

JDBC

JDBC¹ est une API Java permettant à un programme Java de communiquer avec une base de données via le langage SQL. Familiarisons-nous avec cet API.

1 Vue d'ensemble

Nous n'allons pas tout expliquer ici mais plutôt vous donner une première vue d'ensemble après quoi nous vous renvoyons vers le tutoriel de SUN pour les détails.

- ✓ **JDBC** (Java Data Base Connectivity) est une API qui permet de se connecter à des bases de données et de les interroger via des requêtes SQL. Pour cela, il faudra disposer d'un driver JDBC adapté au système de base de données utilisé.
- ✓ **ODBC** (Open Data Base Connectivity) est un protocole uniformisant le dialogue avec des systèmes de base de données. JDBC offre en standard un driver JDBC pour des SGDB que l'on peut présenter en tant que Sgbd Odbc (càd pour lesquels on dispose d'un driver Odbc) . Ainsi vous pourrez tester vos exemples avec une base Access par exemple pour autant que vous l'enregistriez dans ODBC.
- ✓ Dans votre code Java, vous devrez « **charger** » le **bon driver** avant toute opération d'accès à un SGDB. Voyez le tutoriel pour la syntaxe précise.

Sachez que pour une connexion via **ODBC**

- Le driver est : "sun.jdbc.odbc.JdbcOdbcDriver"
- Le nom de la BD est : "jdbc:odbc:OdbcNom"

et pour un accès via un driver natif **Oracle**

- Le driver est : "oracle.jdbc.driver.OracleDriver"
- Le nom de la BD est : "jdbc:oracle:thin:@oracle9:1521:instance"
(où vous aurez indiqué le nom de votre BD en lieu et place de « instance ». Dans notre cas il s'agit d' « esidb ».)
- Il ne faut pas oublier de rendre accessible le jar du driver :
"C:\ORAN9\jdbc\lib\classes111.jar"

- ✓ Une fois le driver chargé, il faut **créer une connexion** à une BD précise. A nouveau, voyez le tutoriel pour le détail.

¹ Actuellement, de nouvelles API permettant de gérer la persistance des données voient le jour mais elles reprennent JDBC. Nous les aborderons l'année prochaine.

- ✓ Un **Statement** est un objet Java qui représente une requête SQL. Il y a 2 étapes : premièrement créer un tel objet; deuxièmement lancer la requête en spécifiant le code SQL.
- ✓ Alors que les requêtes de gestion de la table renvoient un entier indiquant le nombre de lignes modifiées, les requêtes de type « SELECT » retournent la réponse dans un objet de type **ResultSet**. Cette objet fournit toutes les méthodes nécessaires pour examiner les différentes lignes du résultat et prendre connaissance des valeurs.
- ✓ Un **PreparedStatement** diffère d'un Statement par 2 aspects. Premièrement, l'ordre SQL est spécifié lors de sa création ce qui permet de l'envoyer directement pour compilation au SGBD. Deuxièmement, il peut contenir des paramètres. Vous pourrez ainsi réutiliser la même requête plusieurs fois en spécifiant à chaque fois les valeurs à donner aux paramètres. De plus, c'est l'API qui se chargera de la conversion des paramètres (nous n'avons plus, par exemple, à nous occuper de doubler les ' dans les chaînes de caractères).
- ✓ JDBC prévoit l'utilisation des **transactions**. Par défaut, l'API fonctionne en mode **auto-commit** où chaque ordre SQL est traité comme une transaction séparée. Ce comportement peut-être modifié via la méthode *setAutoCommit()* d'une connexion. Voyez aussi les méthodes *commit()* et *rollback()*.
- ✓ Il est également possible de définir et utiliser des **procédures stockées**. Comme vous le savez, l'ordre SQL précis varie d'un SGBD à l'autre

2 Apprentissage

Plutôt que de récrire un tutoriel d'apprentissage, nous préférons vous renvoyer sur celui de SUN qui est bien fait (<http://java.sun.com/docs/books/tutorial/jdbc/basics/index.html>). Lisez et testez jusqu'à 'Using Transactions'.

La base 'CoffeeBreak' vous est fournie en annexe sous Access ainsi que le script de création des deux tables pour Oracle.

3 Exercice

Pour mettre en oeuvre de manière simple les éléments découverts dans le tutoriel précédent, il vous est demandé de réaliser une application 'console' réalisant les différentes actions suivantes:

- affichage de la liste des fournisseurs
- augmentation de 1 unité du prix du café 'Colombian'
- ajout des cafés
 'MonCafé', 101, 8.5, 0, 0
 'TonCafé', 523, 10.2, 0, 0

Attention, l'ajout du deuxième tuple ne peut pas se réaliser puisque le fournisseur 523 n'existe pas: veillez à présenter un message de non réalisation de l'ajout.

- Affichage de la liste des cafés dont le prix est inférieur à 9 unités.

Testez votre programme avec Access et Oracle.