Aplicatii JAVA

10 JAVA
Lucrul cu baze de date în Java

Adrian Runceanu
www.runceanu.ro/adrian

Curs 10

Lucrul cu baze de date în Java

Lucrul cu baze de date în Java

1. Generalitati despre baze de date
2. Ce este JDBC?
3. Conectarea la o baza de date
4. Efectuarea de secvente SQL
5. Obtinerea si prelucrarea rezultatelor
6. Exemplu

1. Generalitati despre baze de date

Definitie

O baza de date reprezinta o modalitate de stocare a unor informatii (date) pe un suport extern, cu posibilitatea regasirii acestora.

- Uzual, o baza de date este memorata într-unul sau mai multe fisiere.
- Modelul clasic de baza de date este cel relational, în care datele sunt memorate în tabele.
- Pe lânga tabele, o baza de date mai poate contine: proceduri si functii, utilizatori si grupuri de utilizatori, tipuri de date, obiecte, etc.
- Dintre producatorii cei mai importanti de baze de date amintim Oracle, Sybase, IBM, Informix, Microsoft, etc.

1. Generalitati despre baze de date

Crearea unei baze de date

- Se face cu aplicatii specializate oferite de producatorul tipului respectiv de baza de date.

Accesul la o baza de date

- Se face prin intermediul unui driver specific tipului respectiv de baza de date.
- Acesta este responsabil cu accesul efectiv la datele stocate, fiind legatura între aplicatie si baza de date.

Lucrul cu baze de date în Java

1. Generalitati despre baze de date
2. Ce este JDBC?
3. Conectarea la o baza de date
4. Efectuarea de secvente SQL
5. Obtinerea si prelucrarea rezultatelor
6. Exemplu

2. Ce este JDBC?

Definitie

JDBC (Java Database Connectivity) este o interfata standard SQL de acces la baze de date.

- **JDBC** este constituita dintr-un set de clase si interfete scrise în Java, furnizând mecanisme standard pentru proiectantii aplicatiilor de baze de date.

- Pachetul care ofera suport pentru lucrul cu baze de date este **java.sql**.

2. Ce este JDBC?

- *Folosind JDBC este usor sa transmitem secvente SQL catre baze de date relationale.*
- Cu alte cuvinte, nu este necesar sa scriem un program pentru a accesa o baza de date Oracle, alt program pentru a accesa o baza de date Sybase si asa mai departe.
- Este de ajuns sa scriem un singur program folosind API-ul JDBC si acesta va fi capabil sa trimita secvente SQL bazei de date dorite.
- Bineînteles, scriind codul sursa în Java, ne este asigurata portabilitatea programului.
- Deci, iata doua motive puternice care fac combinatia Java - JDBC demna de luat în seama.

2. Ce este JDBC?

Fiind robust, sigur, usor de folosit, usor de înteles, Java este un excelent limbaj pentru a dezvolta aplicatii de baze de date.

In linii mari, **JDBC** face trei lucruri:

1. *stabileste o conexiune cu o baza de date*

2. *trimite secvente SQL*

3. *prelucreaza rezultatele*

Lucrul cu baze de date în Java

1. Generalitati despre baze de date
2. Ce este JDBC?
3. Conectarea la o baza de date
4. Efectuarea de secvente SQL
5. Obtinerea si prelucrarea rezultatelor
6. Exemplu

3. Conectarea la o baza de date

Procesul de conectare la o baza de date implica doua operatii:

1. încarcarea în memorie a unui driver corespunzator

2. realizarea unei conexiuni propriu-zise

3. Conectarea la o baza de date

Definitie

O conexiune (sesiune) la o baza de date reprezinta un context prin care sunt trimise secvente SQL si primite rezultate.

Intr-o aplicatie pot exista mai multe conexiuni simultan la baze de date diferite sau la aceeasi baza.

3. Conectarea la o baza de date

Clasele si interfetele responsabile cu realizarea unei conexiuni sunt:

- **clasa DriverManager**, ce se ocupa cu înregistrarea driverelor ce vor fi folosite în aplicatie

- **interfata Driver**, pe care trebuie sa o implementeze orice clasa ce descrie un driver

- **clasa DriverPropertyInfo**

- **interfata Connection**, descrie obiectele ce modeleaza o conexiune propriu-zisa cu baza de date

3. Conectarea la o baza de date

Incarcarea în memorie a unui driver

- Primul lucru pe care trebuie sa-l faca o aplicatie în procesul de conectare la o baza de date este sa încarce în memorie clasa ce implementeaza driver-ul necesar comunicarii cu respectiva baza de date.

3. Conectarea la o baza de date

Incarcarea în memorie a unui driver(continuare)

➢ Acest lucru poate fi realizat prin mai multe modalitati:

1. DriverManager.registerDriver(new sun.jdbc.odbc.JdbcOdbcDriver());

2. Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

3. System.setProperty("jdbc.drivers", "sun.jdbc.odbc.JdbcOdbcDriver");

4. java -Djdbc.drivers=sun.jdbc.odbc.JdbcOdbcDriver

3. Conectarea la o baza de date

- O data ce un driver **JDBC** a fost încarcat în memorie cu DriverManager, acesta poate fi folosit la stabilirea unei conexiuni cu o baza de date.

- Având în vedere faptul ca pot exista mai multe drivere înregistrate în memorie, trebuie sa avem posibilitatea de a specifica pe lânga identificatorul bazei de date si driverul ce trebuie folosit.

3. Conectarea la o baza de date

Aceasta se realizeaza prin intermediul unei adrese specifice, numita **JDBC URL**, ce are urmatorul format:

jdbc:sub-protocol:identificator_baza_de_date

- Câmpul *sub-protocol* denumeste tipul de driver ce trebuie folosit pentru realizarea conexiunii si poate fi *odbc, oracle, sybase, db2* si asa mai departe. *Identificatorul bazei de date* este un indicator specific fiecarui driver care specifica baza de date cu care aplicatia doreste sa interactioneze.

3. Conectarea la o baza de date

In functie de tipul driver-ului acest identificator poate include numele unei masini gazda, un numar de port, numele unui fisier sau al unui director, etc.

jdbc:odbc:testdb
jdbc:oracle:thin@persistentjava.com:1521:testdb
jdbc:sybase:testdb
jdbc:db2:testdb

3. Conectarea la o baza de date

- La primirea unui **JDBC URL**, DriverManager-ul va parcurge lista driver-elor înregistrate în memorie, pâna când unul dintre ele va recunoaste URL-ul respectiv.

- Daca nu exista nici unul potrivit, atunci va fi lansata o exceptie de tipul SQLException, cu mesajul *no suitable driver.*

3. Conectarea la o baza de date

Realizarea unei conexiuni

Metoda folosita pentru realizarea unei conexiuni este getConnection din **clasa DriverManager** si poate avea mai multe forme:

```
Connection c = DriverManager.getConnection(url);
Connection c = DriverManager.getConnection(url,
username, password);
Connection c = DriverManager.getConnection(url,
dbproperies);
```

3. Conectarea la o baza de date

O conexiune va fi folosita pentru:

- crearea de secvente **SQL** ce vor fi folosite pentru interogarea sau actualizarea bazei
- aflarea unor informatii legate de baza de date (meta-date)

Clasa Connection asigura suport pentru controlul tranzactiilor din memorie catre baza de date prin metodele commit, rollback, setAutoCommit.

Lucrul cu baze de date în Java

1. Generalitati despre baze de date
2. Ce este JDBC?
3. Conectarea la o baza de date
4. Efectuarea de secvente SQL
5. Obtinerea si prelucrarea rezultatelor
6. Exemplu

4. Efectuarea de secvente SQL

O data facuta conectarea cu DriverManager.getConection(), se poate folosi obiectul Connection rezultat pentru a se crea un obiect de tip Statements, cu ajutorul caruia putem trimite secvente **SQL** catre baza de date.

Cele mai uzuale comenzi **SQL** sunt cele folosite pentru:

1. interogarea bazei de date (SELECT)

2. actualizarea bazei de date (INSERT, UPDATE, DELETE)

4. Efectuarea de secvente SQL

```
Connection c = DriverManager.getConnection(url);
Statement s = c.createStatement();
ResultSet r = s.executeQuery("SELECT * FROM
un_tabel ORDER BY o_coloana");
s.executeUpdate("DELETE * FROM un_tabel");
```

Metoda executeQuery trimite interogari **SQL** catre baza de date si primeste raspuns într-un obiect de tip ResultSet.

Lucrul cu baze de date în Java

1. Generalitati despre baze de date
2. Ce este JDBC?
3. Conectarea la o baza de date
4. Efectuarea de secvente SQL
5. Obtinerea si prelucrarea rezultatelor
6. Exemplu

5. Obtinerea si prelucrarea rezultatelor

Interfata ResultSet

```
String query = "SELECT cod, nume FROM angajati
ORDER BY nume";
ResultSet r = s.executeQuery( query );
while (r.next()) {
        System.out.println (r.getString ("cod") + "," +
r.getString ("nume") );
}
```

5. Obtinerea si prelucrarea rezultatelor

Interfata ResultSetMetaData

```
ResultSet r = s.executeQuery("SELECT * FROM angajati" );
ResultSetMetaData rsmd = r.getMetaData();
System.out.println("Coloane: " +
rsmd.getColumnCount());
```

Lucrul cu baze de date în Java

1. Generalitati despre baze de date
2. Ce este JDBC?
3. Conectarea la o baza de date
4. Efectuarea de secvente SQL
5. Obtinerea si prelucrarea rezultatelor
6. Exemplu

Exemplu de conectare

```
import java.sql.*;
import java.io.*;
public class TestJDBC {
public static void main (String[] args) {
        String dbUrl = "jdbc:odbc:test";
        String user = "dba";
        String password = "sql";
```

Exemplu de conectare

```
try{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
}
catch(ClassNotFoundException e) {
    e.printStackTrace();
    System.out.println("Eroare incarcare driver!\n" +
e);
}
```

Exemplu de conectare

```
try{
        Connection c=DriverManager.getConnection(dbUrl, user,
password);
        Statement s= c.createStatement();
        ResultSet r = s.executeQuery(
        " SELECT cod, nume FROM localitati"+
        " ORDER BY nume");
        while (r.next()) {
                System.out.println (
                r.getString ("cod") + "," +
                r.getString ("nume") );
        }
        s.close();
}
catch(SQLException e) {
        e.printStackTrace();
        }
}
}
```

Exemplul 2 de conectare la o baza de date

http://www.tutorialspoint.com/jdbc/jdbc-sample-code.htm

```
// STEP 1. Import required packages
import java.sql.*;

public class FirstExample {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER =
"com.mysql.jdbc.Driver";
   static final String DB_URL =
"jdbc:mysql://localhost/EMP";
```

Exemplul 2 de conectare la o baza de date

```
// Database credentials

   static final String USER = "username";
   static final String PASS = "password";

   public static void main(String[] args) {
   Connection conn = null;
   Statement stmt = null;
```

Exemplul 2 de conectare la o baza de date

```
try{
    //STEP 2: Register JDBC driver
    Class.forName("com.mysql.jdbc.Driver");

    //STEP 3: Open a connection
    System.out.println("Connecting to database...");
    conn = DriverManager.getConnection(DB_URL,USER,PASS);

    //STEP 4: Execute a query
    System.out.println("Creating statement...");
    stmt = conn.createStatement();
    String sql;
    sql = "SELECT id, first, last, age FROM Employees";
    ResultSet rs = stmt.executeQuery(sql);
```

Exemplul 2 de conectare la o baza de date

```
//STEP 5: Extract data from result set
    while(rs.next()){
      //Retrieve by column name
      int id  = rs.getInt("id");
      int age = rs.getInt("age");
      String first = rs.getString("first");
      String last = rs.getString("last");

      //Display values
      System.out.print("ID: " + id);
      System.out.print(", Age: " + age);
      System.out.print(", First: " + first);
      System.out.println(", Last: " + last);
    }
    //STEP 6: Clean-up environment
    rs.close();
    stmt.close();
    conn.close();
  }
```

Exemplul 2 de conectare la o baza de date

```
catch(SQLException se){
    //Handle errors for JDBC
    se.printStackTrace();
 }
 catch(Exception e){
    //Handle errors for Class.forName
    e.printStackTrace();
 }
```

Exemplul 2 de conectare la o baza de date

```
finally{
    //finally block used to close resources
    try{
      if(stmt!=null)
         stmt.close();
    }
    catch(SQLException se2){
    }  // nothing we can do
    try{
      if(conn!=null)
         conn.close();
    }
    catch(SQLException se){
      se.printStackTrace();
    } //end finally try
  } //end try
  System.out.println("Goodbye!");
 } //end main
} //end FirstExample
```

Exemplul 2 de conectare la o baza de date

```
C:\>java FirstExample

Connecting to database...
Creating statement...
ID: 100, Age: 18, First: Zara, Last: Ali
ID: 101, Age: 25, First: Mahnaz, Last: Fatma
ID: 102, Age: 30, First: Zaid, Last: Khan
ID: 103, Age: 28, First: Sumit, Last: Mittal
```

Referinte

- Curs practic de Java, Cristian Frasinaru – capitolul Conexiunea cu bazele de date in Java

Întrebări?