Laboratorium 8

Wstęp

Dostęp do bazy danych:

1. Załadowanie sterownika (drivera) do pamięci np.

   Class.*forName*("org.apache.derby.jdbc.ClientDriver");

2. Podłączenie się do bazy (klasa Connection)

   Connection con = DriverManager.*getConnection*(dbUrl, userName, passwd);

   String dbUrl = konfiguracja połączenia: jdbc:protokół://adres_serwera:port/nazwa_bazy

   Np. String dbUrl ="jdbc:derby://localhost:1527/komunikator";

3. Stworzenie zapytania (klasa Statement):

   ```
   Statement stmt = con.createStatement();
   stmt. executeUpdate(„INSERT INTO data (name) VALUES('Marcin') ")
   …
   stmt.close();
   con.close();
   ```

Typy sterowników:

- Typ 1 – Sterownik który implementuje API JDBC poprzez mapowanie protokołu na inny sterownik dostępu do bazy danych (najczęściej ODBC). Sterowniki tego typu najczęściej bazują na natywnych bibliotekach, które ograniczają ich przenośność. Przykładem sterownika typu 1 jest most JDBC-ODBC

- Typ 2 – Sterowniki częściowo napisane w języku Java oraz częściowo w postaci natywnego kodu. Sterowniki tego typu używają natywnych bibliotek klienta przystosowanych do podłączania się do danego źródła danych. Podobnie jak wyżej ze względu na kod natywny ich przenośność jest ograniczona.

- Type 3 – Sterowniki które używają kodu w pełni napisanego w Javie i komunikują się z serwerem pośredniczącym wykorzystując niezależny od bazy danych protokół. Serwer pośredniczący następnie komunikuje się bezpośrednio ze źródłem danych.

- Type 4 – Sterowniki wykorzystujący kod napisany w pełni w Javie i bazujące na protokole sieciowym uzyskując dostę bezpośrednio do źródła danych.

Wywoływanie zapytań:

- ResultSet rst = stmt.executeQuery("SELECT * FROM APP");

- executeQuery wywoływane jest gdy w wyniku zapytania zwracane są wyniki działania zapytania

- stmt.executeUpdate(„INSERT INTO COFFEES (COF_NAME) (‘Nescafe’)”)

- executeUpdate wywoływane jest gdy w wyniku zapytanie do BD nic nie jest zwracane.

- stmt.execute(„INSERT INTO COFFEES (COF_NAME) (‘Nescafe’)”)

- Execute – wywołanie dowolnego zapytania do BD, jeśli w wyniku zapytania zwracane są jakieś wartości dostęp do nich możliwy jest poprzez getResultsSet() lub getUpdateCount()

Wyniki zapytań – klkasa ResultsSet

Obiekt klasy ResultsSet zwracany jest jako wynik operacji na bazie. Możliwe są 3 typy wyników:

- TYPE_FORWARD_ONLY — Wyników nie można „przewijać”, kursor przewija się jedynie wprzód, od przed pierwszego wiersza do ostatniego +1. Wiersze zawarte w zbiorze wyników zależą od sposobu zwracania wyników przez bazę danych. To znaczy, zawierają wiersze które pasują do zapytania bądź w momencie gdy zapytanie jest wykonywane, lub jak wiersze są pozyskane.

- TYPE_SCROLL_INSENSITIVE — Wyniki są przewijalne; kursor może poruszać się w przód i w tył względem obecnej pozycji, jak i może poruszać się do określonej pozycji absolutnej. Wyniki ogólnie nieczułe na zmiany wprowadzane przez innych

- TYPE_SCROLL_SENSITIVE — Wyniki są przewijalne; kursor może poruszać się w przód i w tył względem obecnej pozycji, jak i może poruszać się do określonej pozycji absolutnej. Czuły na zmiany wprowadzane przez innych

Ponadto można określić dodatkowy parametr wyników:

- CONCUR_READ_ONLY – czy wyniki są tylko read-only

- CONCUR_UPDATABLE – czy wyniki są aktualizowalne

Określenie typu wyników:

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

Poruszanie się po zbiorze wyników

Po stworzeniu obiektu ResultsSet kursor znajduje się przed pierwszym wierszem. Do poruszania się niem służą metody:

- next() – przesuwa kursor w przód o jeden wiersz. Zwraca logiczną 1 jeśli kursor jest na jednym z istniejących wierszy, lub fałsz jeśli przeszedł ostatni wiersz.

- previous() – przesuwa kursor w tył o jeden wiersz, zwraca true jeśli kursor znajduje się na jednym z istniejących wierszy lub false, jeśli jest przed pierwszym wierszem.

- first() – przesuwa kursor do pierwszego wiersza zbioru ResultsSet. Zwraca true jeśli kursor jest na pierwszej pozycji, lub fałsz jeśli zbiór wyników jest pusty

- last() – przesuwa kursor do ostatniego wyniku, zwraca true jeśli kursor zostanie umieszczony na ostatnim wierszu lub fałsz jeśli zbiór wyników jest pusty
- beforeFirst() – Przesuwa kursor do pozycji przed pierwszej.
- afterLast() - przesuwa kursor do pozycji ostatni + 1
- relative(int rows) – Przesuwa kursor do danej pozycji względnej
- absolute(int row) – przesuwa kursor do danej pozycji bezwzględnej

Dostęp do wyników z ResultsSet

W aktualnej pozycji kursora dostęp realizowany jest przez:

getXXX(String colName) lub getXXX(int id)

Gdzie

- colName – nazwa kolumny
- id - numer kolumny
- XXX – nazwa typu danych np. getString(„Name”), getBoolean(2), getByte(„wiek”)

_____

```
Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                     ResultSet.CONCUR_READ_ONLY);
ResultSet srs = stmt.executeQuery(
   "SELECT COF_NAME, PRICE FROM COFFEES");
while (srs.next()) {
      String name = srs.getString("COF_NAME");
      float price = srs.getFloat("PRICE");
      System.out.println(name + "     " + price);
}
```

Aktualizacje wyników

W JDBC można dokonywać aktualizacji przez operacje na obiekcie klasy ResultSet:

Polecenia:

updateXXX(String colName, <XXX> value)

updateXXX(int id, <XXX> value)

updateString, updateByte, updateFloat itp.

Akceptacja wprowadzonych zmian – metoda:

updateRow()

Anulowanie wprowadzonych zmian:

cancelRowUpdates();

Przykład:

```
uprs.last();
uprs.updateFloat("PRICE", 10.99);
uprs.cancelRowUpdates();
uprs.updateFloat("PRICE", 10.79);
uprs.updateRow();
```

Dodanie nowego wiersza do ResultsSet

W JDBC można dodać nowy wiersz przez operacje na obiekcie klasy ResultSet:

Polecenia:

moveToInsertRow() – ustawienie kursora na wolnym miejscu

updateXX(A,B) – patrz poprzedni slajd

insertRow(); - dodanie wiersza do bazy

Np..

```
rsr.moveToInsertRow();
rsr.updateInt(1, 150);
rsr.updateString(2, "Madonna");
rsr.updateString(3, "Dummy");
rsr.updateString(4, "Jazz");
rsr.updateString(5, "Image");
rsr.updateInt(6, 5);
 rsr.updateDouble(7, 5);
rsr.updateInt(8, 15);
rsr.insertRow();
```

Usuwanie wiersza

W JDBC można usuwać wiersze przez operacje na obiekcie klasy ResultSet:

Polecenia:

deleteRow() – usuwa aktualny wiersz.

Zadanie

1) Stwórz program łączący się z bazą sample znajdującą się na serwerze 157.158.131.191 i pobierający z niej z tabeli APP.CUSTOMER listę użytkowników. Pobraną listę należy

wyświetlić podając na ekranie NAME, CITY, CUSTOMER_ID. Dodaj do bazy nowego użytkownika

2) Zbuduj komunikator w oparciu o bazę danych. W tym celu wykorzystaj bazę Komunikator na serwerze 157.158.131.191 urzytkownik student hasło: ZAQ12wsx. Baza zawiera tabelę komunikaty składającą się z atrybutów destinationUserID, message, sourceUserID, messageID oraz tabelę Useres składającą się z pól UserID, UserName. Stworzony program powinien składać się z dwóch wątków, w pierwszym zadawany są ciągłe zapytania do bazy danych (co 2s) i jeśli zostanie odnaleziona wiadomość dla danego użytkownika, należy ją wyświetlić i usunąć. Wyświetlanie i usuwanie wiadomości powinno być realizowane w oparciu (w kolejności) o atrybut messageID numerujący poszczególne komunikaty dla danego użytkownika. Drugi wątek powinien wrzucać wiadomości adresowane dla danego użytkownika do bazy, umieszczając jednocześnie odpowiedni znacznik messageID.