Bases de Données : Java DataBase Connectivity

Stéphane Devismes

Université Grenoble Alpes

26 août 2020

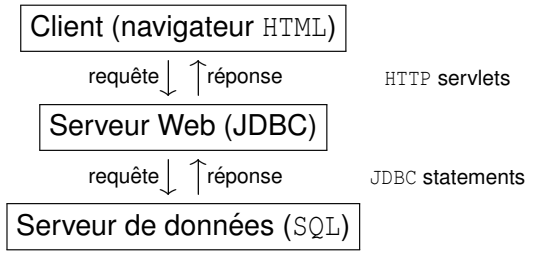
Plan

Intégration SQL

Aujourd'hui, nous allons étudier comment utiliser l'application JDBC et les servlets HTTP pour accéder à une base de données SQL à partir d'une interface web, **afin d'interroger la base de données ou de modifier ses données.**

Organisation à 3 niveaux (≪ three-tier ≫)

- ≪ En haut ≫ : niveau présentation. Typiquement, une interface web : un navigateur (HTML) sur la machine client.
- ≪ En bas ≫ : niveau données. La base de données sur le serveur de bases de données (SGBD : SQL).
- ≪ Au milieu ≫ : niveau logique. Traduction entre les 2 autres niveaux.



Niveau logique

Traduction entre les 2 autres niveaux sur un serveur web.

La communication entre le client et le serveur web est assurée par le protocole de transfert HTTP.

Des servlets permettent de créer dynamiquement des données au sein du serveur HTTP. Ces données sont généralement présentées au format HTML, ou XML, ou autre.

Une servlet s'exécute dynamiquement sur le serveur web et permet l'extension des fonctions de ce dernier : accès à des bases de données, transactions d'e-commerce, *etc.*

Pilote

Des pilotes (drivers) JDBC sont disponibles pour tous les systèmes de bases de données relationnelles.

Par exemple `oracle.jdbc.OracleDriver` pour Oracle.

Le pilote est utilisé pour créer une connexion à la base de données. L'appel à `Class.forName` déclenche un chargement dynamique du pilote et l'enregistre dans la classe `DriverManager`, qui gère la connection à la base de donnée.

Schéma de programmation

JDBC (Java DataBase Connectivity) est une interface de programmation (API, Application Programming Interface) qui permet à des programmes Java d'accéder à une base de données SQL.

JDBC fournit des méthodes pour interroger et mettre à jour les données. JDBC procède en quatre temps :

1. Connexion à la base de données : `Connection conn = ...;`
2. Ouverture d'une « instruction » (statement) : `Statement st = ...; ...;`
3. Interrogation de la base de données : `ResultSet rs = ...; ...;` puis **utilisation des résultats retournés**.

   (Pour une modification de la base de données cette troisième phase est modifiée.)
4. Fermeture et déconnection : `st.close(); rs.close(); conn.close();`

Pilote : exemple

```
import java.sql.*;
public class ExempleJDBC
{
public static void main(String[] Args)
    {
    try {
        Class.forName("oracle.jdbc.OracleDriver");
        }
    catch (Exception e) {
        // affichage dans la console
        System.err.println("Pas de pilote !");
        }
    ... connexion et utilisation de la base ...
    }
}
```

ResultSet

Un objet de la classe `ResultSet` (ensemble de résultats) permet de récupérer le résultat d'une requête sous la forme d'une table.

Il y a plusieurs méthodes dans la classe `ResultSet` pour parcourir un ensemble de résultats, en particulier la méthode `next()` (un itérateur) permet d'accéder à chaque ligne de la table résultat et retourne ≪ faux ≫ lorsque la fin de la table est atteint.

Pour récupérer les valeurs des attributs on utilise les méthodes `getString(p)`, `getInt(p)`, `getFloat(p)`, *etc*, de la classe `ResultSet`, où `p` est la position (à partir de 1) de l'attribut dans le `SELECT`.

Interrogation : exemple (1/2)

```
String url = "jdbc:oracle:thin:@im2ag-oracle.e.ujf-grenoble.fr:1521:tptomcat";
String login = "login", pwd = "mot de passe";

Connection conn = DriverManager.getConnection(url,login,pwd);
Statement st = conn.createStatement();

ResultSet rs = st.executeQuery("SELECT Nom,Prenom,Age FROM personne ORDER BY age");

while (rs.next()) {
     System.out.println("Nom    : " + rs.getString(1));
     System.out.println("Prénom : " + rs.getString(2));
     System.out.println("Age    : " + rs.getString(3));
}

rs.close();
st.close();
conn.close();
```


Interrogation : exemple (2/2)

```
Connection conn;
String url = "jdbc:oracle:thin:@im2ag-oracle.e.ujf-grenoble.fr:1521:tptomcat";
String login = "login", pwd = "mot de passe";

String Requete = "select nocage from LesGardiens where nome = 'Scholl'";

conn = DriverManager.getConnection(url,login,pwd);
Statement st = conn.createStatement();
ResultSet rs = st.executeQuery(Requete);

System.out.println("Cages gardées par Scholl :");
while(rs.next())
     System.out.println(rs.getString(1));

rs.close();
st.close();
conn.close();
```


Interrogation : avec traitement des exceptions

```
Connection conn;
String url = "jdbc:oracle:thin:@im2ag-oracle.e.ujf-grenoble.fr:1521:tptomcat";
String login = "login", pwd = "mot de passe";
String Requete = "select nocage from LesGardiens where nome = 'Scholl'";

try{
     conn = DriverManager.getConnection(url,login,pwd);
     Statement st = conn.createStatement();
     ResultSet rs = st.executeQuery(Requete);

     System.out.println("Cages gardées par Scholl :");
     while(rs.next()) {
          System.out.println(rs.getString(1));
     }

     rs.close();
     st.close();
     conn.close();
}
catch(SQLException se){
     System.err.println("Erreur SQL");
     JDBCTutorialUtilities.printSQLException(se);
}
catch(Exception e){
     System.err.println("Message d'erreur");
     e.printStackTrace();
}
```

Modification

Pour les modifications d'une base de données (`INSERT, UPDATE, DELETE`), on utilise la fonction `executeUpdate` de la classe `Statement`.

Chaque modification des données retourne un entier : c'est le nombre de lignes affectées par la modification.

Utiliser les quotes doubles et le + pour manipuler les chaînes en `Java`.

Attention aux quotes : doubles pour `Java`, simples pour `SQL`.

Lorsque `nom` vaut `Marie` et `age` vaut `25`, où `Marie` et `25` sont des chaînes pour `Java`, alors la chaîne `Java`

```
"VALUES ('" + nom + "', " + age + ")"
```

est formée de la concaténation de `VALUES ('` puis `Marie` puis `',` puis `25` puis `)`, ce qui donne le morceau de code `SQL` :

```
VALUES ('Marie', 25)
```


Ajout : exemple

```
String url = "jdbc:oracle:thin:@im2ag-oracle.e.ujf-grenoble.fr:1521:tptomcat";
String login = "login", pwd = "mot de passe", Nom = "Marie", Age = "25";

Connection conn = DriverManager.getConnection(url,login,pwd);
Statement st = conn.createStatement();

int nb = st.executeUpdate("INSERT INTO personne(Nom,Age) "
        + "VALUES ('" + nom + "', " + age + ")");

System.out.println(nb + " ligne(s) insérée(s)");

st.close();
conn.close();
```


`JDBC` et transactions

Par défaut : `auto-commit`

Sinon

```
String url = "jdbc:oracle:thin:@im2ag-oracle.e.ujf-grenoble.fr:1521:tptomcat";
String login = "login", pwd = "mot de passe", Nom = "Marie", Age = "25";

Connection conn = DriverManager.getConnection(url,login,pwd);
conn.setAutoCommit(false);
try {
      ...
      conn.commit();
    }
catch (SQLException e ) {
      ...
      conn.rollback();
    }
finally {
      conn.close();
}
```


Accès à une base *via* le Web

Ce qui précède doit être utilisé avec `Java` : compilation puis exécution.

Pour y accéder directement depuis une page web, via des pages HTML, on peut utiliser des servlets `HTTP`.

`Package javax.servlet.http`

Les servlets de `Java` sont utilisées pour le développement de pages web. Typiquement, elles permettent à un serveur d'engendrer des pages HTML en utilisant les réponses à des requêtes `SQL` formulées par un client.

Post et Get

La communication entre le client et le serveur web utilise les **requêtes de lecture** GET et POST (identique à GET mais plus sûre) de HTTP.

HTTP **(HyperText Transfer Protocol)** est un protocole de requête-réponse entre un client et un serveur. Typiquement, le serveur est un serveur web et le client est un navigateur. Le client soumet des requêtes, auxquelles le serveur répond.

HttpServlet

Une servlet est une classe Java qui traite des requêtes HTTP.

Une servlet est sous-classe de la classe abstraite HttpServlet créée pour former une servlet HTTP pour un site Web.

Cette sous-classe implante les méthodes :

- `doGet(HttpServletRequest req, HttpServletResponse res)` pour les requêtes GET de HTTP,
- `doPost(HttpServletRequest req, HttpServletResponse res)` pour les requêtes POST de HTTP.

Protocole ≪ sans état ≫

Le protocole HTTP est un protocole non connecté ou protocole ≪ sans états ≫ (stateless protocol), cela signifie que chaque requête est traitée indépendamment des autres et qu'aucun historique des différentes requêtes n'est conservé.

Cependant, on peut

- passer des paramètres d'une page à une autre ;
- on peut sauvegarder des informations durant toute la durée de la session.

Passage de paramètre via la méthode doPost()

Une requête POST n'est utilisable qu'avec un formulaire HTML.

```
<FORM ACTION="Resultat"
METHOD="POST">
<INPUT TYPE="TEXT" NAME="NOM">
<INPUT TYPE="TEXT" NAME="PRENOM">
<Input TYPE="SUBMIT" VALUE="Envoyer">
</FORM>
```

La méthode doPost() doit généralement recueillir les paramètres pour les traiter et générer la réponse.

Pour obtenir la valeur associée à chaque paramètre il faut utiliser la méthode getParameter() de l'objet HttpServletRequest. Cette méthode attend en paramètre le **nom du paramètre** dont on veut la valeur. Ce paramètre est sensible à la casse.

Récupération des paramètres dans Resultat.java

```
import javax.servlet.http.*;
import java.io.*;

public class Resultat extends HttpServlet {
      public void doPost(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
                  // pour l'affichage en HTML
                  PrintWriter out = res.getWriter();

                  String nom = req.getParameter("NOM");
                  String prenom = req.getParameter("PRENOM");

                  out.println("Nom : " + nom + ", prénom : " + prenom);

      }
}
```


HttpSession

Un objet de la classe HttpSession permet d'identifier un utilisateur et de mémoriser des données relatives à cet utilisateur.

Une servlet HTTP utilise un objet de la classe HttpSession pour créer une session entre un client HTTP et un serveur HTTP.

La session persiste pendant un temps limité (elle expire), elle permet à la servlet de voir et manipuler l'information concernant la session (identifiant, date de création, date du dernier accès, ...).

HttpSession : utilisation (1/2)

Un objet de type HttpSession s'obtient grâce à la méthode getSession(boolean) de la classe HttpServletRequest.

La méthode getSession(boolean) de la classe HttpServletRequest, avec l'argument true, permet de créer une session relative à l'utilisateur.

La méthode getSession(true) doit être appelée avant tout envoi de données au navigateur.

```
HttpSession session = req.getSession(true);
```


HttpSession : utilisation (1/2)

La méthode setAttribute(String cle,Object valeur) de la classe HttpSession permet de stocker des informations dans la session.

La méthode getAttribute(String cle) de la classe HttpSession permet de récupérer une valeur (type Object) qui a été précédemment stockée dans un objet de type HttpSession.

Il faut donc effectuer un **surtypage** pour obtenir un type élémentaire de données (par exemple un entier sera renvoyé sous forme d'objet Integer qu'il faudra convertir en int).

HttpSession : exemple (Index.java)

```
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class Index extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res)
           throws ServletException, IOException {
                  PrintWriter out = res.getWriter();

                  out.println("<FORM METHOD=\"POST\" ACTION=\"IndexAction\">");
                  out.println("Login:");
                  out.println("<INPUT TYPE=\"text\" NAME=\"login\">");
                  out.println("Password:");
                  out.println("<INPUT TYPE=\"password\" NAME=\"pwd\">");
                  out.println("<INPUT TYPE=\"submit\" VALUE=\"Connection\">");
                  out.println("<input type=\"reset\" value=\"Reset\">");
                  out.println("</FORM>");
    }
}
```


HttpSession : exemple (IndexAction.java)

```
public class IndexAction extends HttpServlet {

    String pwd,login;

    public void doPost(HttpServletRequest req, HttpServletResponse res)
                throws ServletException, IOException {
            login=req.getParameter("login");
            pwd=req.getParameter("pwd");
            HttpSession session=req.getSession(true);
            session.setAttribute("login",login);
            session.setAttribute("pwd",pwd);

            /* suite du code */
    }
}
```


HttpSession : exemple (Scholl.java)

```
public class Scholl extends HttpServlet {
    public void doPost(HttpServletRequest req, HttpServletResponse res)
                throws ServletException, IOException {
     Connection conn;
     String url = "jdbc:oracle:thin:@im2ag-oracle.e.ujf-grenoble.fr:1521:tptomcat";
     HttpSession session=req.getSession(true);
     String login=(String) session.getAttribute("login");
     String pwd=(String) session.getAttribute("pwd");
     String Requete = "select nocage from LesGardiens where nome = 'Scholl'";

     // pour l'affichage en HTML
     PrintWriter out = res.getWriter();
     conn = DriverManager.getConnection(url,login,pwd);
     Statement st = conn.createStatement();
     ResultSet rs = st.executeQuery(Requete);

     // l'affichage est simplifié : on peut y inclure du HTML
     out.println("Cages gardées par Scholl :");
     while(rs.next())
          out.println(rs.getString(1));

     rs.close();
     st.close();
     conn.close();
    }
}
```