {
28         BookId = bookId;
29     }
30     public String getTitle() {
31         return Title;
32     }
33     public void setTitle(String title) {
34         Title = title;
35     }
36     public String getSubTitle() {
37         return SubTitle;
38     }
39     public void setSubTitle(String subTitle) {
40         SubTitle = subTitle;
41     }
42     public int getPages() {
43         return Pages;
44     }
45     public void setPages(int pages) {
46         Pages = pages;
47     }
48     public Date getPublished() {
49         return Published;
50     }
51     public void setPublished(Date published) {
52         Published = published;
53     }
54
```

```
    public String getDescription() {
55      return Description;
56  }
57  public void setDescription(String description) {
58      Description = description;
59  }
60  private String Description ;
61
}
```

maintenant ont ajoute une méthode a la Classe Connexion “addBook” qui prend en paramètre un objet de la classe **Book**

```
1 public void addBook(Book book) {
2         String query = "";
3         query += "INSERT INTO BOOK VALUES (";
4         query += "'" + book.getBookId() + "', ";
5         query += "'" + book.getTitle() + "', ";
6         query += "'" + book.getSubTitle() + "', ";
7         query += book.getPages() + ", ";
8         query += "'" + book.getPublished().toString() + "', ";
9         query += "'" + book.getDescription() + "' )";
10        try {
11            statement.executeUpdate(query);
12        } catch (SQLException e) {
13            // TODO Auto-generated catch block
14            e.printStackTrace();
15        }
16
17  }
```

Test :

```
1 package com.guendouz.tutdb;
2
3 import java.sql.Date;
4
5 public class Main {
6
7     public static void main(String[] args) {
8         Connexion connexion = new Connexion("Database.db");
9         connexion.connect();
10
11        Book book = new Book("3ff883e6-dfc0-4149-902a-4dbdfa22a408",
12                "Programmation Android", "", 450, Date.valueOf("2012-
13 11-01"),
14                "");
15        connexion.addBook(book);
16
17        connexion.close();
18    }
19
}
```

essayent maintenant d’ajouter un livre avec le titre Système D’Information

```
1 package com.guendouz.tutdb;
2
```

```
import java.sql.Date;

public class Main {

    public static void main(String[] args) {
        Connexion connexion = new Connexion("Database.db");
        connexion.connect();

        Book book = new Book("9aa883e6-dfc0-4149-902a-4dbdfa22a408",
                "Système D'Information", "", 450, Date.valueOf("2012-
11-01"),
                "");
        connexion.addBook(book);

        connexion.close();
    }

}
```

Résultat :

```
Connexion a Database.db avec succès
java.sql.SQLException: near "Information": syntax error
    at org.sqlite.DB.throwex(DB.java:288)
    at org.sqlite.NestedDB.prepare(NestedDB.java:115)
    at org.sqlite.DB.prepare(DB.java:114)
    at org.sqlite.Stmt.executeUpdate(Stmt.java:102)
    at com.guendouz.tutdb.Connexion.addBook(Connexion.java:65)
    at com.guendouz.tutdb.Main.main(Main.java:15)
```

Cet Exception est déclenché lorsque ont écrit une requête SQL avec une mauvaise syntaxe dans notre example voici l'erreur

```
INSERT INTO BOOK VALUES ('9aa883e6-dfc0-4149-902a-4dbdfa22a408',
'Système D'Information', '', 450, '2012-11-01', '' )
```

Question : Comment faire pour corriger ces erreurs en JAVA ?
Réponse : il suffit d'utiliser les requêtes préparés

En Java ont utilisent la Classe **PreparedStatement** pour généré des requêtes préparés voici la Méthode "**addBook**" version **PreparedStatement**

```
public void addBook(Book book) {
        try {
            PreparedStatement preparedStatement = connection
                    .prepareStatement("INSERT INTO Book
VALUES(?,?,?,?,?,?)");
            preparedStatement.setString(1, book.getBookId());
            preparedStatement.setString(2, book.getTitle());
            preparedStatement.setString(3, book.getSubTitle());
            preparedStatement.setInt(4, book.getPages());
            preparedStatement.setDate(5, book.getPublished());
            preparedStatement.setString(6, book.getDescription());
            preparedStatement.executeUpdate();
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
```

Résultat Finale :

```
1 Connexion a Database.db avec succès
2 Insertion Avec Succées
```

Voici le code source des exemples ICI … et Bientôt