CmpE 473
Internet Programming

Pınar Yolum
pinar.yolum@boun.edu.tr

Department of
Computer Engineering
Boğaziçi University

JDBC

Examples from java.sun.com

JDBC (1)

- Java DataBase Connectivity
  - Access to data in relational databases or spreadsheet
- Packages
  - java.sql: Access to DB
  - javax.sql: Also, server side capabilities
- Need a driver between JDBC and the DB
  - Existing JDBC-ODBC bridge
- Open source DBMS: MySql, PostgreSQL

JDBC (2)

- Establish a connection with the DBMS
  - Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  - Loads the jdbc-odbc bridge driver
- Connect with the driver

  String url = “jdbc:odbc:dbname”

  Connection con = DriverManager.getConnection(url, “pyolum", “pass");
- URL always starts with jdbc:
- Followed by the subprotocol name:
- The rest depends on where the database is
  - Ex: jdbc:z1MySQL://luna.oit.unc.edu/CES
- con: open connection to make SQL statements

JDBC (3)

- General rule for executing SQL statement
  - Write the SQL query
  - Convert it into a string
  - Create a statement object
  - Call Statement.executeUpdate with the string for UPDATE and CREATE queries OR
  - Call Statement.executeQuery with the string for SELECT queries
- Use java.sql.Types for datatypes in SQL queries

Example (1)

- CREATE TABLE COFFEES
  (COF_NAME VARCHAR(32),
  SUP_ID INTEGER,
  PRICE FLOAT,
  SALES INTEGER,
  TOTAL INTEGER)
- String createTableCoffees =
  "CREATE TABLE COFFEES " +
  "(COF_NAME VARCHAR(32), SUP_ID INTEGER, PRICE FLOAT, " +
  "SALES INTEGER, TOTAL INTEGER)";
- Statement stmt = con.createStatement();
- stmt.executeUpdate(createTableCoffees);

Example (2)

- ResultSet object keeps a set of rows of data
- Use .next to move through a ResultSet

  ResultSet rs = stmt.executeQuery("SELECT COF_NAME, PRICE FROM COFFEES");
- To retrieve data, use a get*DataType* method that matches the entry

```
String query = "SELECT COF_NAME, PRICE FROM COFFEES";
ResultSet rs = stmt.executeQuery(query);
while (rs.next()) {
    String s = rs.getString("COF_NAME"); //Note conversion to string
    float n = rs.getFloat("PRICE"); //Note conversion to Java float
    System.out.println(s + " " + n);
}
```

- Watch out for type conversions.
  - Ex: You can use getByte to retrieve a float!

Example (3)

- Alternative SELECT (with column numbers)

```
ResultSet rs = stmt.executeQuery(query);
while (rs.next()) {
    String s = rs.getString(1);
    int n = rs.getInt(2);
    System.out.println(n + " pounds of " + s + " sold to date.");
}
```

- Update table

```
String updateString = "UPDATE COFFEES " +
    "SET SALES = 75 " +
    "WHERE COF_NAME LIKE 'Colombian'";
```

- stmt.executeUpdate(updateString);

Prepared Statement

- Prepared Statement
  - Fix the SQL query when creating the statement
  - SQL will be precompiled by the DBMS
  - Speed things up
- PreparedStatement updateSales = con.prepareStatement("UPDATE COFFEES SET SALES = ? WHERE COF_NAME LIKE ?");
- updateSales.setInt(1, 75)
- updateSales.setString(2, “Colombian”)
- updateSales.executeUpdate():

Join

- Consult two tables to get the results

```
String query = " SELECT COFFEES.COF_NAME " +
    "FROM COFFEES, SUPPLIERS " +
    "WHERE SUPPLIERS.SUP_NAME LIKE 'Acme, Inc.' " +
    "and SUPPLIERS.SUP_ID = COFFEES.SUP_ID";

    ResultSet rs = stmt.executeQuery(query);
    System.out.println("Coffees bought from Acme, Inc.: ");

    while (rs.next()) {
            String coffeeName = rs.getString("COF_NAME");
    System.out.println(" " + coffeeName);

    }
```

Transactions

- Group statements to be committed at once

```
con.setAutoCommit(false);
PreparedStatement updateSales = con.prepareStatement( "UPDATE COFFEES
    SET SALES = ? WHERE COF_NAME LIKE ?");
updateSales.setInt(1, 50);
updateSales.setString(2, "Colombian");
updateSales.executeUpdate();

PreparedStatement updateTotal = con.prepareStatement( "UPDATE COFFEES
    SET TOTAL = TOTAL + ? WHERE COF_NAME LIKE ?");
updateTotal.setInt(1, 50);
updateTotal.setString(2, "Colombian");
updateTotal.executeUpdate();
con.commit();
con.setAutoCommit(true);
```