Chat klijent–spremanje u bazu podataka

1. Cilj vježbe

Spremiti poslane i primljene podatke u bazu podataka.

2. Opis

2.1 Rad sa bazom podataka

U Javi se sa bazom podataka komunicira putem JDBC drivera. Putem drivera omogućeno je da se na isti način radi sa bazama podataka bez obzira na vrstu DBA servera. Java API definira set interfacea (java.sql) za rad sa bazom dok proizvođači DBA servera osiguravaju implementaciju. Za rad sa SQL serverom potrebno je skinuti Microsoft implementaciju JDBC drivera:

http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=11774

Iz raspakiranog paketa potrebno je naći sqljdbc4.jar datoteku i kopirati je u lib direktorij eclipse projekta. Nakon toga potrebno je dodati tu jar datoteku na class path (build path->libraries u eclipsu)

Prvo što je potrebno za rad sa bazom je konekcija. Iz konekcije se mogu kreirati ostali objekti koji služe za izvršavanje SQL naredbi. Za kreiranje konekcije zadužena je klasa DriverManager koja putem URL konekcije za spajanje na bazu pokušava kreirati Connection objekt putem prvog drivera koji se može spojiti na traženu bazu. Zbog toga je najprije potrebno DriverManageru specificirati koje JDBC drivere može koristiti. To je najjednostavnije napraviti putem Class.forName metode (metoda učitava definiciju klase putem njenog punog imena) npr.:

- MySQL: `Class.forName("com.mysql.jdbc.Driver").newInstance();`
- Postgres: `Class.forName("org.postgresql.Driver");`
- SQL server: `Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");`

Za kreiranje konekcije prema SQL serverom potreban je slijedeći kod:

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
String connectionUrl = "jdbc:sqlserver://localhost;database=NazivBaze;
user=userName;password=password;";
Connection con = DriverManager.getConnection(connectionUrl);
```

Ako nije drukčije specificirano JDBC driver se pokušava spojiti na port 1433 SQL servera. U Sql server configuration Manageru pod SQL Server Network Configuration->Protocols for SQLEXPRESS mora biti omogućeno spajanje putem TCP/IP-a. U konfiguraciji TCP/IP protokola na tabu IP Addresses pod IPAll->TCP port treba upisati 1433 i restartati sql server.

Nakon kreiranja konekcije rad sa bazom je isti bez obzira o DBA serveru. Statemenet je interface koji predstavlja SQL izraz. Statement objekt se izvršava i vraća ResultSet objekt koji predstavlja rezultate koje je vratila baza. Statement objekt se kreira iz Connection objekta:

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

Postoje tri različita interface za rad sa SQL izrazima:

- Statement – koristi se za zadavanje jednostavnih SQL izraza bez parametara
- PreparedStatement (nasljeđuje Statement) – koristi se za prekompajlirane SQL izraze koji mogu sadržavati parametre
- CallableStatement: (nasljeđuje PreparedStatement) – koristi se za izvršavanje procedura koje mogu sadržavati ulazne i izlazne parametre

Da bi se izvršio SQL izraz treba se koristiti jednu od slijedećih metoda definiranih u Statement interfacu:

- execute: izvršava sql izraz i vraća true ako je DBA server vratio rezultat u obliku ResultSet objekta. (vraća se false kod update, delete i insert naredbi)
- executeQuery: izvršava sql upit i vraća rezultat u obliku ResultSet objekta
- executeUpdate: vraća broj promijenjenih redaka sa SQL izrazom. Koristi se kod update, insert i delete naredbi.

```
ResultSet rs = stmt.executeQuery(query);
```

Podacima koji su rezultat SQL izraza (select naredbe) se pristupa putem kursora ResultSet objekta. Kursor je pokazivač na redak podataka u ResultSet objektu. Inicijalno je kursor je pozicioniran prije prvog retka. Raznim metodama ResultSet objekta se taj kursor može pomicati na odgovarajuću poziciju (next, previous, first, last, beforeFirst, afterLast, relative(int rows), absolute(int row)). Najčešće se koristi metoda next koja pomiče kursor za jednu poziciju naprijed i vraća true ako kursor pokazuje na redak podataka ili false ako je kurosr pozicioniran poslije zadnjeg retka (nema više podataka za dohvat). Kada je kursor pozicioniran na neki redak podataka sa setom getter metoda mogu se dohvatiti vrijednosti kolona. Getter metode se razlikuju po tipu podatka koji se vraća npr getBoolean, getLong, … Vrijednosti kolona se mogu dohvatiti po indeksu kolone (počinje od broja 1) ili po aliasu kolone u select naredbi (dohvat po indeksu je efikasnije od dohvata po aliasu):

```
Statement stmt = null;
    String query =
        "select COF_NAME, SUP_ID, PRICE, " +
        "SALES, TOTAL from COFFEES";

    try {
        stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery(query);
        while (rs.next()) {
            String coffeeName = rs.getString(1);
            int supplierID = rs.getInt(2);
            float price = rs.getFloat(3);
            int sales = rs.getInt(4);
            int total = rs.getInt("TOTAL");
            System.out.println(coffeeName + "\t" + supplierID +
```

```
                                "\t" + price + "\t" + sales +
                                "\t" + total);
        }
    } catch (SQLException e ) {
        e.printStackTrace();
    } finally {
        if (stmt != null) { stmt.close(); }
    }
```

Osim Statement interfacea za rad sa bazom se može koristiti i PreparedStatement. PreparedStatement se koristi, zbog performansi, kada se treba više puta ponoviti isti SQL izraz. Za razliku od Statement-a kod PreparedStatementa se SQL izraz treba definirati prilikom kreiranja i najčešće se odmah šalje bazi da se kompajlira. Kod PreparedStatement-a se u SQL izrazi mogu definirati i parametri pa se isti SQL izraz može više puta izvoditi sa različitim vrijednostima. U SQL izrazu se koristi „?“ kao pozicija na koju se stavlja vrijednost parametra. Prije izvršavanja PreparedStatementa (putem metoda execute, executeQuery, executeUpdate metoda koje ne primaju nikakav parametar) moraju se postaviti vrijednosti parametara. Vrijednosti se postavljaju setom setter metoda, svaka za drugi tip podatka (setInt, setString, ...)

```
    PreparedStatement updateSales = null;
    String updateString =
        "update COFFEES " +
        "set SALES = ? where COF_NAME = ?";

    try {
        updateSales = con.prepareStatement(updateString);
                updateSales.setInt(1, 150);
                updateSales.setString(2, "JAVA");
                updateSales.executeUpdate();
    } catch (SQLException e ) {
                e.printStackTrace()
    } finally {
        if (updateSales != null) {
            updateSales.close();
        }
    }
```

Na kraju korištenja potrebno je otpustiti alocirane resurse pozivom metode close() na objektima ResultSet-a, Statement-a i Connection-a.

Zadatak: Potrebno je u bazi kreirati tablicu u koju će se spremati poslane i primljene poruke u chat klijentu (Id, user, text). Kod pokretanja chat aplikacije potrebno je kreirati Connection objekt a iz njega PreparedStatement objekt koji će služiti za izvršavanje sql naredbi za spremanje poslanih i primljenih podataka. Kod pokretanja potrebno je i učitatati sve podatke koji se nalaze u tablici i prikazati ih u JTextArea komponenti.