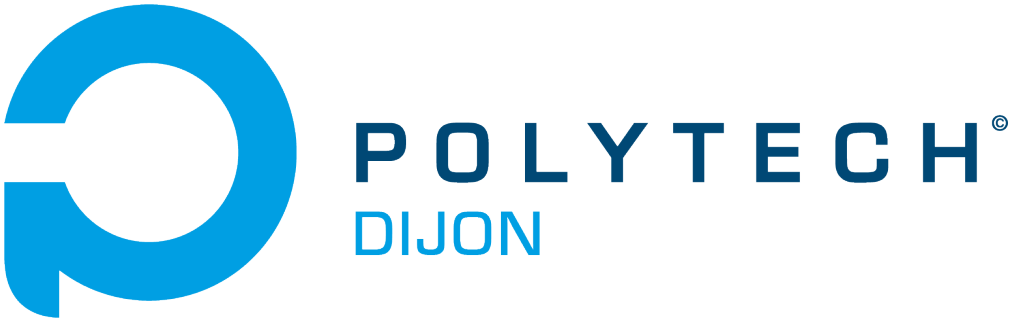
Esirem Informatique & Réseaux

ITC-315 : CR Développement Web

JAVA – BASE DE DONNÉES

Auteur :
BOUQUILLON Erwan

Professeur référent :
CHASSEL Quentin



2023-2024

1 Java-Base de données

Dans le cadre du développement d'applications robustes et performantes, l'utilisation de Java pour se connecter à une base de données s'avère être une compétence essentielle. Une approche efficace pour structurer ce type d'application consiste à créer une architecture qui combine objets métiers et DAO (Data Access Objects). Cette combinaison permet non seulement de simplifier l'accès aux données, mais aussi de favoriser la modularité et la maintenabilité du code. Dans cette optique, nous explorerons les bonnes pratiques et les étapes clés pour intégrer ces concepts dans vos projets Java.

1.1 Connection

On commence donc par réaliser la connexion à la base de donnée MySQL par le biais du driver JDBC. Le code suivant permet ainsi de réaliser la connexion et de récupérer les différentes informations sur un sport de la base de données.

```
    String user = "Erwan";
    String password = "Boubou";

    Class.forName(className:"com.mysql.cj.jdbc.Driver");

    try{
        Connection myConnection = DriverManager.getConnection(
        "jdbc:mysql://localhost:3306/poly_sports",
        user,
        password
        );

        Statement myStatement = myConnection.createStatement();

        ResultSet results = myStatement.executeQuery("SELECT * FROM sport");

        while(results.next()){
            final String sports = results.getString("name");
            final int NumberPart = results.getInt("required_participants");

            System.out.println(sports + " : " + NumberPart + " participants");
        }

    }
    catch(Exception e){
        System.err.println(e.getMessage());
    }
}
```

FIGURE 1 – Test première partie connection

Le texte souligné en rouge ici est dû au fait des modifications apportées dans la suite du programme.

1.2 MySQLDatabase

On va réaliser maintenant une nouvelle classe pour éviter de réaliser tout ça dans le main. Ainsi, on a la classe MySQLDatabase (fournit dans les fichiers sur Github). Pour vérifier le bon fonctionnement, voici ce que l'on a :

```java
//Nouvelle instance de MYSQLDatabase
MYSQLDatabase sport = new MYSQLDatabase(host:"localhost", port:3306, databaseName:"poly_sports", user:"Erwan", password:"Boubou");

sport.connect();

Statement myStatement = sport.createStatement();
ResultSet results = myStatement.executeQuery("SELECT * FROM sport");

while(results.next()){
        final String sports = results.getString("name");
        final int NumberPart = results.getInt("required_participants");

        System.out.println(sports + " : " + NumberPart + " participants");
    }
```

FIGURE 2 – Test MySQL Database

Ainsi après exécution, on récupère à nouveau les informations sur les sports.

1.3 PolySportDatabase

On va à présent créer la classe PolySportsDatabase qui hérite de MySQLDatabase et qui configure automatiquement la connexion vers la base de données. De plus, PolySportsDatabase sera un singleton, ce qui nous assurera que quelque soit le nombre de requêtes que nous ferons dans notre application, seule une connexion vers la base de données sera ouverte.

```java
// Remplacement de MYSQLDatabase par PolySportDatabase
MYSQLDatabase sport = PolySportsDatabase.getInstance();
sport.connect();

Statement myStatement = sport.createStatement();
ResultSet results = myStatement.executeQuery("SELECT * FROM sport");

while(results.next()){
        final String sports = results.getString("name");
        final int NumberPart = results.getInt("required_participants");

        System.out.println(sports + " : " + NumberPart + " participants");
    }
```

FIGURE 3 – Test PolySportDatabase

On va venir créer une instance de database à partir de la classe PolySportDatabase pour réaliser la connexion.

1.4 Sport

Dans la fonction main, on va créer une instance de la classe Sport et afficher la valeur de ces attributs afin de tester le bon fonctionnement. Ainsi, on a :

```java
// Nouvelle instance de sport
Sport football = new Sport(id:14, name:"football", requiredParticipants:26);

System.out.println(football.getId() + " " + football.getName() + " " + football.getRequiredParticipants());
```

FIGURE 4 – Test classe Sport

On va créer un nouveau sport intitulé football avec un id et un un nombre de participants défini par nos soins.

```
PS C:\Users\erwan\Desktop\BDD>  c:; cd 'c:\Users\erwan\Desktop\BDD'; & 'C:\Program Files\Java\jdk-22\bin\java.exe'
cal\Temp\cp_8qakvvm0ey89v3gt0oo8k4406.argfile' 'App'
14 football 26
```

FIGURE 5 – Résultat classe Sport

1.5 SportsDAO

Jusque-là on a d'un côté la classe PolySportsDatabase qui gère la connexion avec la base de données et la classe Sport qui représente un sport. On va à présent créer une classe qui fera le lien entre l'objet métier (Sport) et la base de données : un Data Access Object, plus communément appelé DAO. On implémente la méthode findAll() qui permettera d'afficher tous les sports de la base de données.

```
PS C:\Users\erwan\Desktop\BDD>  c:; cd 'c:\Users\erwan\Desktop\BDD'; & 'C:\Program Files\Java\jdk-22\bin\java.exe'
cal\Temp\cp_8qakvvm0ey89v3gt0oo8k4406.argfile' 'App'
1 Badminton (simple) 2
2 Badminton (double) 4
3 Basket 10
```

FIGURE 6 – Test findAll()

On remarque que la base de données est composé de 3 sports (badminton (simple et double) et basket).

On va dans la suite, implémenter une nouvelle méthode findbyId() qui permettera de rechercher le sport en fonction d'un id fourni par l'utilisateur. Après implémentation, on peut réaliser un test avec l'id = 3 qui correspond au basket :

```
PS C:\Users\erwan\Desktop\BDD>  c:; cd 'c:\Users\erwan\Desktop\BDD'; & 'C:\Program Files\Java\jdk-22\bin\java.exe'
cal\Temp\cp_8qakvvm0ey89v3gt0oo8k4406.argfile' 'App'
Sport trouvé : Basket, Nombre de joueurs requis : 10
```

FIGURE 7 – Test finbyId()

Cependant, si on fourni un id qui n'est pas dans la base de données, on renvoie un message d'erreur indiquant que l'id ne correspond à aucun sport.

```
PS C:\Users\erwan\Desktop\BDD>  c:; cd 'c:\Users\erwan\Desktop\BDD'; & 'C:\Program Files\Java\jdk-22\bin\java.exe'
cal\Temp\cp_8qakvvm0ey89v3gt0oo8k4406.argfile' 'App'
Aucun sport trouvé pour l'ID donné.
```

FIGURE 8 – Test findbyId()

On va dans la suite, implémenter une nouvelle méthode findbyName() qui permettera de rechercher le sport en fonction d'un nom ou d'un morceau de chaine de caractère fourni par l'utilisateur. Après implémentation, on peut réaliser un test avec la chaine de caractère 'bad' qui nous renverra les 2 badmintons :

```
PS C:\Users\erwan\Desktop\BDD>  c:; cd 'c:\Users\erwan\Desktop\BDD'; & 'C:\Program Files\Java\jdk-22\bin\java.exe'
cal\Temp\cp_8qakvvm0ey89v3gt0oo8k4406.argfile' 'App'
Recherche de sports contenant le mot 'bad' :
Sport trouvé : Badminton (double), Nombre de joueurs requis : 4
Sport trouvé : Badminton (simple), Nombre de joueurs requis : 2
```

FIGURE 9 – Test findbyName()

Cependant si l'on saisit la chaine de caractère 'bas', on renvoie le sport basket :

```
PS C:\Users\erwan\Desktop\BDD>  c:; cd 'c:\Users\erwan\Desktop\BDD'; & 'C:\Program Files\Java\jdk-22\bin\java.exe'
cal\Temp\cp_8qakvvm0ey89v3gt0oo8k4406.argfile' 'App'
Veuillez saisir le nom du sport à rechercher :
bas
Sport trouvé : Basket, Nombre de joueurs requis : 10
```

FIGURE 10 – Test 2 findbyName()

1.6 PreparedStatement

On va laisser JDBC intégrer les données à votre requête. Pour cela, on va utiliser des PreparedStatement qui prennent en paramètre la requête à exécuter dans laquelle toutes les données ont été remplacées par des point d'interrogation.

```java
public PreparedStatement preparedStatement(String query){
    try{
        PreparedStatement statement = connection.prepareStatement(query);
        return statement;
    }
    catch(SQLException e){
        System.err.println(e.getMessage());
    }
    return null;
}
```

FIGURE 11 – Implémentation de Prepared

On va donc maintenant modifier les fonctions findbyName() et findbyId() à partir de la méthode Prepared implémenté précédemment.

```java
public Sport findById(int idbis){
    try{
        String query = "SELECT * FROM sport WHERE id = ? ;";
        PreparedStatement myStatement = database.preparedStatement(query);
        myStatement.setInt(parameterIndex:1, idbis);
        ResultSet results = myStatement.executeQuery();
        while(results.next())
        {
            final String name = results.getString(columnLabel:"name");
            final int id = results.getInt(columnLabel:"id");
            final int requiredParticipants = results.getInt(columnLabel:"required_participants");

            Sport ajout = new Sport(id, name, requiredParticipants);
            return ajout;
        }

    }

    catch(Exception e){
        System.err.println(e.getMessage());
        return null;
    }
    return null;
}
```

FIGURE 12 – Implémentation de findbyId()

```java
//Page 9 findbyName avec preparedStatement
public ArrayList<Sport> findByName(String Name){
    ArrayList<Sport> sport = new ArrayList<Sport>();
    String query = "SELECT * FROM sport WHERE name LIKE ? ORDER BY name;";
    try{
        PreparedStatement myStatement = database.preparedStatement(query);
        myStatement.setString(parameterIndex:1, "%" + Name + "%");
        ResultSet results = myStatement.executeQuery();

        while(results.next())
        {
            final String name = results.getString(columnLabel:"name");
            final int id = results.getInt(columnLabel:"id");
            final int requiredParticipants = results.getInt(columnLabel:"required_participants");

            Sport ajout = new Sport(id, name, requiredParticipants);
            sport.add(ajout);
        }
        return sport;
    }
    catch(Exception e){
        System.err.println(e.getMessage());
        return null;
    }
}
```

FIGURE 13 – Implémentation de findbyName

On va mettre au point les méthodes permettant d'ajouter, de mettre à jour et de supprimer. Ainsi, après avoir implémenté la méthode insert, on ajoute le sport rugby avec l'id 10 et un nombre de participants requis de 30.

1.7 Insert

```
PS C:\Users\erwan\Desktop\BDD>  c:; cd 'c:\Users\erwan\Desktop\BDD'; & 'C:\Program Files\Java\jdk-22\bin\java.exe'
cal\Temp\cp_8qakvvm0ey89v3gt0oo8k4406.argfile' 'App'
Le sport a été inséré avec succès.
```

FIGURE 14 – Insertion dans la base de données

On aperçoit bien le sport rugby qui est apparu dans la table sport.

		id	name	required_participants
☐	Éditer Copier Supprimer	1	Badminton (simple)	2
☐	Éditer Copier Supprimer	2	Badminton (double)	4
☐	Éditer Copier Supprimer	3	Basket	10
☐	Éditer Copier Supprimer	10	Rugby	30

FIGURE 15 – Vérification dans la base de données

1.8 Update

```
PS C:\Users\erwan\Desktop\BDD>  c:; cd 'c:\Users\erwan\Desktop\BDD'; & 'C:\Program Files\Java\jdk-22\bin\java.exe'
cal\Temp\cp_8qakvvm0ey89v3gt0oo8k4406.argfile' 'App'
Le sport a été mis à jour avec succès.
```

FIGURE 16 – Mise à jour dans la base de données

On aperçoit bien le sport rugby qui est apparu dans la table sport mais avec un nombre de participants différent.

		id	name	required_participants
☐	Éditer Copier Supprimer	1	Badminton (simple)	2
☐	Éditer Copier Supprimer	2	Badminton (double)	4
☐	Éditer Copier Supprimer	3	Basket	10
☐	Éditer Copier Supprimer	10	rugby	10

FIGURE 17 – Vérification dans la base de données

1.9 Delete

```
PS C:\Users\erwan\Desktop\BDD>  c:; cd 'c:\Users\erwan\Desktop\BDD'; & 'C:\Program Files\Java\jdk-22\bin\java.exe'
cal\Temp\cp_8qakvvm0ey89v3gt0oo8k4406.argfile' 'App'
Le sport a été supprimé avec succès.
```

FIGURE 18 – Insertion dans la base de données

On supprime l'id 2 qui correspond au badminton (double) et ainsi dans notre table sport, on remarque que le badminton double a bien été retiré de la base de données :

←T→			id	name	required_participants
☐	Éditer	Copier Supprimer	1	Badminton (simple)	2
☐	Éditer	Copier Supprimer	3	Basket	10
☐	Éditer	Copier Supprimer	10	rugby	10

FIGURE 19 – Vérification dans la base de données