

Objectifs

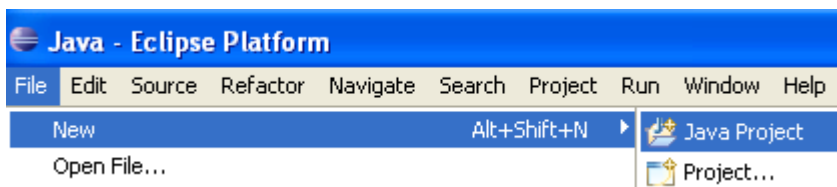
- Mettre en oeuvre l'API JDBC java.sql
- Comprendre les étapes clés pour l'insertion de données
- Comprendre les 6 étapes clés pour la récupération de données
- Découvrir javax.sql

Programme

- Partie 1 : création du projet
- Partie 2 : ajout du driver JDBC MySql
- Partie 3 : création table
- Partie 4 : Insertion données en base
- Partie 5 : récupération de données
- Partie 6 : optimisations

Partie 1 : projet eclipse demojdbc

- A partir d'Eclipse (Standard ou Enterprise), Menu File->New



L'écran suivant apparaît.

New Java Project

Create a Java Project
Create a Java project in the workspace or in an external location.

Project name:

Contents

☒ Create new project in workspace
☐ Create project from existing source

Directory:

JRE

☒ Use default JRE (Currently 'jre6') [Configure JREs...](#)
☐ Use a project specific JRE:
☐ Use an execution environment JRE:


Project layout

☐ Use project folder as root for sources and class files
☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

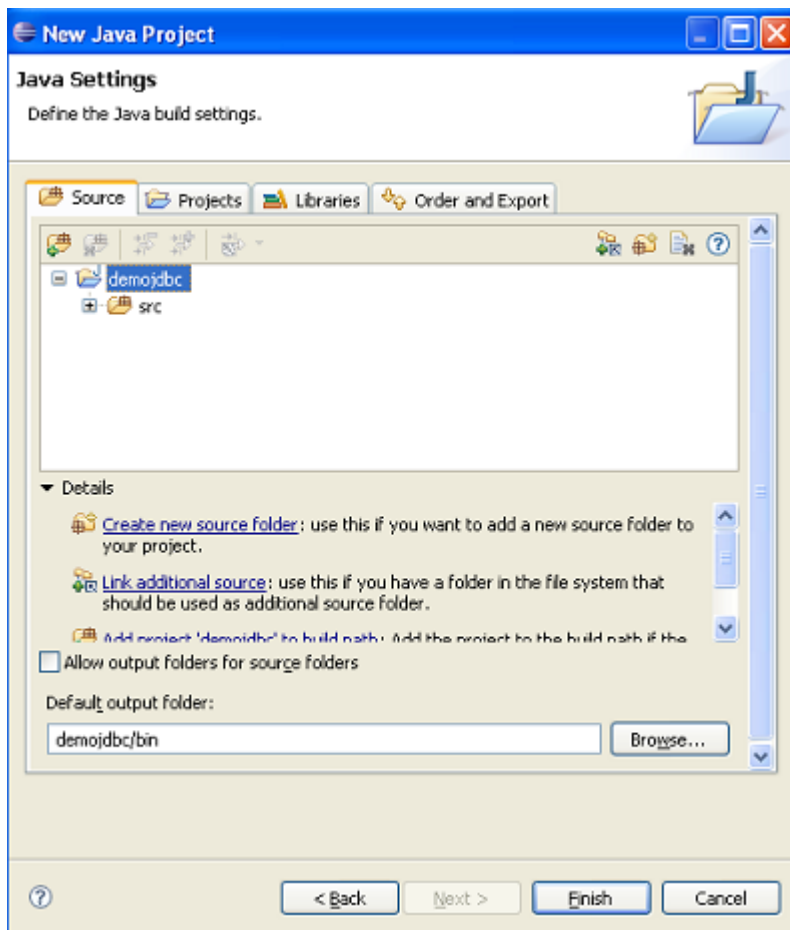
☐ Add project to working sets

Working sets:

 The wizard will automatically configure the JRE and the project layout based on the existing source.

► Ajoutez le nom du projet (ici demojdbc) dans le champ 'Project Name', puis appuyez sur bouton 'Next'.

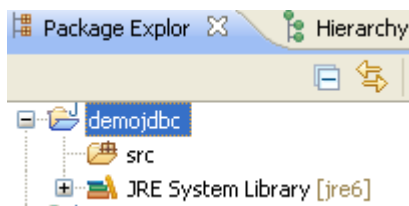
L'écran suivant apparaît.



Eclipse déposera les fichiers compilés (.class) dans le répertoire bin.

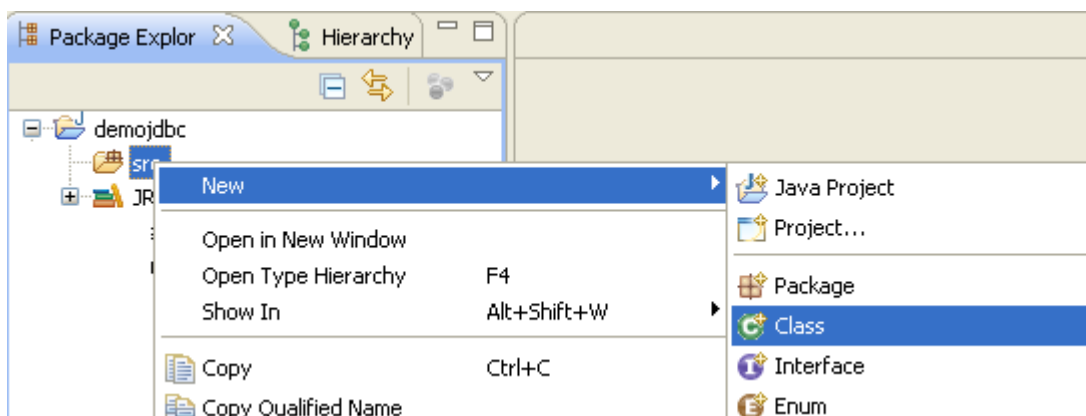
- Cliquez sur 'Finish'

Le projet apparaît dans la vue 'Explorateur de package'.

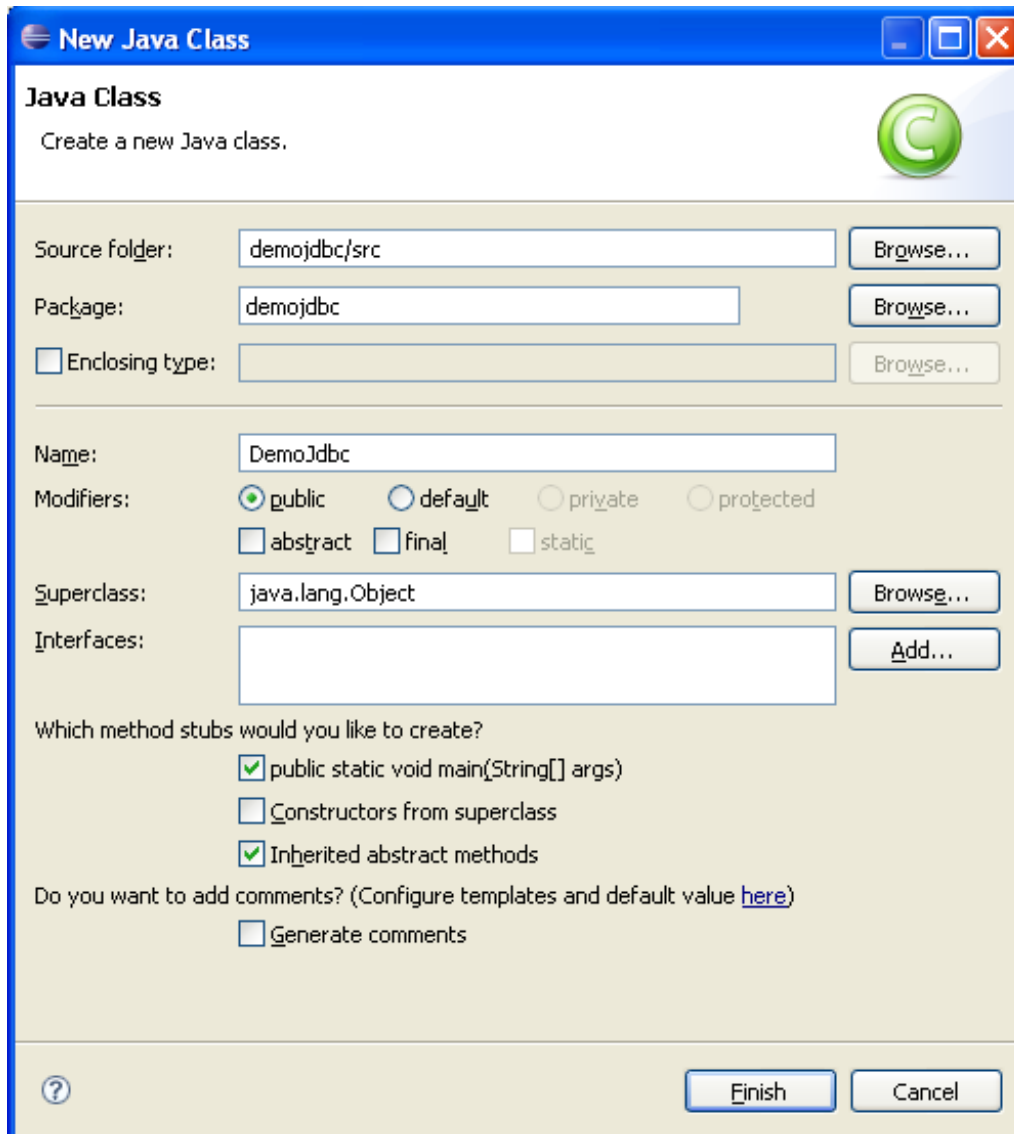


Création package et classe

- Cliquez droit sur src->new->Class

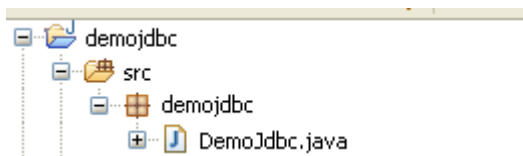


L'écran suivant apparaît.



- Entrez dans le champ Name le nom de classe : **DemoJdbc**
- Entrez dans le champ Package le nom du package dans lequel la classe sera : **demojdbc**
- Cochez la case 'public static void main()' afin de lancer l'application à partir de cette classe.

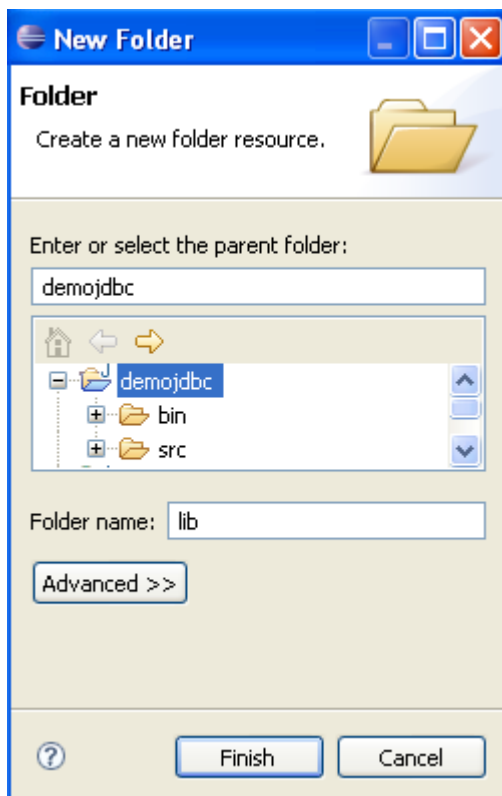
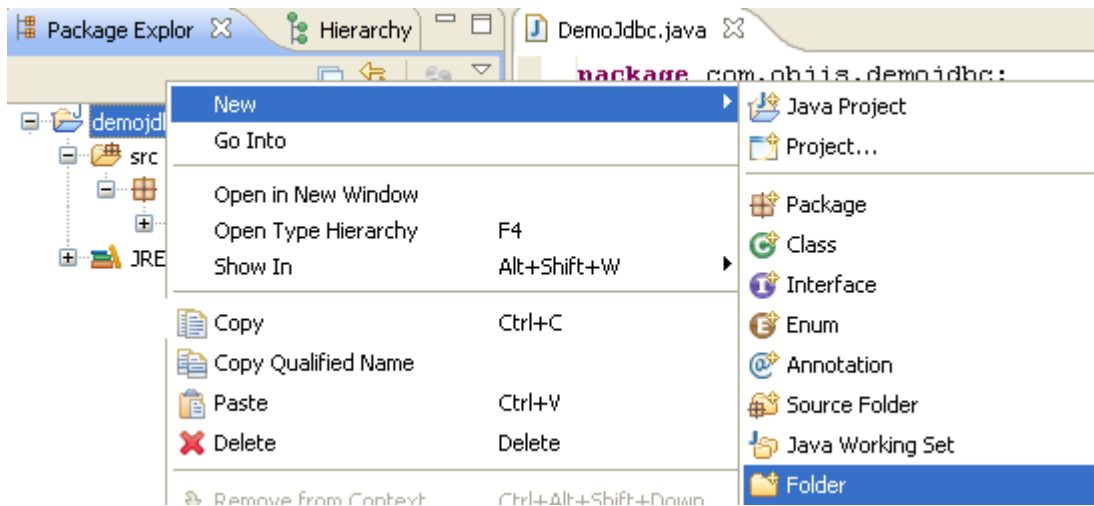
Puis bouton Finish. Eclipse met à jour la vue Explorer.



Partie 2 : Ajout du driver JDBC

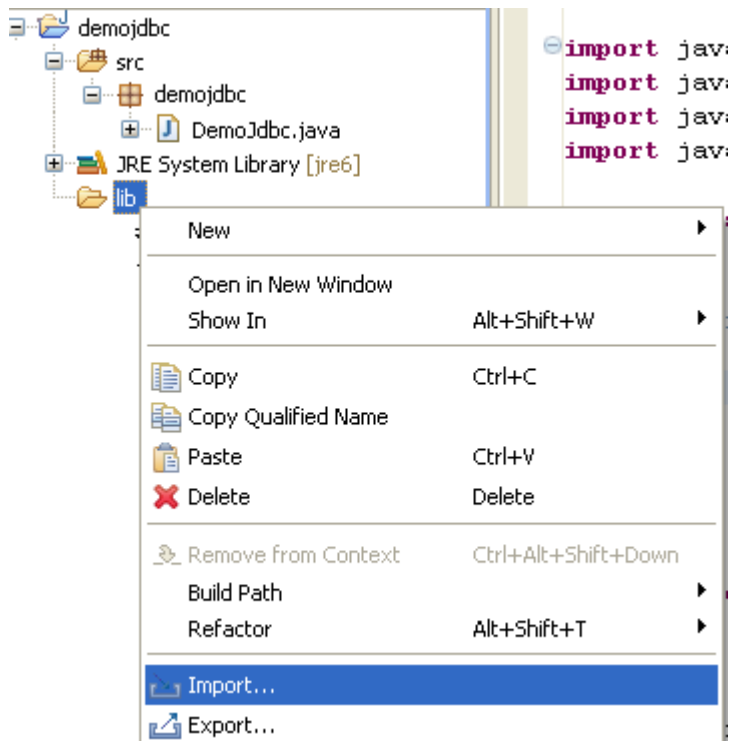
Librairie du projet (Driver JDBC) et CLASSPATH eclipse

Créez un répertoire lib à la racine du projet

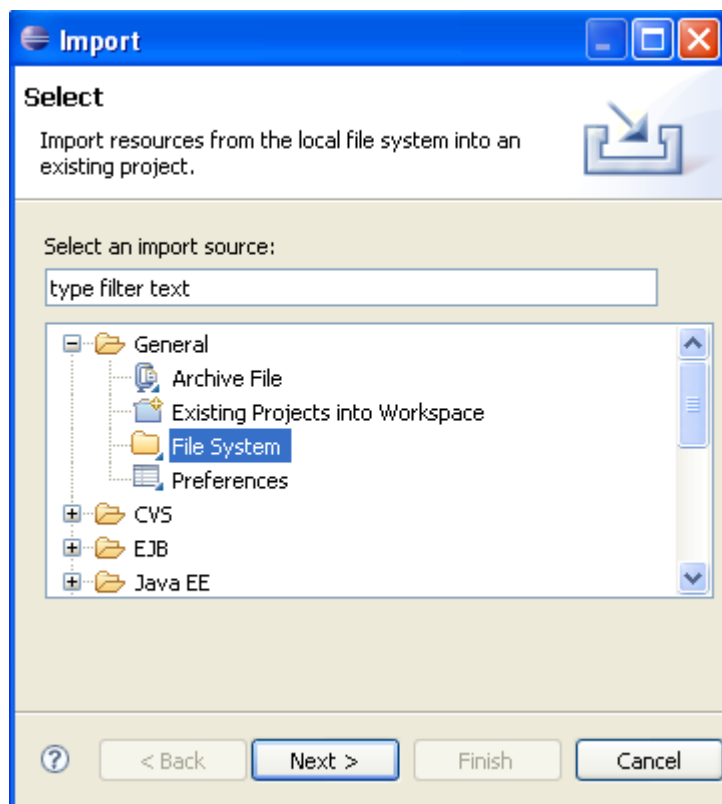


Dans ce répertoire importez le driver jdbc :

- cliquez-droit puis import



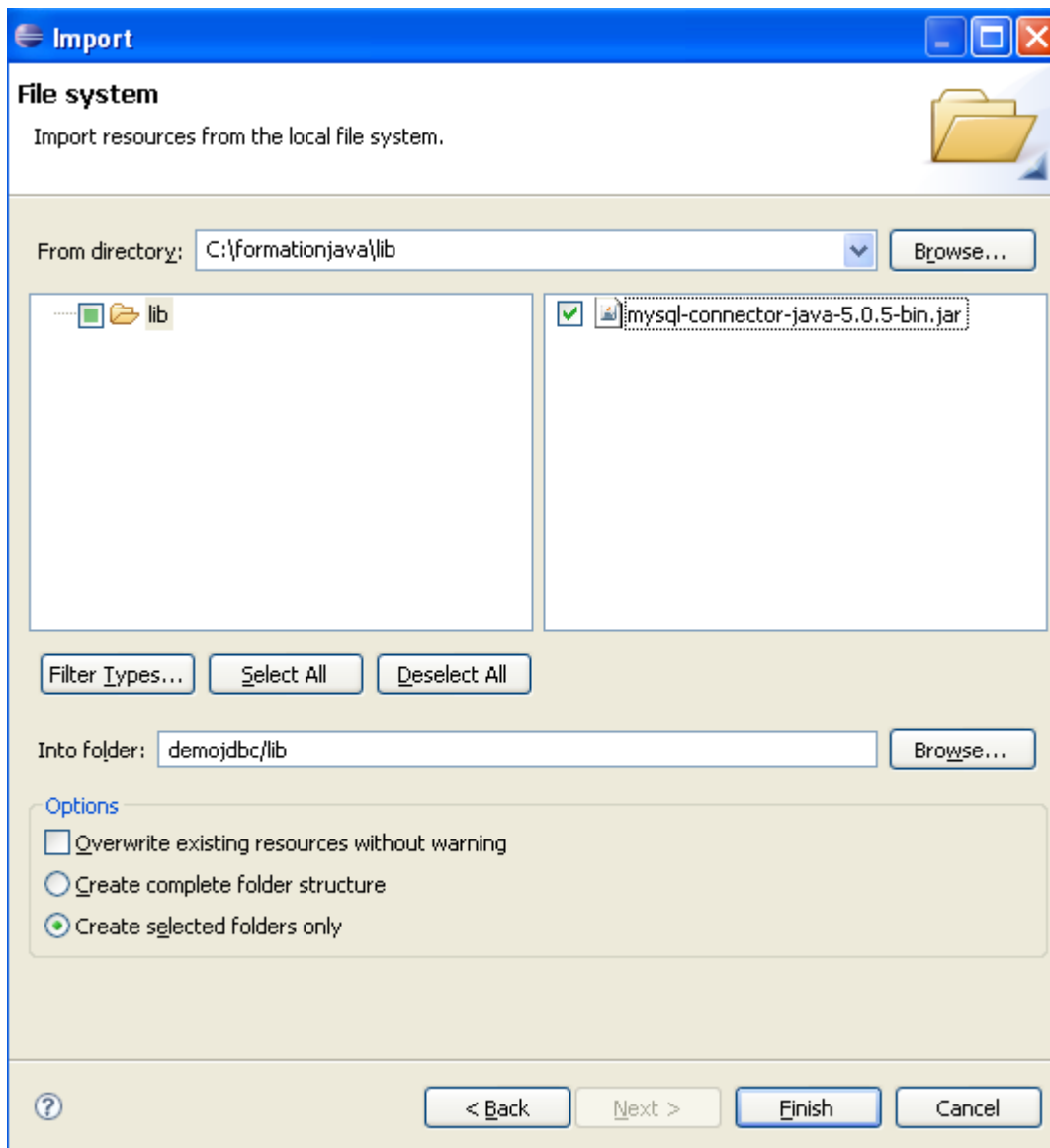
General/File System



naviguez dans votre système jusqu'à trouver le driver (ici mysql-connector-java-5.0.5.jar, récupéré suite au dézippage

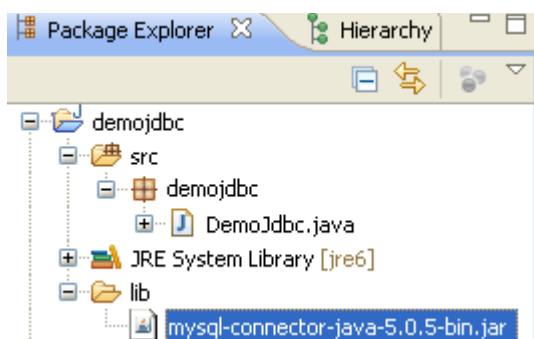
<http://dev.mysql.com/downloads/connector/j/>

)

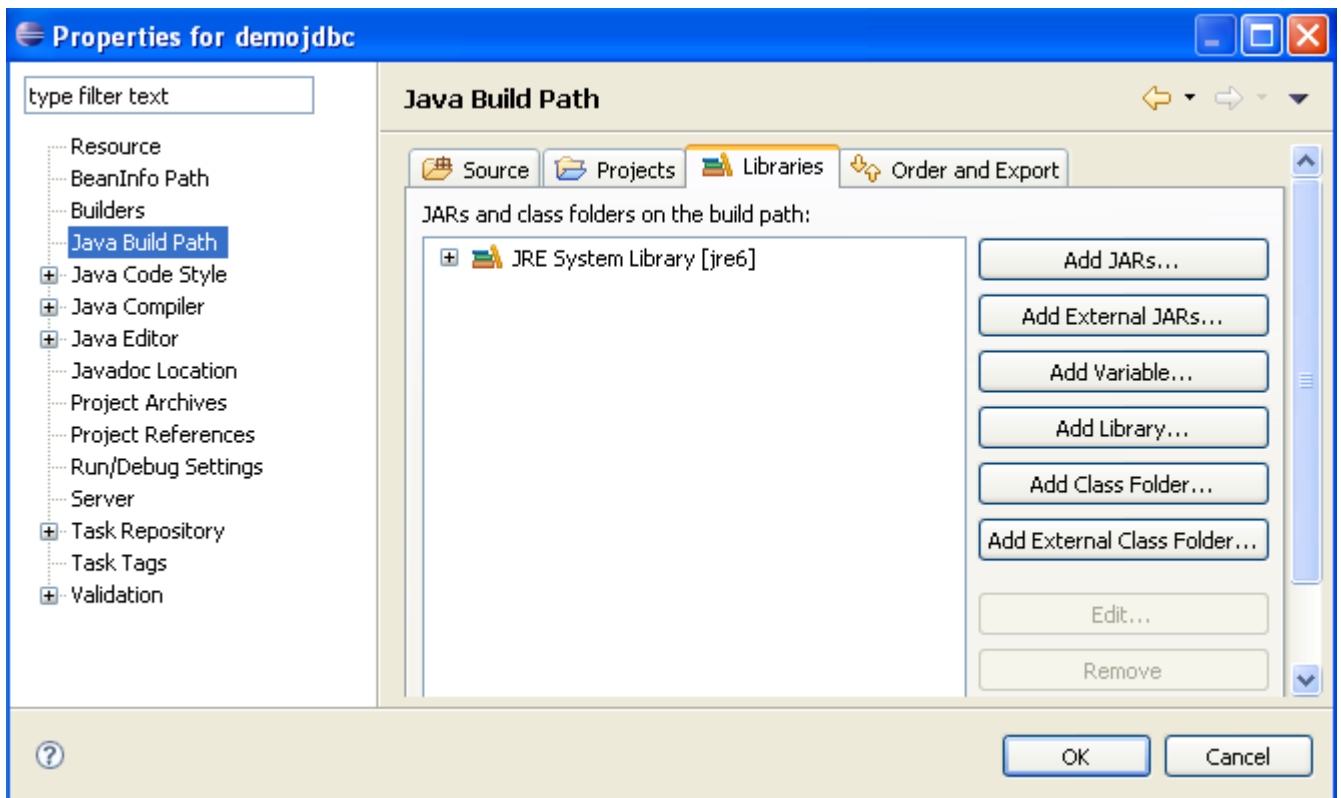


► Cliquez sur Finish

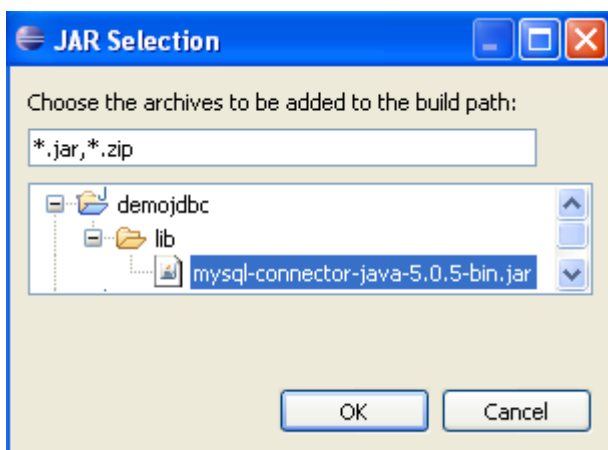
Le driver apparaît dans la vue.



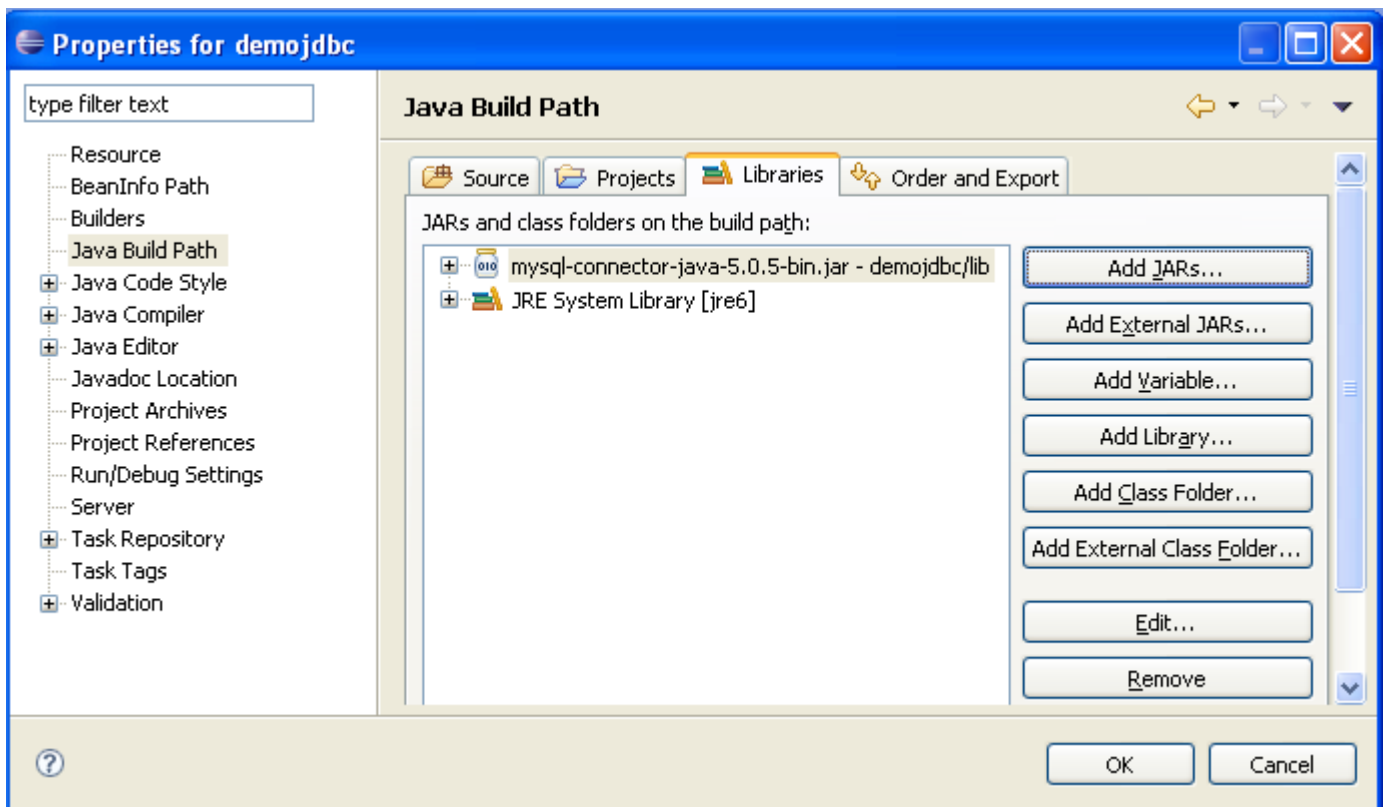
Informez Eclipse de l'existence de ce driver



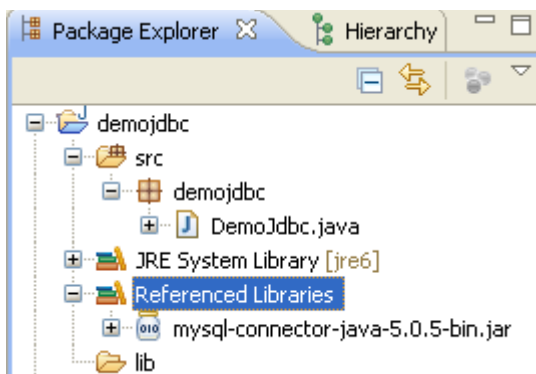
► Cliquez-droit sur le projet->properties. L'écran suivant apparaît.



Dans l'onglet 'Librairies' cliquez sur bouton 'Add Jars' car le driver est déjà dans le projet (Sinon il faudrait cliquer sur Add External Jars)



► sélectionnez le driver puis OK

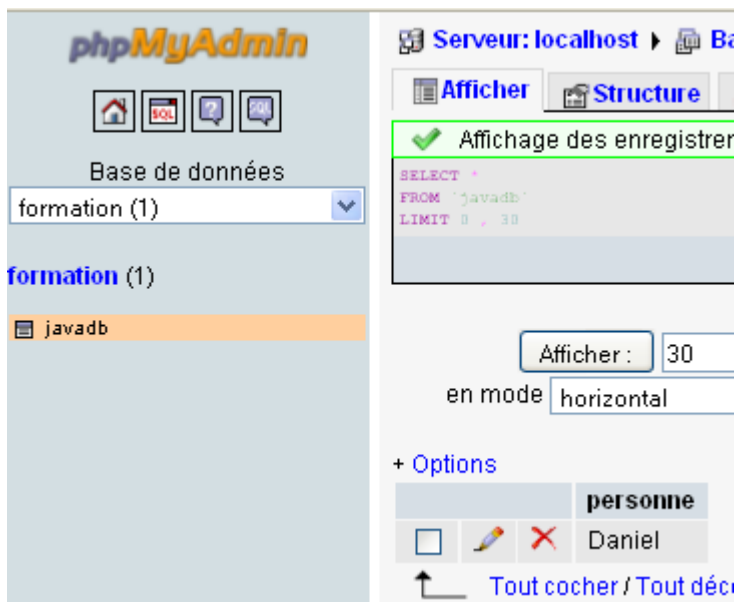


Le driver apparaît désormais dans 'Referenced librairies'.

Partie 3 : création table

En utilisant le client PhpMyAdmin, créer :

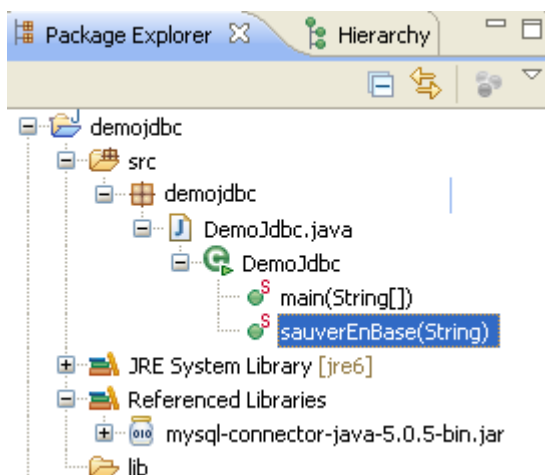
- Une base de données 'formation'
- une table 'javadb' possédant une seule colonne 'personne'



► Expliquez

Partie 4 : insertion de données

Code d'accès aux données



Créez une méthode `sauverEnBase()` suivante, qui prend en paramètre une chaîne de caractère à insérer en base de données.

```

public static void sauverEnBase(String personne) {
    // Information d'accès à la base de données
    String url = "jdbc:mysql://localhost/formation";
    String login = "root";
    String passwd = "";
    Connection cn = null;
    Statement st = null;

    try {
        // Etape 1 : Chargement du driver
        Class.forName("com.mysql.jdbc.Driver");
        // Etape 2 : récupération de la connexion
        cn = DriverManager.getConnection(url, login, passwd);
        // Etape 3 : Création d'un statement
        st = cn.createStatement();
        String sql = "INSERT INTO `javadb` (`personne`) VALUES ('" + personne + "')";

        // Etape 4 : exécution requête
        st.executeUpdate(sql);

    } catch (SQLException e) {
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    } finally {
        try {
            // Etape 5 : libérer ressources de la mémoire.
            cn.close();
            st.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

Cette méthode statique sera appelée à partir de la méthode main() et donc au lancement de l'application.

QUESTION 1 : Dans quel package se trouve Connection ? Est-ce une Classe ou une interface ?

QUESTION 2 : à quoi sert le code : Class.forName ?

QUESTION 3 : à quoi sert le 'try / catch' ?

QUESTION 4 : y a t'il une autre façon de gérer les erreurs potentielles ?

QUESTION 5 : à quoi sert le bloc 'finaly' ?

```

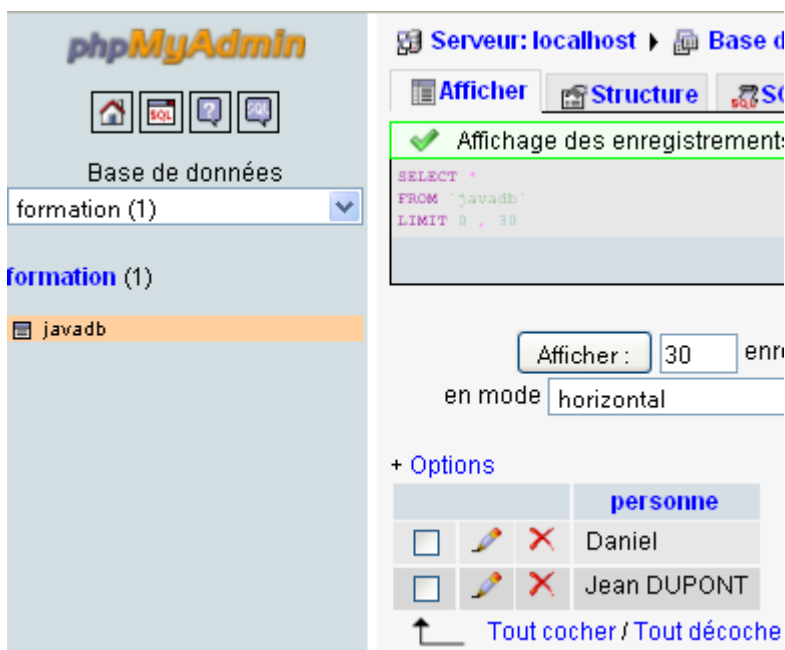
public static void main(String[] args) {

    sauverEnBase("Jean DUPONT");

}

```

► cliquez-droit sur la classe DemoJdbc->Run As->Java Application.



La données a bien été insérée en base de données.

Partie 5 : lecture de données

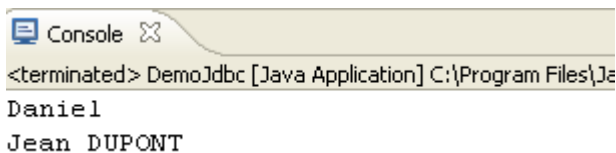
Ci-dessous le code de récupération de données

```

public static void lireEnBase() {
    // Information d'accès à la base de données
    String url = "jdbc:mysql://localhost/formation";
    String login = "root";
    String passwd = "";
    Connection cn = null; Statement st = null; ResultSet rs = null;
    try {
        // Etape 1 : Chargement du driver
        Class.forName("com.mysql.jdbc.Driver");
        // Etape 2 : récupération de la connexion
        cn = DriverManager.getConnection(url, login, passwd);
        // Etape 3 : Création d'un statement
        st = cn.createStatement();
        String sql = "SELECT * FROM javadb";
        // Etape 4 : exécution requête
        rs = st.executeQuery(sql);
        // Etapes 5 (parcours Resultset)
        while (rs.next()) {
            System.out.println(rs.getString("personne"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    } finally {
        try { // Etape 6 : libérer ressources de la mémoire.
            cn.close();
            st.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

Résultat



```

<terminated> DemoJdbc [Java Application] C:\Program Files\Je
Daniel
Jean DUPONT

```

Nous avons récupérés les informations en base de données.

Partie 6 : optimisations : preparedStatement + exceptions

- ▶ Question : qu'est ce qu'un 'PreparedStatement' ? Quelle différence avec un Statement ?
- ▶ Proposez un refactoring du code en utilisant les PreparedStatements
- ▶ Proposez une meilleure stratégie de gestion des exceptions. En particulier, vous pouvez créer un package java 'exceptions' et créer une exception 'DataBaseException' héritant de la classe Exception et dont la méthode getMessage() sera redéfinie pour renvoyer le message 'ERREUR ACCES BASE DE DONNEES'