JAVA DATABASE CONNECTIVITY

Ненко Табаков
Пламен Танов
Технологическо училище “Електронни системи”
Технически университет – София

4 февруари 2009

ЛИТЕРАТУРА НЕОБХОДИМИ ПРОГРАМИ

- Java API документация - http://java.sun.com/javase/6/docs/api/
- JDBC Tutorial - http://java.sun.com/docs/books/tutorial/jdbc/index.html
- Eclipse - www.eclipse.org

CC-BY-SA

ВЪВЕДЕНИЕ

- JDBC API е Java API, което може да бъде използвано за достъп до всякакъв вид данни, оформени като таблици
- JDBC улеснява работата при:
  - Свързване към източник на данни – бази данни
  - Изпращане на запитвания към база данни
  - Промяна на стойностите в база данни
  - Получаване и обработка на резултати получени от база данни след обработка на заявка

CC-BY-SA

КОМПОНЕНТИ НА JDBC

- JDBC API – предоставя възможност за свързване към база данни чрез Java. Част е от Java SE и Java EE платформите
- JDBC Driver Management – дефинира обекти, които могат да свързват Java приложение към JDBC driver
- JDBC Test Suite – помага при определяне дали даден driver са подходящи за употреба в едно Java приложение
- JDBC-ODBC Bridge – достъп до база данни от JDBC с ODBC drivers

CC-BY-SA

ВЪВЕДЕНИЕ

- Едно Java приложение използва JDBC API за работа с база данни
- JDBC може да работи с много и различни бази данни
- Комуникацията между JDBC и съответната база данни се осъществява от JDBC Driver

CC-BY-SA

РЕЛАЦИОННИ БАЗИ ДАННИ

- Базите данни са предназначени за съхраняване на информация и следващото и извличане
- Най – простото определение за база данни е представяне на информация като таблици с редове и колони
- Таблица се нарича релация в смисъл като колекция от обекти от еднакъв тип (редове)
- Релационните бази данни следват определени правила, които гарантират, че данните в тях са правилни и винаги могат да бъдат достигнати
- Тези правила се наричат „нормална форма“
  - Първа нормална форма
  - Втора нормална форма
  - ...

CC-BY-SA

РЕЛАЦИОНИ БАЗИ ДАННИ

Employee_Number	First_name	Last_Name	Date_of_Birth	Car_Number
10001	Axel	Washington	28-Aug-43	5
10083	Arvid	Sharma	24-Nov-54	null
10120	Jonas	Ginsberg	01-Jan-69	null
10005	Florence	Wojokowski	04-Jul-71	12
10099	Sean	Washington	21-Sep-66	null
10035	Elizabeth	Yamaguchi	24-Dec-59	null

CC-BY-SA

SQL

- SQL е езикът, който се използва с релационни бази данни
- Съществува множество от основни SQL команди, които се считат за стандартни и могат да се използват от всички релационни бази данни
  - SELECT
  - INSERT
  - DELETE
  - UPDATE
  - CREATE TABLE
  - DROP TABLE
  - ALTER TABLE

CC-BY-SA

SQL - SELECT

- SELECT израз, още наричан заявка, се използва за извличане на данни от таблица
- Могат да се определят от кои колони да се извлекат данни и от кои таблици

```
SELECT FirstName, SecondName FROM Employees;

SELECT * FROM Employees;
```


CC-BY-SA

SQL - WHERE

- WHERE клаузата, определя критерия, върху който ще се изпълни израза
- Ключовата дума LIKE се използва за сравняване на символни низове
- Предлага възможност за сравняване по шаблон

```
SELECT FirstName, SecondName FROM Employees
   WHERE Car_Number = 1;

SELECT FirstName, SecondName FROM Employees
   WHERE FirstName LIKE 'Washington'
```


CC-BY-SA

SQL - WHERE

- % - един или повече символа до края
- _ - само един символ

```
SELECT FirstName, SecondName FROM Employees
   WHERE Car_Number > 2 AND FirstName LIKE 'Washington';

SELECT FirstName, SecondName FROM Employees
   WHERE FirstName LIKE 'Washington%'

SELECT FirstName, SecondName FROM Employees
   WHERE FirstName LIKE 'Ba_man'
```


CC-BY-SA

SQL - WHERE

- % - един или повече символа до края
- _ - само един символ

```
SELECT FirstName, SecondName FROM Employees
   WHERE Car_Number > 2 AND FirstName LIKE 'Washington';

SELECT FirstName, SecondName FROM Employees
   WHERE FirstName LIKE 'Washington%'

SELECT FirstName, SecondName FROM Employees
   WHERE FirstName LIKE 'Ba_man'
```


CC-BY-SA

SQL - JOINS

- Join е възможността да се получат данни от повече от една таблица

```
SELECT Employees.FirstName, Employees.LastName, Cars.Make,
            Cars.Model, Cars.Year
FROM Employees, Cars
WHERE Employees.Car_Number = Cars.Car_Number
```


CC-BY-SA

SQL - INSERT

- INSERT израз се използва за добавяне на нови редове в таблица

```
INSERT INTO Employees VALUES (100312, 'Nenko', 'Tabakov',
null, 1);

INSERT INTO Employees (FirstName, SecondName) VALUES ('Nenko',
'Tabakov');
```


CC-BY-SA

SQL - DELETE

- DELETE израз се използва за изтриване на редове от таблица

```
DELETE FROM Employees WHERE FirstName='Nenko';

DELETE FROM Employees
   WHERE FirstName='Nenko' AND Car_Number=2;
```


CC-BY-SA

SQL - UPDATE

- UPDATE израз се използва за промяна на стойностите на колони

```
UPDATE Employees SET FirstName='Plamen', SecondName='Tanov'
   WHERE FirstName='Nenko' AND SecondName='Tabakov';
```


CC-BY-SA

ПЪРВИ СТЪПКИ

Стъпки при използване на JDBC:

1. Зареждане на driver
2. Установяване на връзка с базата данни
3. Създаване на израз
4. Изпълнение на заявка
5. Получаване на резултат

CC-BY-SA

ПЪРВИ СТЪПКИ

```
Class.forName("org.apache.derby.jdbc.EmbeddedDriver");

Connection con = DriverManager.getConnection(
                 "jdbc:myDriver:wombat",
                 "myLogin", "myPassword");

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery
                  ("SELECT a, b, c FROM Table1");

while (rs.next()) {
    int x = rs.getInt("a");
    String s = rs.getString("b");
    float f = rs.getFloat("c");
}
```


CC-BY-SA

ЗАРЕЖДАНЕ НА DRIVER

За да се зареди определен driver е нужно да се създаде негова инстанция и тя да се регистрира в Driver Manager. Това става със статичния метод forName на класа Class.

```
Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
```

Друг начин Driver Manager да зареди даден driver, е като той бъде описан като jdbc.driver свойство във файла properties

```
jdbc.drivers=org.apache.derby.jdbc.EmbeddedDriver
```


CC-BY-SA

УСТАНОВЯВАНЕ НА ВРЪЗКА

- Класът Driver Manager работи с интерфейса Driver и управлява всички JDBC Drivers, които са достъпни.
- Когато клиентът се опита да установи връзка и предостави URL, DriverManager е отговорен да намери съответния driver и да го използва за установяване на връзка с базата данни
- URL има следната форма:
  jdbc:driverName:<dbName>[propertyList]
- В документацията за всеки driver трябва да се провери какви стойности могат да се използват след jdbc:

```
Connection con =
   DriverManager.getConnection("jdbc:derby:COFFEES");

Connection con = DriverManager.getConnection(
                 "jdbc:derby:COFFEES",
                 "username", "password");
```

ПОЛУЧАВАНЕ НА РЕЗУЛТАТ

•Интерфейсът ResultSet предоставя методи достъп и редактиране на резултати върнати от заявка
•Типът на ResultSet обекта определя нивото на неговата функционалност в две посоки:
- Начинът, по който курсорът може да бъде управляван
- Как едновременните промените на данните от таблицата се отразяват от ResultSet обекта

CC-BY-SA

ПОЛУЧАВАНЕ НА РЕЗУЛТАТ

- Съществуват три типа ResultSet:
  - TYPE_FORWARD_ONLY – курсорът се движи само напред от първия ред към последния
  - TYPE_SCROLL_INSENSITIVE – курсорът може да се движи както напред така и назад относно текущата позиция. Също така може да се движи и по абсолютна позиция. Не отразява промени направени в базата данни, докато се работи с резултата.
  - TYPE_SCROLL_SENSITIVE - курсорът може да се движи както напред така и назад относно текущата позиция. Също така може да се движи и по абсолютна позиция. Отразява промени направени в базата данни, докато се работи с резултата.

CC-BY-SA

ПОЛУЧАВАНЕ НА РЕЗУЛТАТ

```
Statement Connection#createStatement();
Statement Connection#crateStatement
   (int resultSetType, int resultSetConcurency);

Statement stmt = con.createStatement();

Statement stms =
con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                    ResultSet.CONCUR_READ_ONLY);

ResultSet rs = stmt.executeQuery
                 ("SELECT a, b, c FROM Table1");
```


CC-BY-SA

RESULT SET – МЕТОДИ ЗА НАВИГАЦИЯ

- next ();
- previous();
- first();
- last();
- beforeFirst();
- afterLast();
- relative (int rows);
- absolute (int rows);

CC-BY-SA

RESULT SET – МЕТОДИ ЗА ПОЛУЧАВАНЕ НА СТОЙНОСТ

- За извличане на информация ReslutSet дефинира поредица от getXXX() методи за всички примитивни типове:
  - getBoolean
  - getLong
  - getString
  - ...
- Стойности могат да се извличат по индекс на колона или по името й
- Когато се използва име на колона, то името й не зависи от регистъра (case insensitive)
- По – ефективно е да се използва индекс на колона
- Броенето на колони започва от 1

CC-BY-SA

RESULT SET – МЕТОДИ ЗА ПОЛУЧАВАНЕ НА СТОЙНОСТ

```
Statement stms =
   con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                       ResultSet.CONCUR_READ_ONLY);

ResultSet rs = stmt.executeQuery
                  ("SELECT a, b, c FROM Table1");

while (rs.next()) {
   int x = rs.getInt("a");
   String s = rs.getString(2);
   float f = rs.getFloat("c");
}
```


CC-BY-SA

ОБНОВЯВАНЕ НА ТАБЛИЦИ

- JDBC предоставя възможност за редактиране на стойности в една таблица с помощта на Java методи, вместо използването на SQL израз
- Обновяването на стойност става на две стъпки
  - Новата стойност за всяка колона се записва
  - Новите стойности се съхраняват в таблицата
- Никакви промени не се правят в таблицата докато втората стъпка не е изпълнена
- Интерфейсът ResultSet дефинира по две метода за обновяване за всеки тип – по индекс или по име

CC-BY-SA

ОБНОВЯВАНЕ НА ТАБЛИЦИ

- Съхраняването на новите стойности става с метода updateRow();
- За да е възможно обновяване с Java методи е нужно createStatement метода да се извика с константата ResultSet.CONCUR_UPDATABLE

```
Statement stms =
   con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                       ResultSet.CONCUR_UPDATABLE);

ResultSet rs = stmt.executeQuery
                 ("SELECT COF_Name FROM COFFEES
                   WHERE Price = 7.99");


rs.next();
rs.updateString („COF_NAME“, „Foldgers“);
rs.updateRow();
}
```

ОБНОВЯВАНЕ НА ТАБЛИЦИ

- JDBC предоставя възможност за редактиране на стойности в една таблица с помощта на Java методи, вместо използването на SQL израз
- Обновяването на стойност става на две стъпки
  - Новата стойност за всяка колона се записва
  - Новите стойности се съхраняват в таблицата
- Никакви промени не се правят в таблицата докато втората стъпка не е изпълнена
- Интерфейсът ResultSet дефинира по две метода за обновяване за всеки тип – по индекс или по име

CC-BY-SA

PREPARED STATEMENTS

- Когато често се изпращат едни и същи заявки към базата данни е добре да се обмисли дали да не се използват prepared statements
- Prepared statements са заявки, които се предварително компилирани и няма нужда базата данни да ги компилира, когато ги получи
- Като цяло prepared statements са по – бързи и натоварват базата по – малко
- Един prepared statement може да има параметри – могат да се подават различни стойности с всеки израз

CC-BY-SA

PREPARED STATEMENTS

- Всеки параметър трябва да се инициализира преди да се изпрати заявката
- Инициализирането става с setXXX методи, като първият аргумент е колоната, а втория стойността

```
PreparedStatement updateSales = con.prepareStatement(
        "UPDATE COFFEES SET SALES = ? WHERE COF_NAME LIKE ?
");
updateSales.setInt(1, 75);
updateSales.setString(2, "Colombian");
updateSales.executeUpdate():
```


CC-BY-SA

PREPARED STATEMENTS

```
PreparedStatement updateSales;
String updateString = "update COFFEES " +
                      "set SALES = ? where COF_NAME like ?";
updateSales = con.prepareStatement(updateString);

int [] salesForWeek = {175, 150, 60, 155, 90};
String [] coffees = {"Colombian", "French_Roast", "Espresso",
                     "Colombian_Decaf", "French_Roast_Decaf"};

int len = coffees.length;
for(int i = 0; i < len; i++) {
                updateSales.setInt(1, salesForWeek[i]);
                updateSales.setString(2, coffees[i]);
                updateSales.executeUpdate();
        }
```


CC-BY-SA

PREPARED STATEMENTS

excuteUpdate методът връща броя на обновените редове

```
PreparedStatement updateSales = con.prepareStatement(
        "UPDATE COFFEES SET SALES = ? WHERE COF_NAME LIKE ?
");
updateSales.setInt(1, 75);
updateSales.setString(2, "Colombian");
Int n = updateSales.executeUpdate():
System.out.println („Update rows = “ +n);
```


CC-BY-SA

ТРАНЗАКЦИИ

- Транзакция е поредица от заявки, които се изпълняват или всички заедно или нито една
- Това е удобно ако искаме една заявка да се изпълни след изпълнението на друга заявка, а не преди това
- Когато искаме при обновяване на стойностите в една колона задължително да се обновят стойностите и в други колони
- По подразбиране всяка SQL заявка е транзакция в JDBC, т.е. всяка заявка се предава в момента, в който се изпълни (auto commit)
- За да се групират няколко заявки като транзакция е нужно първо да се изключи автоматичното предаване на всяка заявка (no auto cmomit)

CC-BY-SA

ТРАНЗАКЦИИ

```
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);
```


CC-BY-SA

ТРАНЗАКЦИИ

- Методът rollback връща базата данни в състояние, което е била преди изпълнение на commit метода
- От версия JDBC 3.0 API е добавен нов метод Connection.setSavepoint, който има за цел да установи място за възстановяване (save point) в самата транзакция
- Също така методът rollback е предефиниран. Новият метод има един параметър и това е името на мястото за възстановяване

```
Statement stmt = conn.createStatement();
int rows = stmt.executeUpdate("INSERT INTO TAB1 (COL1)
                                    VALUES(?FIRST?)");

Savepoint svpt1 = conn.setSavepoint("SAVEPOINT_1");
rows = stmt.executeUpdate("INSERT INTO TAB1 (COL1)
                                  VALUES (?SECOND?)");

conn.rollback(svpt1);
```


CC-BY-SA