Java - MySQL

Code: java-mysql

Originaux

url: http://tecfa.unige.ch/guides/tie/html/java-mysql/java-mysql.html

url: http://tecfa.unige.ch/guides/tie/pdf/files/java-mysql.pdf

Auteurs et version

- Daniel K. Schneider- Vivian Synteta

- Version: 1.0 (modifié le 25/1/01)

Prérequis

Module technique précédent: mysql-intro

Module technique précédent: java-intro

Module technique précédent: java-jhtml ou java-jsp (pour une partie)

Module technique précédent: java-swing (pour une partie)

Modules

Module technique suivant: ....

Objectifs

- Java - MySQL basics :)

1. Introduction au JDBC

Principe de base:

- Interface Java - bases de données SQL:
  JDBC = Java Data Base Connection
- Le langage Java définit une interface (commune)
- Chaque vendeur doit fournir une implémentation
  - certaines sont fournis avec le Java core
  - certaines (comme MM pour MySQL doivent être importées)

Quatre types d'interfaces

1. JDBC-ODBC bridge plus ODBC driver
   - nécessite une installation spéciale (driver du côté client par exemple)
2. Native-API partly-Java driver
   - se connecte dans une API client
3. JDBC-Net pure Java driver
   - utilise un middleware (serveur WWW - DB)
4. Native-protocol pure Java driver (on va utiliser ce type)
   - connexion directe Java-DB via le réseau, driver MySQL par exemple

Classes Java et outils utilisés dans ce module

Driver MySQL

url: mm.mysql Driver: http://www.worldserver.com/mm.mysql/

tecfa2: /local/java/classes/mm-jdbc/mysql.jar

Classes Swing (Swing ou le Netscape/Java plugin:

url: http://java.sun.com/products/jfc/index.html

Classes Servlet:

url: servlet SDK: http://java.sun.com/products/servlet/index.html

tecfa2: /local/java/JSDK2.0/lib/jsdk.jar

Copies dans http://tecfa.unige.ch/guides/java/classes/

- on y met une copie de la plupart des classes utilisées

1.1 Les interfaces API de base

Il existe une implémentation pour la plupart des bases de données

- on les appelle des "driver".
- Souvent, il faut trouver driver qq part et mettre son *.jar dans le classpath

JDBC API documentation:

http://tecfa2.unige.ch/guides/java/jdk1.1/docs/api/Package-java.sql.html

MM MySQL Driver documentation:

http://tecfa2.unige.ch/guides/mysql/local/mm-jdbc/doc/apidoc/

A. java.sql.DriverManager

- Gère les drivers JDBC
- on peut indiquer un Driver, qui existe:
  - soit avec la system property jdbc.drivers (Oracle par exemple)
  - soit avec des classes importées (comme dans notre cas)
- Bout de code:

```
try {
    Class.forName("org.gjt.mm.mysql.Driver");
} catch(Exception ex) { ... return; }
```

B. java.sql.Connection

- établir une connexion avec la base de données
- permet d'obtenir de la méta-information sur les tables (et autres)
- Bout de code:

```
String url = "jdbc:mysql://tecfa2.unige.ch:3306/COFFEEBREAK";
String user = "nobody";
String password = null;
Connection con = DriverManager.getConnection(url, user, password);
```

- Attention: (étudiants STAF) url = "jdbc:mysql:/ /localhost:3306/deiaco";

C. java. sql.Statement

- permet d'envoyer un statement SQL vers la base de données
- le résultat de la requête va se retrouver dans un ResultSet
- Bout de code:

```
String queryString = "select COF_NAME, PRICE from COFFEES";
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(queryString);
```

D. java.sql.Resultset

- Le Resultset (résultat d'une requête) est un objet spécial
- On y accède ligne par ligne (méthode *next*) ou avec une positionnement absolue ou relative par rapport à la ligne courante
- De chaque ligne, on extrait les "colonnes"avec des méthodes *getXXX*
  - Voir: http://tecfa.unige.ch/guides/java/tutorial/jdbc/basics/retrieving.html
  - Note: la méthode *getString* marche à peu près avec tout

Bout de code:

```
while (rs.next()) {
  String s = rs.getString("COF_NAME");
  float n = rs.getFloat("PRICE");
  System.out.println(s + "   " + n);
}
```

☞ Plus de détails: 2.3 "Anatomie d'un programme JDBC" [10]

Voir le tutoriel de SUN pour plus des exemples et infos:

url: http://developer.java.sun.com/developer/Books/JDBCTutorial/

2. Une introduction avec les exemples Coffee-break

- Voir le JDBC Database Access du "Java Tutoriel" de SUN
  - http://tecfa.unige.ch/guides/java/tutorial/jdbc/index.html

2.1 La table MySQL

- Il s'agit d'une simple table sur le server MySQL de tecfa2.unige.ch qu'on définit:

```
# Host: tecfa2  Database: COFFEEBREAK
>CREATE TABLE COFFEES (
  COF_NAME varchar(32),
  SUP_ID int(11),
  PRICE float(10,2),
  SALES int(11),
  TOTAL int(11)
);
# Insertion de 2 lignes ....
INSERT INTO COFFEES VALUES ('Colombian',101,7.99,0,0);
INSERT INTO COFFEES VALUES ('Espresso',150,9.99,0,0);
```

Et qui donne le resultat suivant (avec 'mysqlshow'):

```
Database: COFFEEBREAK  Table: COFFEES  Rows: 12
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| COF_NAME | varchar(32) |      | PRI |         |       |
| SUP_ID   | int(11)     | YES  |     |         |       |
| PRICE    | float(10,2) | YES  |     |         |       |
| SALES    | int(11)     | YES  |     |         |       |
| TOTAL    | int(11)     | YES  |     |         |       |
+----------+-------------+------+-----+---------+-------+
```

2.2 Les exemples JDBC/Coffeebreak

Exemple 2-1: Coffeebreak/JDBC examples

- Voir http://tecfa.unige.ch/guides/java/staf2x/ex/jdbc/coffee-break/README.html
- Ces exemples montrent un certain nombre de techniques JDBC avec des applications, servlets et applets

Exemple 2-2: Point de départ: les simples applications JDBC

- Voir http://tecfa.unige.ch/guides/java/staf2x/ex/jdbc/coffee-break/simple/
- Lisez d'abord la section 2.3 "Anatomie d'un programme JDBC" [10]
- Ensuite étudiez dans l'ordre:
  - ConnectCoffeesMM.java
  - CreateCoffeesMM.java (n'essayez pas ce programme sans l'adapter SVP!)
  - InsertCoffees.java
  - QueryCoffees.java

2.3 Anatomie d'un programme JDBC

- Voir aussi section 1.1 "Les interfaces API de base" [5]

A. Définition du driver

- Normalement au début d'un programme, dans une méthode init() ou start()

```
try {
    Class.forName("org.gjt.mm.mysql.Driver");
} catch(Exception ex) {
    ....
    return;}
```

- Attention:
  - N'oubliez pas de mettre le Driver dans le classpath !
  - N'importez pas les packages dans le programme

B. Connexion

- Egalement au début (si vous pensez faire plusieurs requêtes)
- vous avez intérêt à réfléchir comment définir url, user et password:
  - soit au début du fichier, soit dans des paramètres de l'applet, soit en demandant à l'utilisateur
  - (ne mettez pas de mot de passe en clair si possible)

```
String url = "jdbc:mysql://tecfa2.unige.ch:3306/COFFEEBREAK";
String user = "nobody";
String password = null;
try {
    con = DriverManager.getConnection(url, user, password);
}
catch(SQLException ex) { ... }
```

- une connexion se ferme

```
con.close();
```

  - soit à la fin de l'application
  - soit dans une méthode destroy(), close(), etc. pour les applets et servlets

C. La requête SQL

1. Il faut d'abord créer un objet "Statement" qui enverra des requêtes SQL à la base de données

```
Statement stmt = con.createStatement();
```

   - un objet statement peut être utilisé plusieurs fois

2. La classe statement possède deux méthodes pour envoyer du SQL à la DB:
   - executeQuery() pour des requêtes (SELECT)

```
String queryString = "SELECT COF_NAME, PRICE FROM COFFEES";
... stmt.executeQuery(queryString);
```

   - executeUpdate() pour des updates (INSERT, UPDATE etc.)

```
String queryString = "INSERT INTO COFFEES values (NULL, 'Vivian coffee',
'100')";
... stmt.executeUpdate(queryString);
```

3. Récupération du résultat des requêtes(SELECT)

```
String queryString = "SELECT COF_NAME, PRICE FROM COFFEES";
ResultSet rs = stmt.executeQuery(queryString);
```

   - la requête retourne un objet (étrange) de type ResultSet
   - on peut y accéder ligne par ligne ou autrement (charactéristique très recent) voir D. "Traitement d'un ResultSet dont on connaît la structure" [14]
   - ne PAS confondre les colonnes dans la db avec celles du ResultSet !

4. Récupération du résultat des requêtes (INSERT, UPDATE, etc.)

```
String updateString = "UPDATE COFFEES SET SALES = 75 WHERE COF_NAME LIKE
'Colombian'";
int updateRowCount = stmt.executeUpdate(updateString);
```

- retourne le nombre de colonnes mises à jour / insérées
- utile pour donner un feedback
  (0 indique un update sans effet, par exemple le résultat d'une syntaxe juste mais mauvaises valeurs).
- Note: Un faux statement SQL produit une erreur

5. Statements "préparés" (Prepared statements)

```
PreparedStatement prep = con.prepareStatement(
                    "INSERT into Data values (?, ?)");
```

- *Connection.prepareStatement*, à la place de: *Connection.createStatement*
- quand on veut répéter le même "statement" plusieurs fois
- les "?" remplacent les différentes données qu'on va envoyer à la bd chaque fois avec les méthodes *setXXX*, par ex:

```
prep.setString(1, "Jim");
prep.setInt(2, 70);
```

- NOTE: Il faut faire attention aux types des données!

Par ex: *setInt*, *setFloat*, *setDouble* pour Integer, Float et Double, *setString* pour char et varchar, et *setDate* pour dates.

- *** On exécute avec *executeUpdate()* ***

D. Traitement d'un ResultSet dont on connaît la structure

- En règle général vous avez intérêt à séparer requête et affichage:
  - copier les données dans une table que vous passez ensuite à une méthode qui fait l'affichage (enfin faut savoir utiliser les Vectors...)
  - ou encore passer des objets à une fonction qui fait l'affichage
  - on accède la première ligne des resultats avec *result.next();*
- On extrait les colonnes que l'on veut avec des méthodes *getXXX*,

  voir: http://tecfa.unige.ch/guides/java/tutorial/jdbc/basics/retrieving.html
  - Note: la méthode *getString* marche à peu près avec tout, mais on a intéret à récuperer un nombre comme nombre si on désire faire des calculs par exemple

    getInt, *getFloat*, *getDouble* pour Integer, Float et Double,

    getString pour char et varchar,

    et *getDate* pour dates.

  - Récuperation de 2 colonnes:

```
while (rs.next()) {
  String s = rs.getString("COF_NAME");
  float n = rs.getFloat("PRICE");
  System.out.println(s + "   " + n);
  }
```

E. Les meta data

- Le ResultSet contient aussi des meta data (données sur les colonnes, le nombre de colonnes, etc. (PAS le nombre de lignes!)

```
ResultSetMetaData rsMeta = (ResultSetMetaData) rs.getMetaData();
// Get the N of Cols in the ResultSet
int noCols = rsMeta.getColumnCount();
// initalize the Vector that holds labels
String [] resColNames = new String [noCols];
for (int c=1; c<=noCols; c++) {
     String el = rsMeta.getColumnLabel(c);
     resColNames[c-1]= el;  }
```

F. Traitement d'un ResultSet dont on ne connaît pas la structure

- Récupération de toutes les colonnes et stockage dans une table (sans savoir ce qui a dans le ResultSet: (Ex. avec une matrice (à ne pas copier, faudrait utiliser Vector() à la place .....)

```
int noCols = rsMeta.getColumnCount();
int line=0;
String [][] resultData = new String [100][noCols];
while (rs.next()) {
    for (int c=1; c<=noCols; c++) {
        String el = rs.getString(c);
        resultData[line][c-1] = el;
    }
    line++;      }
```

G. Gestion des erreurs SQL (SQLExceptions, Warnings, etc)

- Il existe 3 types d'exceptions et 3 classes alors: SQLException, SQLWarning et DataTruncation et vous avez intérêt à faire ça sérieusement, par exemple:

```
catch(SQLException ex) {
    System.err.println("==> SQLException: ");
    while (ex != null) {
        System.out.println("Message:   " + ex.getMessage ());
        System.out.println("SQLState:  " + ex.getSQLState ());
        System.out.println("ErrorCode: " + ex.getErrorCode ());
        ex = ex.getNextException();
        System.out.println("");
    }
}
```

Exercice 1: Les premiers pas avec le JDBC

1. Créez une table dans une base de données (ou travaillez avec une que vous avez déjà) ... demandez des permissions pour "nobody" (si vous n'avez pas)
2. Adaptez les exemples simples ci-dessus

- Classpath pour les exemples simples:
  - doit contenir le driver mm.mysql (mysql.jar)
  - Sous Windows, tout dépend de votre installation ..... :)
  - Sous Unix/Tecfa, on conseille de taper:

```
'source /local/env/java-sql-xml-ser-swing.csh'
```

2.4 Programme Java complet: simple query

```
import java.sql.*;

public class QueryCoffees {
    public static void main(String args[]) {
     // ---- configure this for your site
     String username = "nobody";
     String password = null;
     // The URL that will connect to TECFA's MySQL server
     // Syntax: jdbc:TYPE:machine:port/DB_NAME
     String url = "jdbc:mysql://tecfa2.unige.ch:3306/COFFEEBREAK";
     // A canned query string
     String queryString = "SELECT COF_NAME, PRICE FROM COFFEES";
     // ---- configure END
     // INSTALL/load the Driver (Vendor specific Code)
     try {
         Class.forName("org.gjt.mm.mysql.Driver");
     } catch (java.lang.ClassNotFoundException e) {
         System.err.print("ClassNotFoundException: ");
         System.err.println(e.getMessage());
     }
     try {
         Connection con;
         Statement stmt;
         // Connection to the database at URL with usename and password
         con = DriverManager.getConnection(url,username,password);
         System.out.println ("Ok, connection to the DB worked.");
```

```
            System.out.println ("Let's see can retrieve something with: " +
    queryString);
            // Create a Statement Object
            stmt = con.createStatement();
            // Send the query and bind to the result set
            ResultSet rs = (ResultSet) stmt.executeQuery(queryString);
            while (rs.next()) {
                String s = rs.getString("COF_NAME");
                float n = rs.getFloat("PRICE");
                System.out.println(s + "   " + n);
            }
            // Close resources
            stmt.close();
            con.close();
        }
        // print out decent error messages
        catch(SQLException ex) {
            System.err.println("==> SQLException: ");
            while (ex != null) {
            System.out.println("Message:   " + ex.getMessage ());
            System.out.println("SQLState:  " + ex.getSQLState ());
            System.out.println("ErrorCode: " + ex.getErrorCode ());
            ex = ex.getNextException();
            System.out.println("");
        }
     }
    }
```

3. Servlets ou pages JSP

- L'archive mysql.jar est installé sur nos serveurs Java
- Faites un répertoire "WEB-INF" dans votre espace travaux et là dedans un sous-répertoire "classes" où vous allez mettre les classes
- URL pour executer: http:/ /tecfa.unige.ch/staf/staf-f/deiaco/servlet/

3.1 Une simple page JSP

Exemple 3-1: Une simple page JSP avec fenêtre "Query"

url: http://tecfa2.unige.ch/guides/jsp/ex/jdbc/coffee-break/CoffeeBreakLine.jsp

url: http://tecfa2.unige.ch/guides/jsp/ex/jdbc/coffee-break/CoffeeBreakLine.jsp.text

```
    <h1>Coffee Break JSP Example</h1>
<%@ page errorPage="error.jsp" import="java.sql.*" %>
<%! // --------------- inits for the servlet -------------
// The database connection
Connection con;
// The statement
Statement stmt;
// The queryString
String queryString = null;
// ---- configure this for your site
String username = "nobody";
String password = null;
```

```
// The URL that will connect to TECFA's MySQL server
// Syntax: jdbc:TYPE:machine:port/DB_NAME
// String url = "jdbc:mysql://localhost:3306/COFFEEBREAK";
String url = "jdbc:mysql://tecfa2.unige.ch:3306/COFFEEBREAK";
%>
<%  // --------------- code for the service method  --------------
    // Let's see if we got a request
queryString = request.getParameter ("QUERYSTRING");
if ((queryString != "") && (queryString != null)) {

     try {
         Class.forName("org.gjt.mm.mysql.Driver");
         // Establish Connection to the database at URL with usename and
password
         con = DriverManager.getConnection(url, username, password);
         out.println ("Ok, connection to the DB is working.");
     } catch (Exception e) // (ClassNotFoundException and SQLException)
     {
         throw(new UnavailableException(this, "Sorry! The Database didn't
load!"));
     }
     try {
         out.println ("<h2>You asked: </h2>");
         out.println ( "Query:  " + queryString + "<BR>" );
         out.println("<h3>Query Result</h3>");
         out.println("<table border>");
         stmt = con.createStatement();
         ResultSet rs = stmt.executeQuery(queryString);
```

```
        ResultSetMetaData rsMeta = rs.getMetaData();
        // Get the N of Cols in the ResultSet
        int noCols = rsMeta.getColumnCount();
        out.println("<tr>");
        for (int c=1; c<=noCols; c++) {
            String el = rsMeta.getColumnLabel(c);
            out.println("<th> " + el + " </th>");
        }
        out.println("</tr>");
        while (rs.next()) {
            out.println("<tr>");
            for (int c=1; c<=noCols; c++) {
                String el = rs.getString(c);
                out.println("<td> " + el + " </td>");
            }
            out.println("</tr>");
        }
        out.println("</table>");
    } catch (SQLException ex ) {
        out.println ( "<P><PRE>" );
        while (ex != null) {
            out.println("Message:   " + ex.getMessage ());
            out.println("SQLState:  " + ex.getSQLState ());
            out.println("ErrorCode: " + ex.getErrorCode ());
            ex = ex.getNextException();
            out.println("");
        }
        out.println ( "</PRE><P>" );
```

```
      }
}
%>

<hr>You can now try to retrieve something.

<FORM METHOD=POST ACTION="CoffeeBreakLine.jsp">
Query: <INPUT TYPE=TEXT SIZE=50 NAME="QUERYSTRING">
<INPUT TYPE=SUBMIT VALUE="GO!">
</FORM>
<hr><pre>e.g.:
SELECT * FROM COFFEES
SELECT * FROM COFFEES WHERE PRICE > 9
SELECT PRICE, COF_NAME FROM COFFEES
<pre>

<hr><a href="CoffeeBreakLine.jsp">Encore une fois ?</a> |
Source: <A HREF="CoffeeBreakLine.jsp.text">CoffeeBreakLine.jsp.text</A>
```

Exemple 3-2: Une simple page JSP avec fenêtres Query et UpDate

- presque pareil que l'exemple précédent, mais rajoute une fenêtre pour faire des updates

  url: http://tecfa2.unige.ch/guides/jsp/ex/jdbc/coffee-break/CoffeeBreakLine2.jsp

3.2 Simple Servlet

Exemple 3-3: Simple Servlet

url: http://tecfa2.unige.ch:8080/servlet/CoffeeBreakServlet (Sun/Jws)
url: http://tecfa2.unige.ch/servlets/CoffeeBreakServlet (Apache/JServ)
url: http://tecfa2.unige.ch/guides/java/staf2x/ex/jdbc/coffee-break/servlet/

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

  [ ........]

public class CoffeeBreakServlet extends HttpServlet {
    // The database connection
    Connection con;
    // The statement
    Statement stmt;
    // The queryString
    String queryString = null;

    public void init(ServletConfig conf) throws ServletException {
         super.init(conf);
         // ---- configure this for your site
         String username = "nobody";
         String password = null;
```

```
          // The URL that will connect to TECFA's MySQL server
          String url = "jdbc:mysql://tecfa2.unige.ch:3306/COFFEEBREAK";
          // ---- configure END

          try {
              Class.forName("org.gjt.mm.mysql.Driver");
              // Connection to the database at URL with username and password
              con = DriverManager.getConnection(url, username,password);
              System.out.println ("Ok, connection to the DB is working.");
          } catch (Exception e) // (ClassNotFoundException and SQLException)
          {
              throw(new UnavailableException(this, "Sorry! The Database didn't
load!"));
          }
     }
     public void service (HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException {

          res.setContentType ( "text/html" );
          PrintWriter  out = res.getWriter ( );

          try {
              String title = "Coffee Break JDBC Demo Java Servlet";
              out.println ( "<html><head><title>" + title
        + "</title></head>" );
              out.println ( "<body><H1>" + title + "</H1>" );
              String  queryString = req.getParameter ("QUERYSTRING");
```

```
        if ((queryString != "") && (queryString != null)) {
            out.println ("<h2>You asked: </h2>");
            out.println ( "Query:  " + queryString + "<BR>" );
            out.println("<h3>Query Result</h3>");
            out.println("<table border>");
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery(queryString);
            ResultSetMetaData rsMeta = rs.getMetaData();
            // Get the N of Cols in the ResultSet
            int noCols = rsMeta.getColumnCount();
            out.println("<tr>");
            for (int c=1; c<=noCols; c++) {
                String el = rsMeta.getColumnLabel(c);
                out.println("<th> " + el + " </th>");
            }
            out.println("</tr>");
            while (rs.next()) {
                out.println("<tr>");
                for (int c=1; c<=noCols; c++) {
                    String el = rs.getString(c);
                    out.println("<td> " + el + " </td>");
                }
                out.println("</tr>");
            }
            out.println("</table>");
        }
    } catch { [ .. gérer les erreurs de la requête......] }
```

```
      out.println ("<hr>You can now try to retrieve something.");
      out.println("<FORM METHOD=POST ACTION='/servlet/CoffeeBreakServlet'>");
      out.println("Query: <INPUT TYPE=TEXT SIZE=50 NAME='QUERYSTRING'> ");
      out.println("<INPUT TYPE=SUBMIT VALUE='GO!'>");
      out.println("</FORM>");
      out.println("<hr><pre>e.g.:");
      out.println("SELECT * FROM COFFEES");
      out.println("SELECT * FROM COFFEES WHERE PRICE > 9");
      out.println("SELECT PRICE, COF_NAME FROM COFFEES");
      out.println("<pre>");

      out.println ("<hr><a href='/servlet/CoffeeBreakServlet'>Enocre une fois
?</a> | Source: <A HREF='/develop/servlets-ex/coffee-break/
CoffeeBreakServlet.java'>CoffeeBreakServlet.java</A>");
      out.println ( "</body></html>" );
      return ;
   }
}
```

4. Applications et applets (à REFAIRE et compléter)

A. Applications

- il faut les classes nécessaires dans le Classpath !

B. Applets

- L'archive mysql.jar doit être accessible par l'applet, exemple:

```
<APPLET
  CODE="MySqlQueryLineApplet.class" WIDTH=500 HEIGHT=300
  ARCHIVE="/guides/java/classes/mysql.jar"
>
</APPLET>
```

- Pour utiliser Swing il faut l'installer côté client (ou utiliser Java 1.2)
  - Mettre swing.jar dans Netscape\Communicator\Program\Java\Classes\
- Pour Java 1.2 / Netscape il faut utiliser le <embed> tag
  - (pour IE <object>, voir la doc !)

```
 <EMBED type="application/x-java-applet;version=1.2" width="200"
    height="200" align="baseline" code="XYZApp.class"
    codebase="html/" model="models/HyaluronicAcid.xyz"
    pluginspage="http://java.sun.com/products/plugin/1.2/plugin-install.html">
 <NOEMBED>
    No JDK 1.2 support for APPLET!!
 </NOEMBED>
 </EMBED>
```

  - voir: http://204.160.241.19/products/plugin/1.2/docs/tags.html

4.1 Applets

Exemple 4-1: Very simple Query (My)SQL Applet

- http://tecfa2.unige.ch/guides/java/staf2x/ex/jdbc/coffee-break/simple-query-applet/MySqlQueryLineApplet.html
- Montre comment récuper un query SQL dans un textField et l'envoyer à la DB
- Fonctionalité limitée !

Exemple 4-2: Update / Query SQL Applet for COFFEES

- http://tecfa2.unige.ch/guides/java/staf2x/ex/jdbc/coffee-break/simple-query-applet/MySqlUpdateLineApplet.html
- Montre comment récuperer des queries ou updates dans deux simples textFields et comment afficher (simplement) le résultat

Exemple 4-3: Update / Query MySQL Swing Applet for COFFEES

- http://tecfa2.unige.ch/guides/java/staf2x/ex/jdbc/coffee-break/simple-query-applet/MySqlUpdateSwingApplet.html
- Même chose que ci-dessus, mais avec un affichage (plus ou moins) correct
- N'oubliez pas de mettre swing.jar dans vos java/classes de Netscape !

A. Pages HTML

- Voici la section qui inclut l'applet, notez:
  - l'utilisation de paramètres (cela rend l'applet plus portable)
  - l'archive mysql.jar

```
<APPLET
  CODE="MySqlUpdateLineApplet.class" WIDTH=600 HEIGHT=400
  ARCHIVE="/guides/java/classes/mysql.jar"
>
  <param name=URL value="jdbc:mysql://tecfa2.unige.ch:3306/COFFEEBREAK">
  <param name=USER value="nobody">
  <param name=PASSWORD value="">

</APPLET>
```

- Voici le code Java qui récupère les paramètres

```
String url = null;
String user = null;
String password = null;
.....
 url = getParameter("URL");
 user = getParameter("USER");
 // password = getParameter("PASSWORD");
 password = null;
```