

Java DataBase Connectivity (JDBC)

Dans ce tutoriel, vous allez apprendre à utiliser JDBC (Java DataBase Connectivity). JDBC est une bibliothèque Java qui permet d'accéder à des bases de données par le biais d'une interface commune.

Nous allons utiliser une base de données MySQL hébergée sur un serveur de l'ENSMM dont le nom DNS est `nemrod.ens2m.fr`. Cette base porte le nom `tp_jdbc`. Son accès est possible à l'aide de l'identifiant `etudiant` et du mot de passe `YTDVvj9TR3CDYcMf`.

Cette base contient deux tables, l'une contenant des joueurs, l'autre des objets. Le contenu de ces tables est librement inspiré du jeu PokémonGo. La table des joueurs enregistre les joueurs avec leurs identifiants (`pseudo`, `email` et `motDePasse` chiffré en MD5), leurs coordonnées (en degrés décimaux) et la date de la dernière connexion (`date` et `heure`).

#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra
<input type="checkbox"/> 1	pseudo	varchar(20)	utf8_general_ci		Non	Aucune	
<input type="checkbox"/> 2	email	varchar(100)	utf8_general_ci		Non	Aucune	
<input type="checkbox"/> 3	motDePasse	varchar(32)	utf8_general_ci		Non	Aucune	
<input type="checkbox"/> 4	latitude	double			Non	Aucune	
<input type="checkbox"/> 5	longitude	double			Non	Aucune	
<input type="checkbox"/> 6	dateDerniereConnexion	datetime			Non	Aucune	

La table des objets liste les pokémons avec leur description, leurs coordonnées, leur visibilité, la date de leur ajout dans la table et leur propriétaire éventuel (`pseudo` d'un joueur ou `admin` si non attribué).

#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra
<input type="checkbox"/> 1	id	int(11)			Non	Aucune	AUTO_INCREMENT
<input type="checkbox"/> 2	description	varchar(200)	utf8_general_ci		Non	Aucune	
<input type="checkbox"/> 3	latitude	double			Non	Aucune	
<input type="checkbox"/> 4	longitude	double			Non	Aucune	
<input type="checkbox"/> 5	visible	tinyint(1)			Non	Aucune	
<input type="checkbox"/> 6	dateCreation	datetime			Non	Aucune	
<input type="checkbox"/> 7	proprietaire	varchar(20)	latin1_general_ci		Non	admin	

L'accès à une base de données avec JDBC se fait en quatre étapes :

- Connexion à la base de données
- Préparation d'une requête
- Envoi de la requête
- Récupération et exploitation du résultat

I Se connecter à une base de données

La première étape consiste à se connecter à la base de données. JDBC fournit une classe bien pratique pour cela, la classe `DriverManager`. Pour établir la connexion, il suffit de construire un objet `DriverManager` en précisant l'adresse complète de la base, un identifiant et un mot de passe.

Dans notre cas, l'adresse complète est constituée du préfixe `jdbc:mysql://`, suivi du nom DNS du serveur `nemrod.ens2m.fr`, puis du numéro du port `:3306`, et enfin du nom de la base `/tp_jdbc`, ce qui donne :

```
Connection connexion = DriverManager.getConnection("jdbc:mysql://nemrod.ens2m.fr:3306/tp_jdbc\n?serverTimezone=UTC", "etudiant", "YTDVj9TR3CDYcMP");
```

Une fois la connexion établie, vous pouvez lancer une ou plusieurs requêtes. Il faut ensuite fermer la connexion avec :

```
connexion.close();
```

Remarque

Il est fortement recommandé de limiter le nombre de connexions. Le mieux est de n'ouvrir qu'une seule connexion et de faire toutes les requêtes avec cette unique connexion.

II Consulter une table – SELECT

Nous allons maintenant voir comment réaliser une requête de type `SELECT`. Il faut d'abord préparer la requête à l'aide d'une connexion précédemment ouverte. Par exemple, si vous voulez récupérer la liste des pseudos des joueurs avec leurs coordonnées, vous pouvez écrire la requête suivante :

```
PreparedStatement requete = connexion.prepareStatement("SELECT _pseudo, _latitude, _longitude_\nFROM _joueur;");
```

Il suffit ensuite d'exécuter la requête :

```
ResultSet resultat = requete.executeQuery();
```

Si la requête ne comporte pas d'erreur, on récupère le résultat dans un objet `ResultSet`. La méthode `next` permet d'accéder aux lignes successives. Cette méthode renvoie `false` si il n'y a plus de ligne à traiter. On récupère ensuite les valeurs des différents champs à l'aide des méthodes `getString`, `getDouble`, `getBoolean` ou encore `getDate` en précisant le nom du champ désiré.

```
while (resultat.next()) {\n    String pseudo = resultat.getString("pseudo");\n    double latitude = resultat.getDouble("latitude");\n    double longitude = resultat.getDouble("longitude");\n    System.out.println(pseudo + " _ (" + latitude + " ; _ " + longitude + " )");\n}
```

Une fois les résultats traités, vous devez fermer la requête.

```
requete.close();
```

- 1 Ouvrez le projet Netbeans `TP_JDBC` et exécutez le fichier `TestSelect.java`.
- 2 En vous inspirant de `TestSelect`, écrivez un programme listant les objets de la table des objets avec toutes leurs caractéristiques. Le nom de la table des objets est `objet`.

Pour mettre au point, une requête SELECT sans avoir à personnaliser le code de traitement du résultat, vous pouvez utiliser la méthode maison `OutilsJDBC.afficherResultSet(resultat)` qui affiche tous les résultats dans la console quels qu'ils soient.

3 Utilisez le fichier *TestSelectAll.java* pour tester quelques requêtes SELECT.

III Modifier une donnée – UPDATE

La préparation d'une requête UPDATE est similaire à celle d'une requête SELECT. Par exemple, la requête suivante vous permet de modifier les coordonnées d'un joueur dans la table des joueurs.

```
PreparedStatement requete = connexion.prepareStatement("UPDATE_joueur_SET_latitude=?,_longitude=?,_WHERE_pseudo=?");
```

On remarque la présence de points d'interrogation qui permettent de préciser les données dans un second temps à l'aide de méthodes spécifiques qui vont les formater aux standards de SQL.

```
requete.setDouble(1, 47.25097);  
requete.setDouble(2, 5.99432);  
requete.setString(3, "aurore");
```

Remarque

Les numéros désignent les points d'interrogation dans l'ordre d'apparition dans la requête. Attention, contrairement aux indices de Java, les indices de SQL commencent à 1 !

Une fois les données explicitées, vous pouvez exécuter la requête mais comme cette requête va modifier la base, il faut utiliser la méthode `executeUpdate` et non `executeQuery`.

```
requete.executeUpdate();
```

4 Ouvrez et testez *TestUpdate.java*

5 Écrivez un programme similaire permettant de modifier le propriétaire d'un objet dans la table des objets.

6 Vérifiez le fonctionnement de votre programme à l'aide du programme *TestSelectAll*.

IV Ajouter des données – INSERT

L'utilisation d'une requête INSERT suit la même logique que les exemples précédents. L'exemple suivant vous montre comment préparer et exécuter une requête permettant d'ajouter un joueur dans la table des joueurs.

```
PreparedStatement requete = connexion.prepareStatement("INSERT INTO_joueur_VALUES_(?,?,?,?,"  
NOW())");  
requete.setString(1, "pierre");  
requete.setString(2, "pierre@ens2m.fr");  
requete.setString(3, OutilsJDBC.MD5("onix"));  
requete.setDouble(4, 47.25097);  
requete.setDouble(5, 5.99432);  
requete.executeUpdate();
```

La méthode `OutilsJDBC.MD5("onix")` permet de chiffrer le mot de passe en MD5 dans la requête pour ne pas le transmettre en clair.

7 Ouvrez et testez *TestInsert.java*

8 Écrivez un programme similaire permettant d'ajouter un objet dans la table des objets.

V Supprimer des données – DELETE

Pour supprimer des données avec une requête DELETE, la procédure est identique. Par exemple, pour supprimer un joueur connu par son nom, vous pouvez écrire :

```
PreparedStatement requete = connexion.prepareStatement("DELETE FROM _joueur WHERE _pseudo=_?")
;
requete.setString(1, "pierre");
requete.executeUpdate();
```

Attention !

Un requête DELETE mal faite peut détruire tout ou partie de la table ! Testez votre requête préalablement en remplaçant DELETE par SELECT.

9 Ouvrez et testez *TestDelete.java*

10 Écrivez un programme similaire permettant de supprimer un objet connu par son identifiant dans la table des objets.

VI Mise en pratique

Pour finir ce tutoriel, nous vous proposons des exercices permettant d'expérimenter les requêtes les plus courantes.

11 Écrivez un programme permettant de récupérer la liste des descriptions des objets distincts (liste sans doublons).

12 Écrivez un programme permettant de récupérer la liste des objets attribués à un joueur (connu par son pseudo).

13 Écrivez un programme permettant de savoir si un couple pseudo/mot de passe existe dans la table des joueurs.

14 Écrivez un programme permettant de remplacer la date de connexion d'un joueur par la date du serveur.

15 Écrivez un programme permettant de récupérer à l'aide d'une jointure la liste des couples joueur.pseudo / objet.description rangée par ordre alphabétique des pseudos.

16 Écrivez un programme permettant de récupérer le nombre d'objets possédés par chaque joueur.

17 Écrivez un programme permettant de lister les objets qui ont une distance à un joueur inférieure à 0.001° en latitude et en longitude.