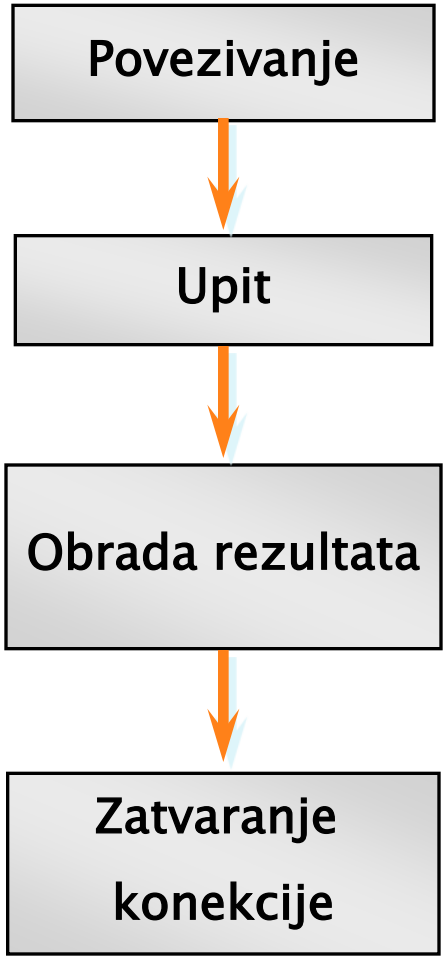
Razvoj Web aplikacija

JDBC

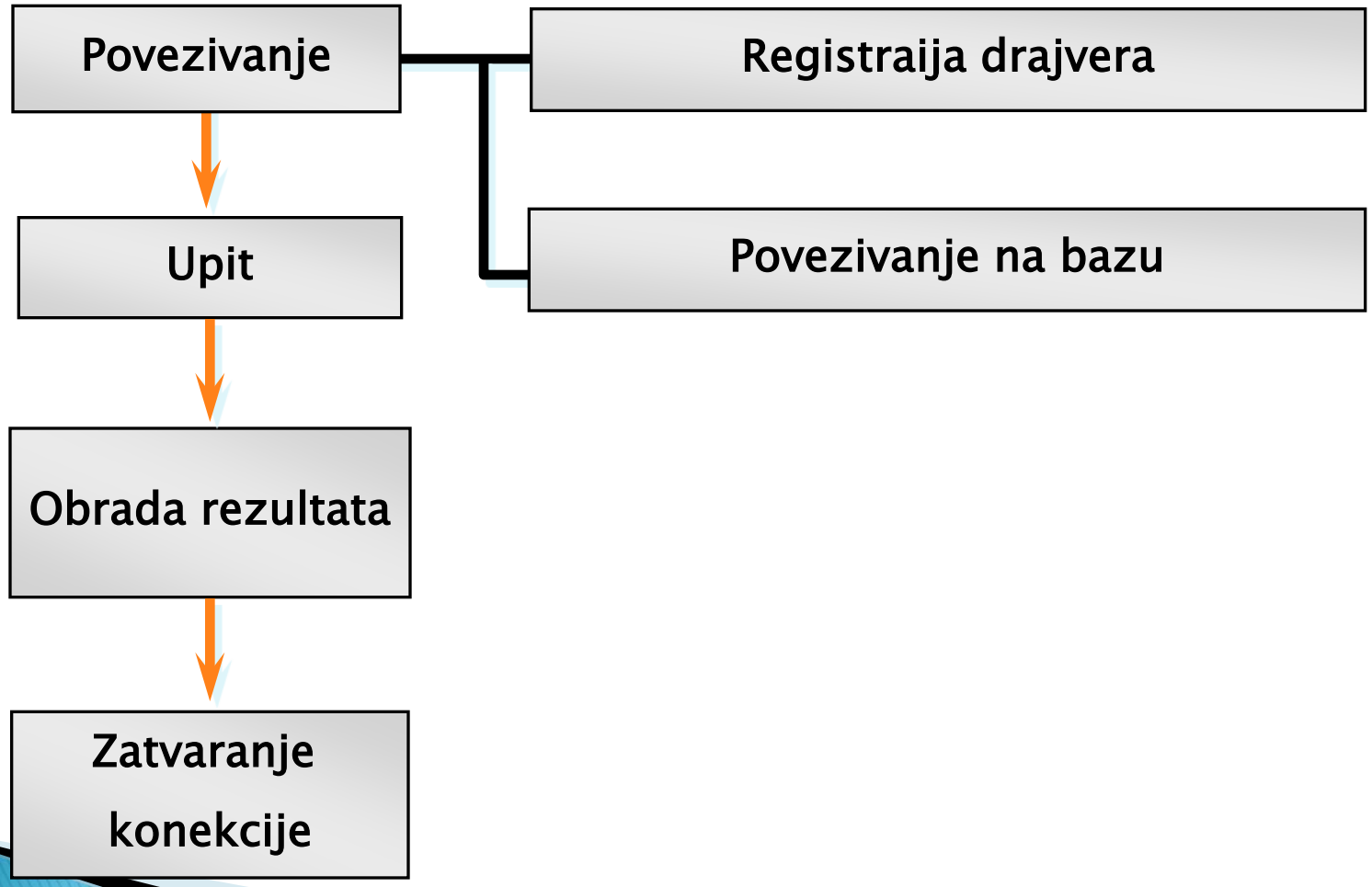
JDBC

- JDBC (**J**ava **D**ata**b**ase **C**onnectivity) je standardni interfejs za povezivanje na relacione baze podataka iz Jave.
- JDBC klase i interfejsi se nalaze u java.sql paketu.

Proces dobijanja podataka

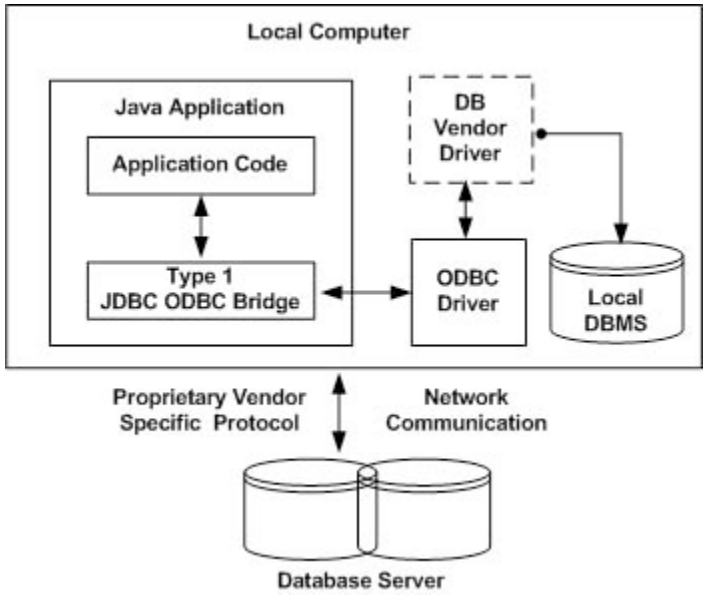


Prvi korak – povezivanje



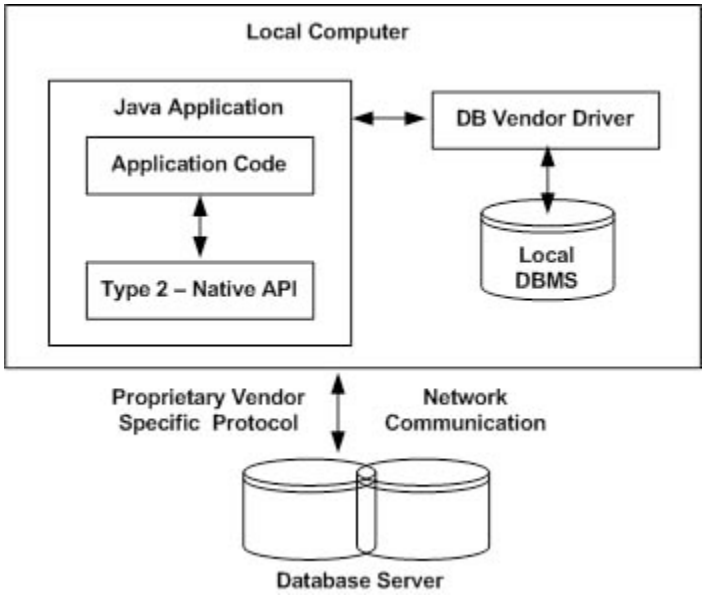
Tipovi JDBC drajvera

- JDBC Type 1 Driver – JDBC/ODBC Bridge drivers
  - ODBC (Open Database Connectivity) je standardni API nezavisan od programskog jezika.



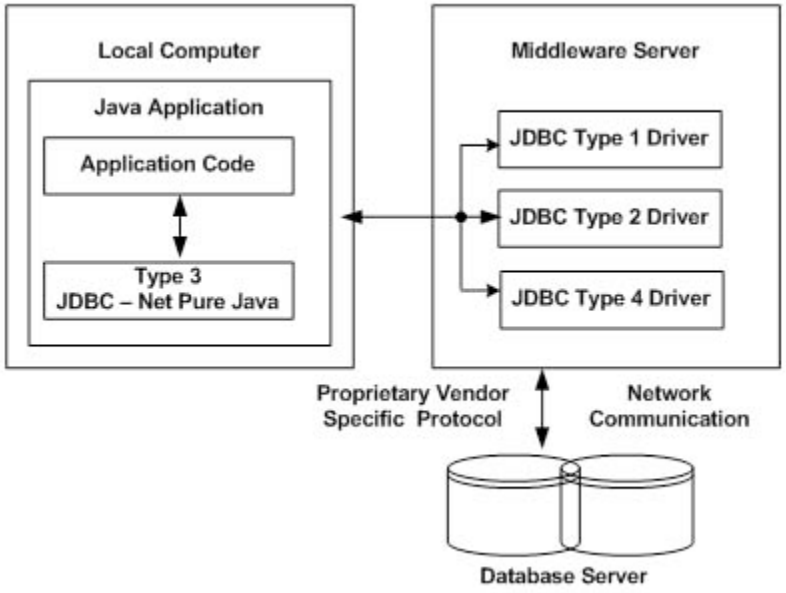
Tipovi JDBC drajvera

- JDBC Type 2 Driver – JDBC-Native API
  - JDBC API pozivi se prevode u pozive (C/C++) koji su specifični za bazu podataka.



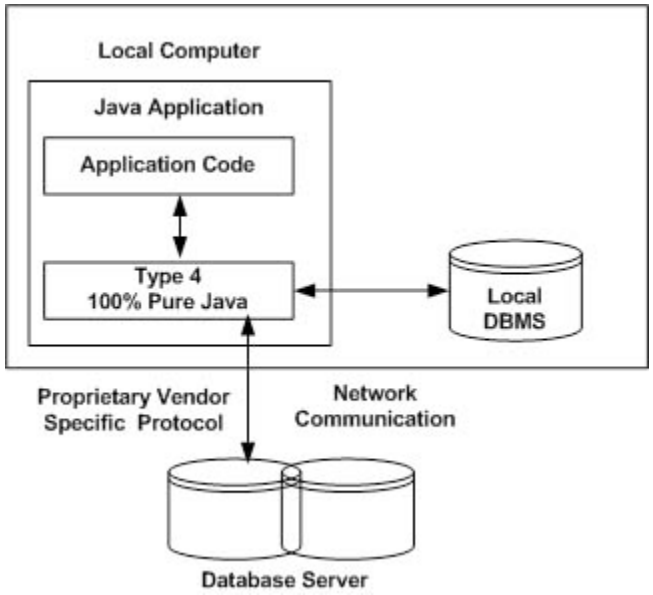
Tipovi JDBC drajvera

- JDBC Type 3 Driver – JDBC-Net pure Java
  - JDBC klijenti komuniciraju sa middleware serverom kroz standardne mrežne socket – e, a server dalje komunicira sa bazom.



Tipovi JDBC drajvera

- JDBC Type 4 Driver – 100% Pure Java
  - Java drajver komunicira direktno sa bazom podataka.



Baze i JDBC drajveri

Baza podataka	JDBC driver	Klasa
MySQL	Connector/J	com.mysql.jdbc.Driver
Oracle	OracleThin	oracle.jdbc.driver.OracleDriver
MSSQL	MSSQL(MicrosoftDriver)	com.microsoft.jdbc.sqlserver.SQLServerDriver
...		

Prvi korak – povezivanje

- Učitavanje odgovarajućeg drajvera
  - Class.forName("oracle.jdbc.driver.OracleDriver");
    - Statički metod Metod forName() vraća kao rezultat Class objekat za dato ime klase.
    - Ukoliko klasa nije pronađena dešava se ClassNotFoundException.
  - Alternativno, DriverManager.registerDriver()

Registracija drajvera – primer

```
try {
  Class.forName("oracle.jdbc.driver.OracleDriver");
} catch (ClassNotFoundException ex) {
  System.out.println("Error: unable to load driver
    class!");
}
```

```
try {
  Driver myDriver = new
    oracle.jdbc.driver.OracleDriver();
  DriverManager.registerDriver(myDriver);
} catch(ClassNotFoundException ex) {
    System.out.println("Error: unable to load driver
    class!");
}
```

Kreiranje konekcije

- Za konekciju na bazu je potreban URL do baze, kao i dodatni parametri.
- Oni se prosleđuju DriverManager – u kroz Connection String.

```
Connection conn = DriverManager.getConnection(cs)
```

- Format URL-a
  - jdbc:mysql://[***host***][,***failoverhost***...][:***port***]/[***database***] [?***propertyName1***] [=***propertyValue1***] [&***propertyName2***][=***propertyValue2***]...

Parametri URL-a za konekciju

Parametar	Opis	Podrazumevano
user	Korisničko ime za bazu	
password	Lozinka	
connectTimeout	Vreme isteka (u milisekundama), 0 za neograničeno vreme.	0
autoReconnect		false
maxReconnects		3
useUnicode		true
characterEncoding		autodetect
...		

http://dev.mysql.com/doc/connector-j/en/connector-j-reference-configuration-properties.html

Drugi korak – upit

- Kreiranje Statement objekta

```
Statement st = conn.createStatement();
```

```
PreparedStatement st = conn.prepareStatement(query);
```

- Izvršenje upita
  - ResultSet executeQuery(String query)
    - Za izvršenje Select query – ja
  - int executeUpdate(String query)
    - Za Insert, Update i Delete query – je.
  - Može da se desi SqlException

Treći korak – obrada rezultata

- Rezultat izvršenja upita je najčešće ResultSet.
- ResultSet objekat omogućava pristup tabeli koja je generisana kao rezultat upita.
- Samo jedan ResultSet po Statement objektu može biti otvoren u jednom trenutku.
- ResultSet vodi računa o pozicioniranju kursora na trenutni red.

ResultSet – metodi

- boolean next()
  - pozicionira kursor na sledeći red
  - prvi poziv postavlja kursor na prvi red
  - vraća false kada nema više redova
- void close()
  - uništava ResultSet objekat
  - omogućava novo korišćenje Statement objekta

ResultSet – metodi

- Integer getInt(int columnIndex)
- String getString(int columnIndex)
- ...
  - Vraća vrednost odgovarajuće kolone iz trenutnog reda kao dati tip. Ukoliko vrednost nije datog tipa, dešava se SQLException.
  - Indeksi kolona počinju od 1, ne od 0!
- Integer getInt(String columnName)
  - Isto kao i prethodni, samo po imenu kolone.
  - Manje efikasan metod od prethodnog.

Mapiranje tipova

SQL type	Java Type
CHAR, VARCHAR, LONGVARCHAR	String
NUMERIC, DECIMAL	java.math.BigDecimal
BIT	boolean
TINYINT	byte
SMALLINT	short
INTEGER	int
BIGINT	long
REAL	float
FLOAT, DOUBLE	double
BINARY, VARBINARY, LONGVARBINARY	byte[]
DATE	java.sql.Date
TIME	java.sql.Time
TIMESTAMP	java.sql.Timestamp

Null vrednosti

- U SQL-u, null znači da je polje prazno.
- Razlikuje se od 0 i “”
- U JDBC, mora eksplicitno da se proveri
  - ResultSet.wasNull()
  - Mora prvo da se pozove geter za kolonu, pa da se onda pita da li je vrednost bila null.

ResultSetMetaData

- Objekat koji može da se iskoristi za dobijanje informacija o kolonama.
- Korisni metodi
  - getColumnCount()
  - getColumnName()
  - getColumnType()
  - getTableName()
  - ...
- Takođe postoji i DatabaseMetaData koji se doija iz konekcije.

Četvrti korak – oslobađanje resursa

- Klase Connection, Statement i ResultSet imaju metod close() koji oslobađa zauzete resurse.
- Prvo se zove close() za ResultSet, zatim za Statement i na kraju za Connection.

Transakcije

- Transakcije se ne započinju i završavaju eksplicitno.
- Konekcija ima stanje AutoCommit.
- Ako je AutoCommit postavljen na true, svaka naredba se automatski komituje.
- Ako je AutoCommit false, svaka naredba se dodaje u trenutnu transakciju.
- AutoCommit je podrazumevano postavljen na true.

Transakcije

- Kada se AutoCommit postavi na false, korisnik mora eksplicitno da potvrdi transakciju sa Connection.commit() ili da odustane od nje sa Connection.rollback().
- Napomena: DDL naredbe (npr. kreiranje i brisanje tabela) mogu da budu ignorisane ili da prouzrokuju komit.
  - Ponašanje zavisi od implementacije DBMS – a.

Pristup ResultSet – u

- Definiše se pri kreiranju Statement objekta
  - Statement createStatement(int resulSetType, int resultSetConcurrency)
- resultSetType
  - ResultSet.TYPE_FORWARD_ONLY
    - kursor može da se pomera samo unapred.
    - kada next() vrati false, podaci više nisu dostupni.
  - ResultSet.TYPE_SCROLL_INSENSITIVE
    - kursor se pomera proizvoljno
    - promene podataka u bazi se ne vide posle prebacivanja podataka u memoriju.
  - ResultSet.TYPE_SCROLL_SENSITIVE
    - kursor se pomera proizvoljno
    - promene podataka u bazi se vide i posle prebacivanja podataka u memoriju.

Pristup ResultSet – u

- resultSetConcurrency
  - ResultSet.CONCUR_READ_ONLY
    - Podaci mogu samo da se čitaju iz baze.
  - ResultSet.CONCUR_UPDATEABLE
    - Podaci mogu i da se menjaju.
- Napomena: Postavljanje vrednosti ova dva parametra ne znači da će i drajver da podržava zahtevano ponašanje, a moguće je da i ako podržava, ima ograničenja.
  - Upit tipa “SELECT * FROM …” neće podržavati promene podataka, već moraju da se navedu imena kolona.