Java cours 14

JDBC (java database connectivity)

**Diapo 1**

JDBC c’est pour les bases de données relationnelles. Un API java pour accéder au BD.

JDBC, est indépendant du SGBD avec lequel il travaille (il y a tt de même des petites différences).

**Diapo 2**

C’est basé sur des interfaces définissant les protocoles de communication entre notre machine et le serveur distant ayant la base de données.

Accès au méta-modèle, le schéma de la BD (les tables, les titres de colonnes …)

Ce sont des interfaces comme cela c’est indépendant de l’implémentation (on ne verra pas de différence suivant le SGBD utilisé).

**Diapo 4**

Pilote SGDB c’est les ensembles de classe correspondant au SGBD que l’on utilise. Il faut donc les bons drivers pour le bon SGBD (pilote est dépendant de la version aussi).

**Diapo 12**

Javax.sql très utile pour le travail avec les serveurs web.

**Diapo 13**

Java.sql contient quelques classes

DriverManager, charge et configure le driver pour se connecter avec un SGBD.

L’objet connexion permet de faire des requetés, les objets de requetés (statement) permettent d’obtenir les résultats (ResultSet, ResultSetMetaData).

**Diapo 16**

Avec le Class.forName on a besoin d’une chaine de caractère. Cette chaine n’est pas en dur dans le code. Mais va aller chercher dans un fichier .config (ou .properties) pour charger le bon SGBD sans à avoir changer le code. (Ceci on ne le fait plus c’est géré directement par le serveur).

**Diapo 18**

Il faut fermer une connexion quand on en a plus besoin car le nombre de connexion pour un SGBD est limité. Pour être sur de bien fermer la connexion il faut mettre un bloc finally pour la méthode close(). Attention la variable conn est connue que dans son bloc try, on ne la connait pas dans le bloc finally. Le close() peut aussi envoyer une exception donc il faut try catch

La variable conn est déclarée en dehors du bloc try comme ça tous les blocs sous-jacent la connaisse.

Clause finally avec JDBC, c’est vraiment lourd pour pas grand-chose. Utilisation du try-with-ressources, c’est bcp plus propre.

Quand un programme java est fermé il ferme toutes les connexions. Mais attention si sur un serveur, il faut alors bien fermer les connexions une fois qu’elles ne sont plus utilisées.

**Diapo 23**

ExecuteQuery quand on fait un select

ExecuteUpdate pour les formes qui ne retourne pas de résultats (insert, delet, update …).

Execute si on ne sait pas ce qui va être fait, select ou autre. Execute renvoie un booleen qui indiquera si on a fait un select ou une autre commande.

**Diapo 24**

Un statement est initié à partir de la connexion, on récupère l’objet ResultSet ayant exécuté l’executeQuery sur l’objet statement.

Statement et ResultSet sont des objets closables, donc on peut faire des try avec ressources sur ces objets. On aura ainsi une fermeture automatique car on passe par un finally implicite qui ferme les connections.

**Diapo 25**

Méthode next() permet de passer à la ligne suivante, et renvoie un booleen. Donc permet de rester dans le while tant qu’il y a des lignes. Toujours réaliser un next() pour accéder aux lignes du ResultSet.

Dans le cas où il y a une seule ligne on peut faire un if au lieu du while.

**Diapo 26**

La méthode get permet de récupérer les informations dans la table et les mémoriser dans des variables.

**Diapo 27**

On peut faire un getString quel que soit le type de départ. Les autres get doivent être du même type que les éléments de la colonne, sinon on aura une SQLException.

**Diapo 32**

PreparedStatement, une requête avec des points d’interrogation qui indique où est ce qu’il y a du code variable. Utilisation de méthode setString(), setInt() … les méthodes set demande deux paramètres. Le premier est pour localiser quel point d’interrogation il faut remplacer, le deuxième c’est par quoi on remplace le « ? ». Ainsi on ne peut plus faire d’injection SQL.

L’avantage preparedStatement, évite injection SQL, on peut exécuter une requête en boucle et comme la requête a été donnée au préalable, le SGBD a déjà pu optimiser une partie de la requête.

Dès qu’il y a injection de code dans une requête, il faut passer par un PreparedStatement.

TP JDBC

Pour la récupération des infos du driver de la connexion, du pseudo, et du mdp. Soit on utilise un fichier properties, soit on utilise des classes pour le faire en console. Méthode readPassword, pas d’echo donc on ne voit pas en claire le message tapé. Remarque, l’API console n’est pas géré par netbeans (eclipse non plus).

Clean sur un projet, cela nettoie la partie build, il faut tout recompiler. Dans le repertoire dist, on retrouve notre .jar du programme. Et en plus les .jar des BD que l’on utilise (ojdbc7.jar).

Dans les méthodes finales, on trhows l’exception SQL et elle sera gérée dans le main à l’endroit où cette méthode est appelée.

Attention quand on utilise dans la requête une information rentrée par l’utilisateur, il y a un risque d’injection SQL.

Pour se connecter au vpn : vpn.grenet.fr