

# JDBC – L’art de stocker ses données

---

## Introduction

Nous allons va utiliser l’API JDBC (Java DataBase Connectivity) qui propose un pilote associé à un grand nombre de bases de données. Nous allons utiliser la base de données **SQLite** qui sera amplement suffisamment pour nos besoins, à savoir, écrire et lire les tâches créées par l’utilisateur de notre programme.

## Installation

Au moment où ces lignes sont écrites, la version **sqlite-jdbc-3.21.0.jar** est disponible. Pour l’installer, rendez-vous sur :

<https://bitbucket.org/xerial/sqlite-jdbc/downloads/>

Nous allons placer ce fichier dans notre workspace : `eclipse-workspace/votre-projet/lib` (créez le répertoire s’il n’existe pas).

Sous eclipse, depuis votre vue « Projet Explorer », vérifiez que vous avez bien ce nouveau répertoire « lib » ainsi que l’archive Java (**sqlite-jdbc-3.21.0.jar**) qui apparaissent.

Faites un clique-droit sur l’archive > Build Path > Add to Build Path (Illustration 1).

Cela permettra au compilateur d’intégrer cette archive dont a besoin notre programme pour fonctionner.

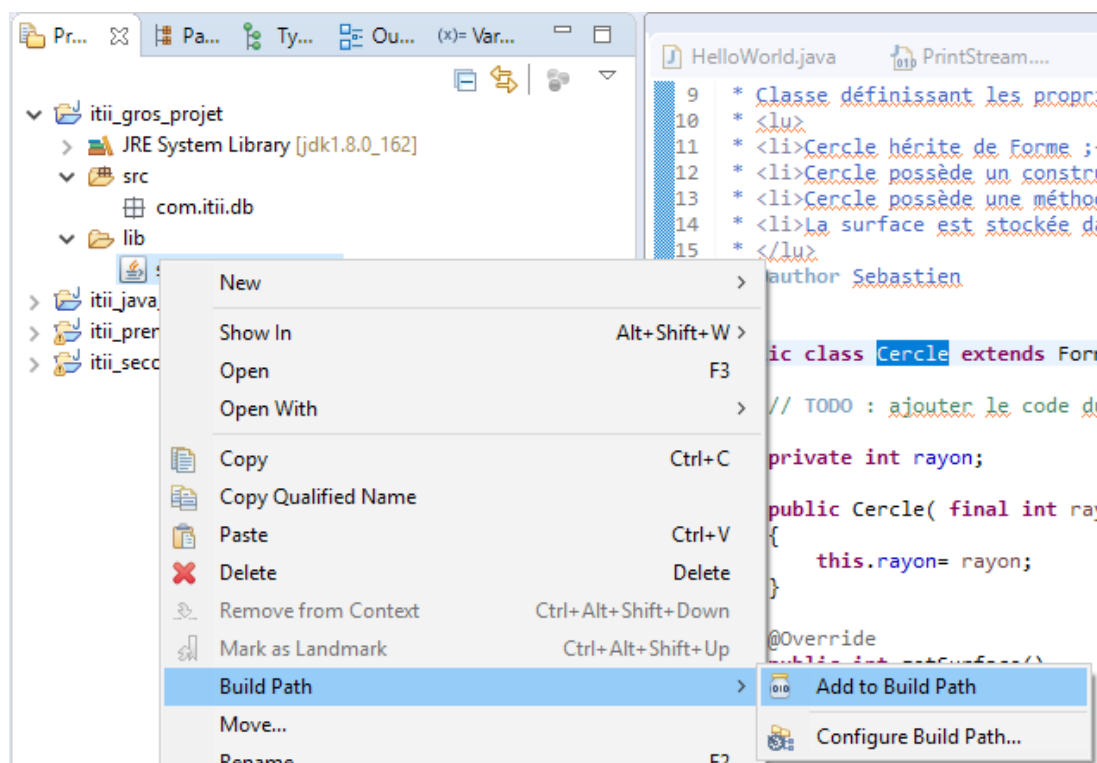


Illustration 1: Ajout de l'archive au Build Path

# Test de bon fonctionnement

Le fichier dans lequel seront stockées les données sera nommé **planning.db**. Il sera créé dans le répertoire eclipse-workspace/**database** par le bout de code donné ci-après. Pensez à créer le répertoire database, soit à la main, soit au travers de l'utilisation de la classe File directement dans votre code java.

Le code suivant vous permet de tester le bon fonctionnement de **SQLite** et du **pilote**. JDBC

```
package com.itii.db; // Adapter en fonction de votre package

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

/**
 * Test de connection / exécution de requêtes SQL / déconnction de SQLite
 * @author Sebastien
 */
public class SQLiteTest
{
    private static final String TABLE_NAME = "Tasks";
    private static final String FIELD_ID = "id";
    private static final String FIELD_NAME = "name";
    private static final String FIELD_DATE = "date";
    private static final String FIELD_DETAILS = "details";
    private static final String FIELD_STATE = "state";

    public static void main(String[] args)
    {
        Connection connection = null;
        Statement statement = null;

        // Création de la table
        try
        {
            // Chargement du Driver. Stockage des données dans le fichier planning.db
            connection = DriverManager
                .getConnection("jdbc:sqlite:database/planning.db");
            // Objet permettant l'exécution des requêtes SQL
            statement = connection.createStatement();
            // Timeout en cas de non-réponse de la base de données.
            statement.setQueryTimeout(15);

            statement.execute("drop table " + TABLE_NAME);
            // Création de la table
            statement.executeUpdate("create table " + TABLE_NAME + " ( "
                + FIELD_ID + " integer primary key autoincrement, " // Primary key
                + FIELD_NAME + " string, " // Name
                + FIELD_DATE + " text, " // Details
                + FIELD_DETAILS + " text, " // date as ISO8601 strings ("YYYY-MM-DD HH:MM:SS.SSS").
                + FIELD_STATE + " boolean " + " )"); // marquée
            System.out.println("table \"" + TABLE_NAME + "\" créée ");
        } catch (SQLException e)
        {
            System.out.println(" Table non créée ou déjà existante");
            e.printStackTrace();
        }

        // Test d'écriture dans la table
        try
        {
            PreparedStatement stmt = connection.prepareStatement(
                "insert into " + TABLE_NAME + " ( " + FIELD_NAME + ", "
                + FIELD_DATE + ", " + FIELD_DETAILS + ", "
                + FIELD_STATE + " ) " + "values ( ?, ?, ?, ? )");
            stmt.setString(1, "TP #1");
            stmt.setString(2, "2018-04-20 12:00");
            stmt.setString(3, "penser à rendre le tp");
        }
    }
}
```

```

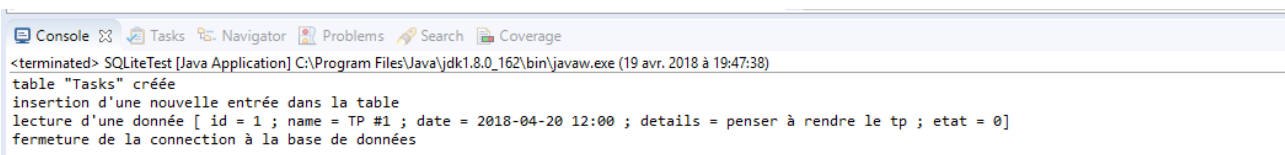
        stmt.setString(4, "0");
        stmt.executeUpdate();
        System.out.println("insertion d'une nouvelle entrée dans la table");
    } catch (SQLException e)
    {
        System.out.println("problème dans l'insertion d'une nouvelle enrée dans la table.");
    }

    // Test de lecture depuis la table
    try
    {
        ResultSet rs = statement.executeQuery("select * from " + TABLE_NAME);
        while (rs.next())
        {
            System.out.print("lecture d'une donnée [");
            System.out.print(" id = " + rs.getString(FIELD_ID));
            System.out.print(" ; name = " + rs.getString(FIELD_NAME));
            System.out.print(" ; date = " + rs.getString(FIELD_DATE));
            System.out.print(" ; details = " + rs.getString(FIELD_DETAILS));
            System.out.println(" ; etat = " + rs.getString(FIELD_STATE) + "]\n");
        }
    } catch (SQLException e)
    {
        System.out.println("erreur à la lecture de la table");
    } finally
    {
        try
        {
            if (connection != null)
            {
                connection.close();
            }
            System.out.println("fermeture de la connection à la base de données");
        } catch (SQLException e)
        {
            System.out.println("erreur lors de la fermeture de la connection");
        }
    }
}
}

```

Si tout se déroule bien, à la première exécution vous devriez obtenir le résultat suivant :

- Le fichier **planning.db** est créé dans le répertoire */database*.
- La console sous Eclipse affiche le résultat suivant :



```

<terminated> SQLiteTest [Java Application] C:\Program Files\Java\jdk1.8.0_162\bin\javaw.exe (19 avr. 2018 à 19:47:38)
table "Tasks" créée
insertion d'une nouvelle entrée dans la table
lecture d'une donnée [ id = 1 ; name = TP #1 ; date = 2018-04-20 12:00 ; details = penser à rendre le tp ; etat = 0]
fermeture de la connection à la base de données

```

A l'aide de ce code d'exemple fourni, écrivez les méthodes permettant la création de la table, l'écriture et la lecture d'une entrée dans la base de données.

Ces méthodes seront à intégrer à votre programme pour que :

- À chaque démarrage, le programme vérifie si la table existe et la crée si elle n'existe pas.
- Le programme lise l'ensemble des tâches écrites en base de données, les charge puis les affiche.
- À chaque création d'une tâche par l'utilisateur du programme, celle-ci est immédiatement écrite en base de données.