LINKgroup

Distance Learning System

Core Java Programming

Upravljanje podacima JDBC

RDBMS sistemi

- RDBMS (Relational Database Management System) je sistem za upravljanje bazom podataka koji direktno komunicira sa nekom bazom i njenim korisnicima
- Tržište poznaje veliki broj RDBMS sistema, i većinu možemo koristiti pomoću Jave
- Najčešće korišćeni RDBMS sistemi u kombinaciji sa Javom su: Oracle, MySql, Postgres, MS SqlServer
- Svakako, osim RDBMS-a, Java poznaje i druge vidove skladištanje podataka (indekseri, keš sistemi, key value store-ovi...)
- Java SE prilikom instaliranja i Java DB (implementaciju Apache Derby baze podataka)

Koncepti rada sa bazom

- Kod rada sa bazom podataka prepoznajemo koncepte:
  - Konektovani koncept (pesimističan)
    - Perzistentna konekcija
      - Aplikacija je konektovana na bazu podataka sve vreme od aktivacije, do deaktivacije
    - Konekcija po potrebi
      - Aplikacija se konektuje na bazu, obavlja poslove koje ima, a zatim se diskonektuje sa baze
  - Diskonektovani koncept (optimističan)
    - Aplikacija se kači na bazu, preuzima podatke, diskonektuje se, a zatim se, nakon ažuriranja podataka, ponovo konektuje na bazu i vrši ažuriranje
      - Objektno mapiranje

Koraci u radu sa bazom

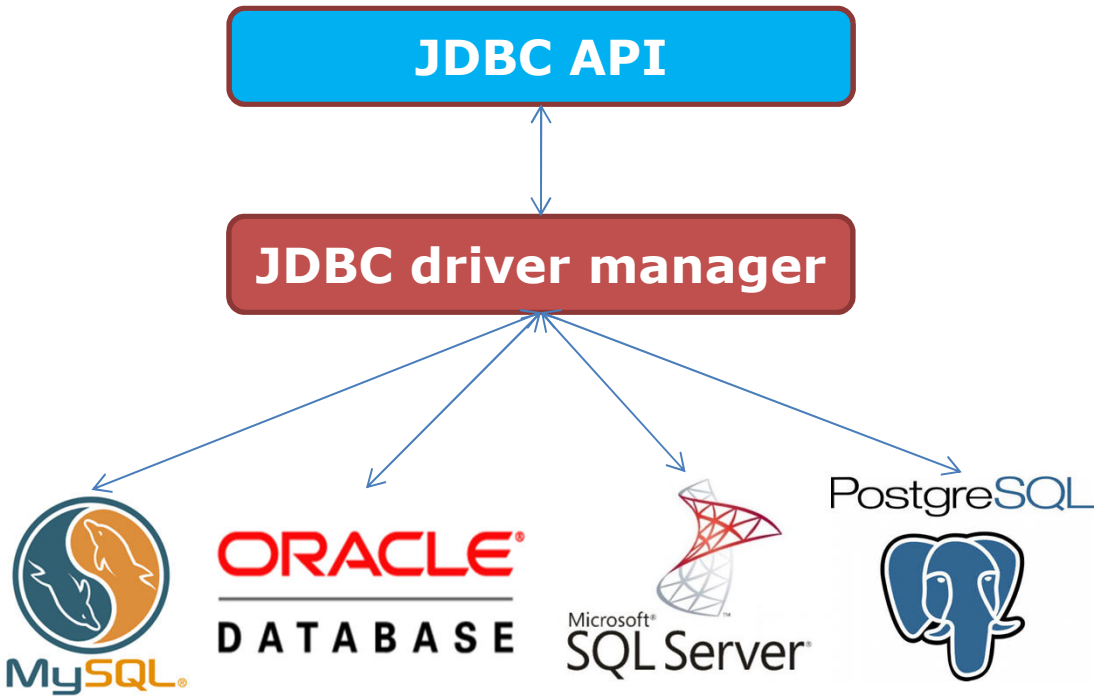
- U radu sa bazom podataka, najčešće se moraju ispratiti sledeći koraci:
  - Konekcija
    - Može biti pulovana ili svaki put nova
  - Aktivacija jednog ili serije upita u bazi
  - Prekid konekcije

Java i baze podataka

- Osnovni sistem za rad sa bazama podataka u javi je JDBC (**Java DataBase Connectivity**)
- U Javi se u sklopu JavaSE dobija i **Java DB** sistem za upravljanje podacima. Ovaj sistem je dobar za programe koji nemaju više simultanih korisnika i ne zahtevaju velike performanse
- Podrazumevano, JavaDB se aktivira zajedno sa java aplikacijom ali je moguće koristiti JavaDB i u formi posvećene aplikacije (zasebnog servera), kada se može koristiti i konkurentno
- Za ozbiljnije aplikacije, sa većim brojem korisnika ili konkurentnih konekcija, koriste se drugi rdbms sistemi (mysql, postgres, sql server, oracle...)

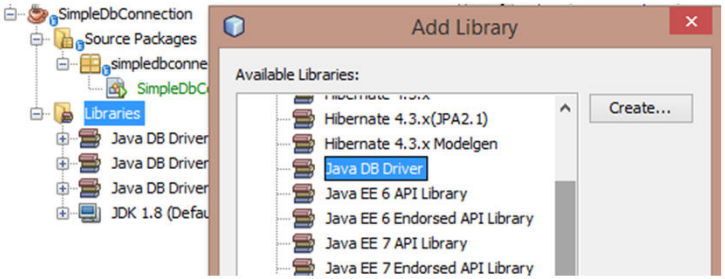
JDBC

- JDBC je sistem koji nam Java omogućava kako bi nam olakšala upravljanje bazom podataka
  - Osnovni koncept ovog sistema je transparentnost – ne moramo se fokusirati na pozadinski sistem već možemo da se fokusiramo na same podatke
- Većina komponenti za rad sa bazom, nalazi se u paketu **java.sql**



Učitavanje drajvera i ostvarivanje JDBC konekcije (jcex142014 SimpleDbConnection)

- Prvi korak u radu sa bazom je učitavanje biblioteke, a zatim I drajvera (samo za ranije verzije Jave) za željeni izvor podataka:



Nije obavezno u novijim verzijama Jave (>7)

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

- Konektovanje na izvor podataka, vrši se pomoću metode **getConnection**, klase **DriverManager** (ili klase **DataSource** u naprednom korišćenju)

```
Connection conn = DriverManager.getConnection("jdbc:derby:mydatabase;create=true");
System.out.println(conn.isClosed());
```

```
run:
false
BUILD SUCCESSFUL (total time: 0 seconds)
```

Kreiranje strukture baze

- Nakon uspešnog konektovanja na bazu, možemo komunicirati sa njom iz aplikacije, pomoću kombinacije metoda JDBC API-ja i SQL-a
- Da bi mogli da koristimo bazu podataka, ona mora imati neku strukturu
- Struktura baze obično podrazumeva tabele
- Strukturu baze možemo kreirati programabilno, ili pomoću neke aplikacije za menadžment
- Za kreiranje strukture koriste se DDL naredbe SQL jezika.

```
Statement st = conn.createStatement();
String db_structure = "create table hiscores ("+
                      "id int primary key "+
                      "GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1),"+
                      "name varchar(128),"+
                      "score int"+
                      ")";
st.execute(db_structure);
```

- Ovaj program uspešno možemo startovati samo jednom. Svaki sledeći put, tabela će biti već kreirana i doći će do greške pa sve tri linije treba zakomentarisati nakon uspešnog startovanja

```
Exception in thread "main" java.sql.SQLException: Table/View 'HISCORES' already exists in Schema 'APP'.
        at org.apache.derby.impl.jdbc.SQLExceptionFactory40.getSQLException(Unknown Source)
        at org.apache.derby.impl.jdbc.Util.generateCsSQLException(Unknown Source)
```

JDBC API

- Iako sam API sadrži dosta metoda i klasa one koje se najčešće koriste su:

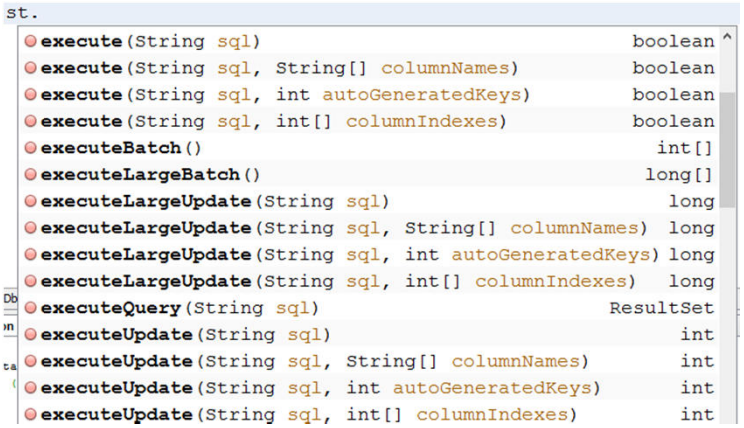
- **Connection** //Rukovanje konekcijom
- **Statement** //Rukovanje upitima
- **PreparedStatement** //Rukovanje pripremnljenim upitima
- **ResultSet** //Rukovanje rezultatima upita

- Jednom upoznat JDBC API, omogućava nam da radimo sa bilo kojim pozadinskim sistemom, uz male modifikacije koda

- Referenc lista JDBC API-ja (https://docs.oracle.com/javase/7/docs/api/java/sql/package-summary.html)

Unos podataka

- Za sve CRUD operacije može se koristiti interfejs **Statement**. Ova klasa/interfejs sadrži metode za izvršavanje različitih vrsta upita, i ona je srce JDBC API-ja.
- Instancu ovog interfejsa možemo dobaviti pomoću metode **createStatement** objekta klase Connection




```
Statement st = conn.createStatement();
try {
    st.execute("insert into hiscores (name,score) values ('player 1',25)");
    System.out.println("Score successfully inserted");
} catch(SQLException ex) { System.out.println("Insert failed"); }
```

Preuzimanje podataka

- Preuzimanje podataka iz baze se može izvršiti metodom **executeQuery** (takođe iz klase Statement)
- Ovaj metod vraća objekat klase (interfejsa) **ResultSet**.
- ResultSet se ponaša slično iteratoru. Nakon što dobavimo ResultSet objekat, marker pomeramo metodom **next**. Next vraća true ili false u zavisnosti od toga da li rezultat postoji ili ne postoji
- Result set nudi veliki broj metoda za dobavljanje kolona različitih tipova (**getString,getInt,getTimestamp...**)
- Određivanje kolone može se vršiti pomoću naziva kolone ili njene pozicije (pozicije idu od 1, a ne od 0 kao kod indeksiranja nizova)

```
ResultSet all_hiscores = st.executeQuery("select * from hiscores");
if(all_hiscores.next()){
    System.out.println("player name\tscore");
    System.out.println(
            all_hiscores.getString("name")+"\t"+
            all_hiscores.getString("score")
    );
}
```

Vežba

- Kreirati jednostavan program koji će omogućavati korisniku da unosi podatke u hiscores listu pomoću konzole
- Prilikom svakog unosa, prikazuje se lista svih igraca i njihovih poena

Preuzimanje više redova

- Moguće je preuzeti više redova pomoću metode **ResultSet**. Sve dok metod next vraća vrednost true:

```
//Get multiple rows
all_hiscores = st.executeQuery("select * from hiscores");
System.out.println("**********************");
System.out.println("player name\tscore");
while(all_hiscores.next()){
    System.out.println(
            all_hiscores.getString("name")+"\t"+
            all_hiscores.getInt(3)
    );
}
```

Brisanje i ažuriranje redova

- Iako je kod ažuriranja i brisanja moguće takođe upotrebiti Statement objekat, češće se koristi objekat tipa PreparedStatement
- Ovaj objekat ima mogućnost parametrizacije što obezbeđuje upit od eventualnog sql injection-a

```
//Update rows
PreparedStatement st_prep =
        conn.prepareStatement("update hiscores set score = 55 where id = ?");
st_prep.setInt(1, 2);
int updated_rows_count = st_prep.executeUpdate();
if(updated_rows_count>0){
    System.out.println(updated_rows_count + " row(s) successfully updated");
} else {
    System.out.println("No updates performed");
}
```

Vežba 1 (jcex142014 Check)

- Treba napraviti program: Kasa
- Kada korisnik startuje program, prikazuje se primitivni meni od tri stavke:
- 1. Novi racun, 2. listanje svih racuna 3. brisanje racuna
- Kada korisnik odabere opciju 1, program mu nudi da unese naziv artikla i cenu po kojoj je prodat, nakon čega se prodati artikal unosi u derby bazu podataka pod odgovarajućim datumom
- Kada korisnik odabere opciju 2, prikazuju se svi postojeći računi
- Kada korisnik odabere opciju 3, program mu nudi da unese id računa, nakon čega se račun briše i ponovo izlistavaju svi računi
- Poželjno je da program bar delimično bude objektno koncipiran

Rad sa MySql bazom podataka
(jcex142014 SimpleMysqlConnection)

- MySql je najpopularniji besplatni sistem (i open source) za upravljanje bazama podataka
- Koristi se u kombinaciji sa različitim tehnologijama, a vrlo često je u open source okruženjima
- Paleta korisnika mysql-a je impozantna

- http://www.mysql.com/customers/

NetBeans i MySql

- NetBeans ima ugrađen jednostavan sistem za komunikaciju sa MySql-om.
- Ukoliko ne želimo da koristimo neko drugo (više posvećeno) okruženje za rad sa MySql-om, možemo koristiti ovaj sistem