Programski Sistemi

Java i baze podataka

JDBC

- Pristup bazama podataka u Javi obavlja se preko API-ja JDBC
- JDBC je specifikacija koja obezbeđuje sve što je potrebno za rad za bazom (povezivanje, izvršavanje SQL upita, prikaz rezultata itd.)
- JDBC obezbeđuje standardan API, a proizvođači DBMS sistema daju drajvere koji služe kao stvarna veza između Java aplikacije i baze.

JDBC drajveri

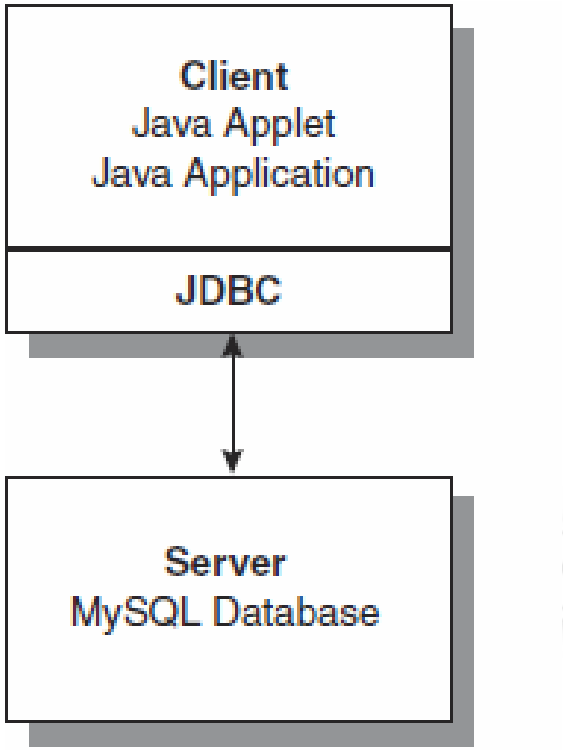
- Skup klasa koje implementiraju interfejse iz paketa java.sql
- Pošto je i većina JDBC drajvera takođe programirana u Javi, oni se mogu koristiti na različitim platformama
- Ne samo da se može menjati platforma na kojoj se aplikacija izvršava ili na kojoj se nalazi baza, već se može menjati i platforma na kojoj se baza izvršava

JDBC i ODBC

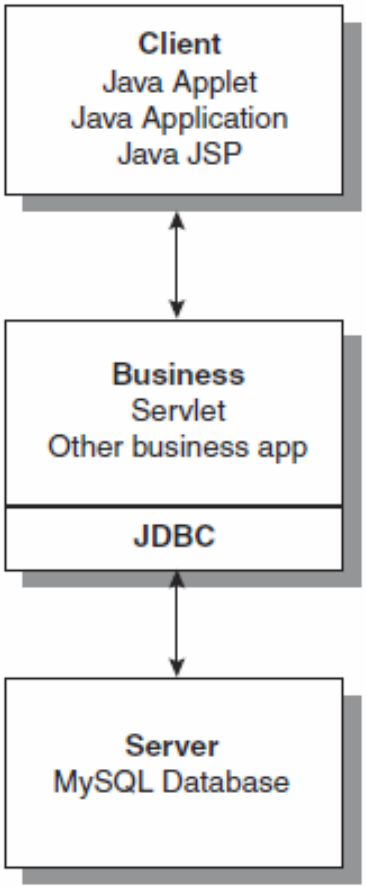
- ODBC (Open Database Connectivity) je Microsoftov API za pristup bazama
- U vreme kada je JDBC bio u povoju, koristio se tzv. JDBC-ODBC bridge koji je Java komande iz JDBC API-ja slao u ODBC drajver, koji ih je zatim izvršavao
- Sada više nema nikakve potrebe da se ODBC koristi

Dvoslojni (klijent-server) model

Aplikacija direktno komunicira sa bazom podataka preko JDBC drajvera



Troslojni model

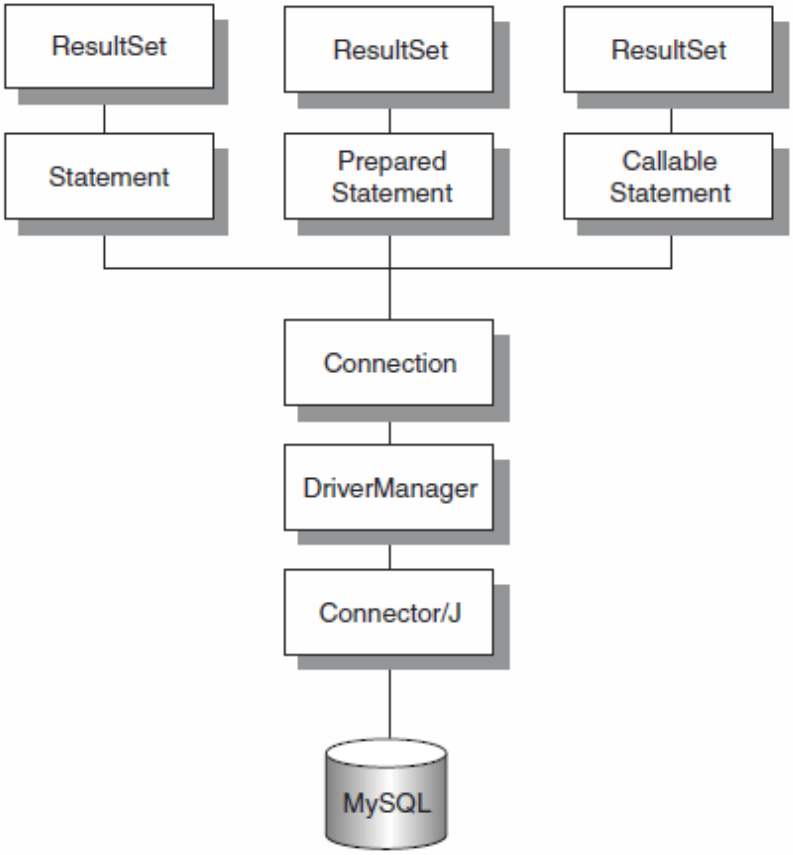


JDBC verzije

- Prva verzija pojavila se još 1997
- Trenutno aktuelna verzija u Javi 1.7 je 4.1 i podržava SQL standard SQL99
- Ceo API je implementiran u paketu **java.sql**
- MySQL drajver se zove Connector/J i može se preuzeti sa adrese http://dev.mysql.com/downloads/connector/

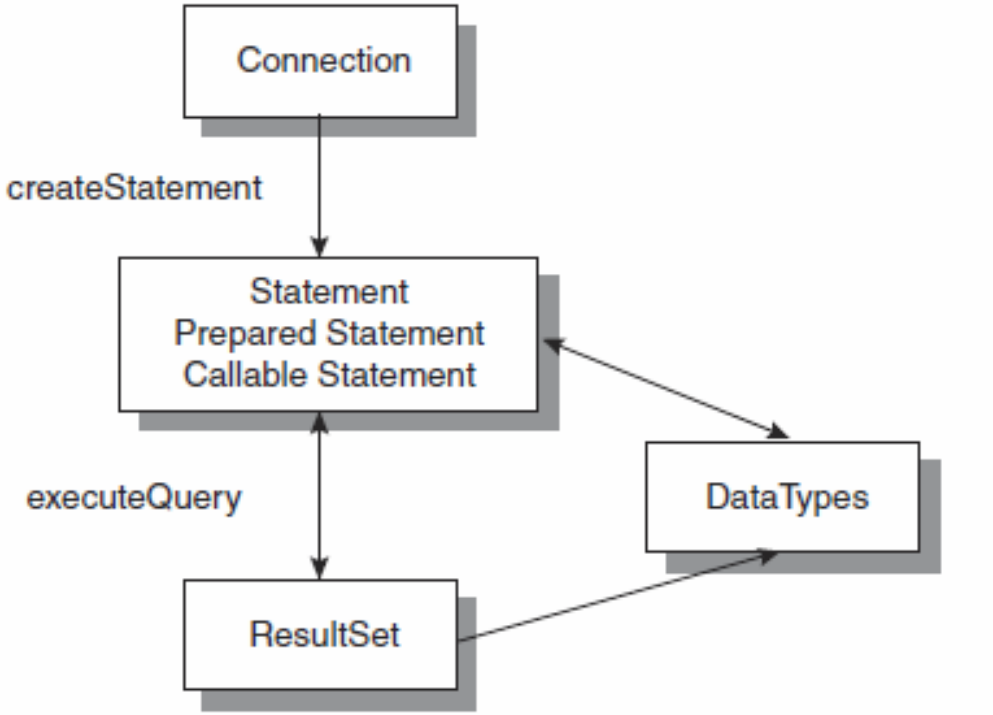
JDBC API: osnovna struktura

- Implementirana u paketu **java.sql**



JDBC API: opciona struktura

- Implementirana u paketu javax.sql



Inicijalizacija

- JDBC URL ima sledeću strukturu: jdbc:<subprotocol>:<name>
- Subprotocol je ime određene vrste mehanizma pristupa bazi
- Sadržaj i sintaksa dela *name* zavisi od potprotokola
  - //<host>[:<port>][/<databaseName>]
- JDBC mora da zna koji drajver je raspoloživ i koji drajver se koristi

Instaliranje drajvera

- Drajver za bazu treba smestiti u direktorijum $JAVA_HOME/jre/lib/ext i on će automatski biti dostupan svim aplikacijama koje izvršava virtuelna mašina

```
public class ProbajVezu {
public static void main(String[] args) {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            System.out.println("OK");
        } catch (ClassNotFoundException e) {
            System.out.println("JDBC Driver error");
        }
    }
}
```

Metod Class.forName()

- Dinamički učitava Java klasu tokom izvršavanja
- JVM koristi classpath tekućeg sistema da pronađe klasu koja se metodu prosleđuje kao parametar
- Ovde sistem pokušava da pronađe drajversku klasu koja se nalazi u paketu **com.mysql.jdbc**

Klasa DriverManager

- Drajver sam sebe registruje pomoću statičke klase **DriverManager** koja je odgovorna za upravljanje svim JDBC drajverima instaliranim u sistemu
- Pošto DriverManager može da radi sa više JDBC drajvera, moguće je programirati Java aplikaciju koja se povezuje sa različitim DBMS sistemima
- Samo učitavanje drajvera za bazu ne otvara konekciju sa njom

JDBC: osnovne klase

- **java.sql.DriverManager**
  - Učitava drajver baze podataka i omogućava podršku za uspostavljanje nove konekcije
- **java.sql.Connection**
  - Predstavlja konekciju sa određenom bazom podataka

JDBC: osnovne klase

- **java.sql.Statement**
  - Kontejner za izvršavanje SQL naredbi u okviru uspostavljene konekcije
- **java.sql.ResultSet**
  - Kontroliše pristup rezultatima dobijenim izvršavanjem određene SQL naredbe

JDBC: osnovne klase

- **java.sql.Statement**: interfejs sa dva važna podtipa:
  - **java.sql.PreparedStatement**
    - Za izvršavanje prekompajlirane SQL naredbe
  - **java.sql.CallableStatement**
    - Za pozivanje stored procedure u okviru baze podataka

Uspostavljanje veze

```
import java.sql.*;

public class ProbajVezu {
public static void main(String[] args) {
        try {
            Class.forName("com.mysql.jdbc.Driver");//ucitavanje drajvera
            Connection conn = DriverManager.getConnection
                    ("jdbc:mysql://3dme.dir.rs:3306/evidencija_prijava", "student", "student");
        } catch (ClassNotFoundException e) {
            System.out.println("JDBC Driver error");
        } catch (SQLException ex) {
            System.out.println("Connection error");
    }
    }
}
```

Korišćenje klase Properties

```
Properties prop = new Properties();
prop.setProperty("user", "student");
prop.setProperty("password", "student");
Connection conn = DriverManager.getConnection
        ("jdbc:mysql://3dme.dir.rs:3306/evidencija_prijava", prop);
```

```
Properties dbProps = new Properties();
ClassLoader loader = Thread.currentThread().getContextClassLoader();
InputStream stream = loader.getResourceAsStream("baza/db.properties");
try {
    dbProps.load(stream);
} catch (IOException ex) {
    System.out.println("problem loading properties file");
}
Connection conn = DriverManager.getConnection
        ("jdbc:mysql://3dme.dir.rs:3306/evidencija_prijava", dbProps);
```

Korišćenje objekta ResultSet

- Primarni mehanizam za vraćanje zapisa iz baze
- Konceptualno podseća na prilagodljiv dvodimenzionalni niz
- Kolone objekta su zapisi iz baze, onako kako je zadato u SQL naredbi
- Kada se ResultSet instancira, interni pokazivač je postavljen pre prvog zapisa u skupu

Izvršavanje SQL naredbi

```
Statement statement = conn.createStatement();
ResultSet rs = statement.executeQuery("SELECT * from student");
while(rs.next()) {
    System.out.println(rs.getInt(1));
    System.out.println(rs.getString(2));
}
rs.close();
statement.close();
conn.close();
```

```
while(rs.next()) {
    System.out.println(rs.getInt("id"));
    System.out.println(rs.getString("ime"));
}
```

Izvršavanje SQL naredbi

- Ako baza podataka nije u stanju da izvrši SQL naredbu, biće bačen izuzetak tipa **SQLException**
- Ako je naredba uspešna, metod **executeQuery()** vratiće objekat **ResultSet** sa rezultatima iz baze

SQLException

- Ovaj izuzetak može da se desi iz mnogo razloga, između ostalog:
  - Konekcija sa bazom više nije validna
  - SQL naredba sadrži sintaksne greške
  - Trenutno prijavljeni korisnik nema dozvolu za tabelu koja se koristi u SQL naredbi
- Odluka o tome od kojih grešaka će se pokušati oporavak zavisi od dizajna aplikacije

Zatvaranje objekata

- Kada se završi rad, objekti ResultSet, Statement i Connection treba da se zatvore da bi JVM i drajver oslobodili memoriju koju su zauzimali
- Zatvaranje se obavlja u obrnutom redosledu od otvaranja: prvo ResultSet, a zatim Statement, i na kraju Connection

Naredbe INSERT i DELETE

```
Statement statement = conn.createStatement();
String query = "INSERT INTO student(id, ime, prezime, indeks, rfid) VALUES (3, 'Nenad', 'Nenadovic', '2008202040', '122')";
statement.executeUpdate(query);
statement.close();
```

```
String query = "DELETE FROM student WHERE id=3";
statement.executeUpdate(query);
```

Izvršavanje istih SQL naredbi više puta uzastopno

```
PreparedStatement stmt = conn.prepareStatement(
        "INSERT INTO student(id,ime,prezime,indeks, rfid) values (?, ?, ?, ?, ?)");
stmt.setInt(1, 3);
stmt.setString(2, "Marko");
stmt.setString(3, "Markovic");
stmt.setInt(4, 2008123);
stmt.setInt(5, 220);
stmt.executeUpdate();
stmt.setInt(1, 4);
stmt.setString(2, "Nenad");
stmt.setString(3, "Nenadovic");
stmt.setInt(4, 2008567);
stmt.setInt(5, 250);
stmt.executeUpdate();
stmt.close();
```

Stored procedure

- Sačuvana procedura je niz SQL naredbi snimljenih u bazi; njih mogu da koriste svi SQL korisnici ili programeri, uključujući i JDBC developere.
- Sačuvane procedure se koriste iz istog razloga kao pripremljene naredbe, tj. zbog efikasnosti i pogodnosti

Stored procedure

- Način na koji se stored procedura čuva u bazi zavisi od konkretnog DBMS sistema
- Za pozivanje stored procedura služi klasa Callable Statement

```
CallableStatement cs = connection.prepareCall("{ call ListDefunctUsers }");
ResultSet rs = cs.executeQuery( );
```

Transakcije

- Skup SQL naredbi se može grupisati u transakciju.
- Ako sve prođu dobro, za transakciju se izvršava commit; ako se desi greška, izvršava se rollback, tj. poništava se dejstvo i onih komandi koje su dobro prošle pre nego što se desila greška

Transakcije

- Najvažniji razlog za uvođenje transakcija jeste održavanje integriteta baze
- Baza je standardno u režimu autocommit, odnosno nakon svake uspešno izvršene SQL komande radi se commit sa trajnim efektima
- Automatski commit menja se naredbom

  conn.setAutoCommit(false);

Transakcije

- Objekat Connection pravi se na uobičajen način; zatim se proizvoljan broj puta poziva executeUpdate()
- Kada se sve naredbe u transakciji izvrše, potrebno je pozvati metod commit

```
conn.commit();
```

- Ako se desila greška, potrebno je pozvati

```
conn.rollback();
```

- Rollback se obično poziva u catch bloku uhvaćenog izuzetka SQLException