Elektrotehnički fakultet Univerziteta u Beogradu
Katedra za računarsku tehniku i informatiku

Rad sa bazama podataka

Infrastruktura za elektronsko poslovanje (SI3IEP)

Nemanja Kojić

2010/2011

Sadržaj časa

- Uvod
- Pregled postojećih okruženja za rad sa bazama podataka
  - ODBC
  - OLE DB
  - ADO .NET
  - JDBC
- Napredne tehnike i detalji

Uvod

- Danas postoje različite baze podataka (Sql Server, MySQL, Sybase IQ, Oracle, DB2 ...)
- Svaki proizvođač baze podataka nudi svoj API za komunikaciju sa bazom
- Aplikativni softver treba da podrži rad sa različitim bazama. Na koji način?
- Apstrahovanje pristupa bazi podataka (*database abstraction layer*)
- API koji unificira komunikaciju između baza podataka i aplikacija
- Postoji mnogo API-a (sa raznolikim interfejsima) u različitim programskim jezicima

ODBC
(Open Database Connectivity)

- Sloj koji apstrahuje pristup bazama podataka
- Nezavisan od programskog jezika, baze podataka i operativnog sistema
- Razvoj aplikacija na programskom jeziku C
- Pristup odgovarajućoj bazi podataka se ostvaruje instaliranjem odgovarajućeg ODBC drajvera
- Aplikacija sa bazom komunicira isključivo kroz interfejs ODBC-a
- Razvijen 1992, SQL Access Group.

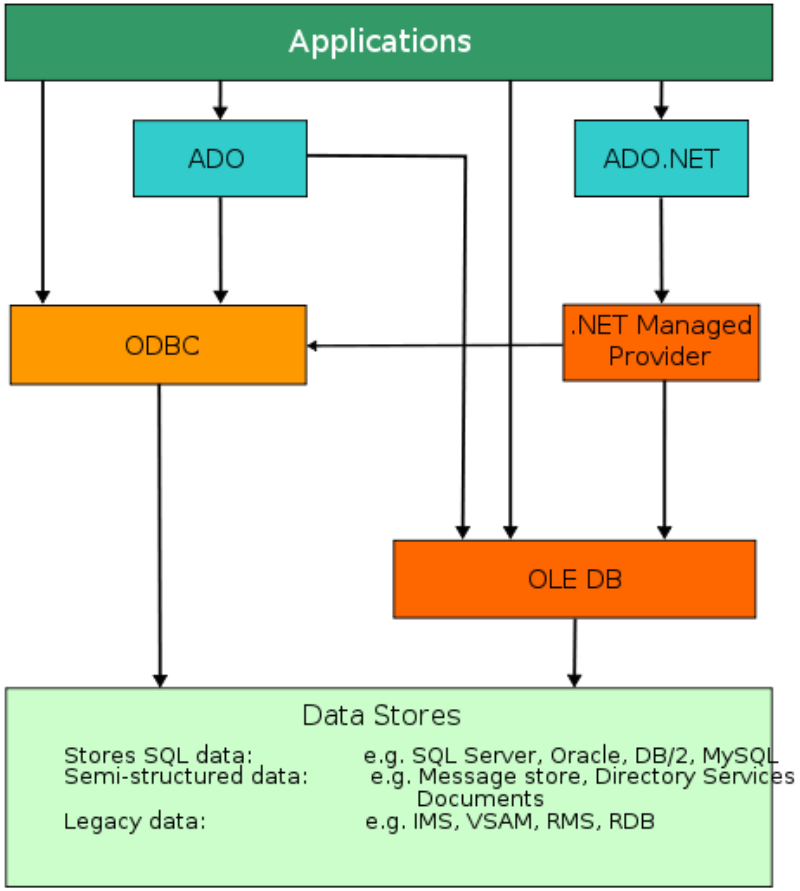
ODBC (2)
(Open Database Connectivity)

- ODBC nudi interfejs pristupa bazama podataka preko SQL upita
- ODBC "preuređuje" upite pre nego što ih izda na izvršavanje konkretnom drajveru baze
- Neke mane ODBC-a:
  - stabilnost drajvera
  - performanse drajvera

OLE DB
(Object Linking and Embedding Database)

- API dizanjniran od strane Majkrosoft-a
- Baziran na COM modelu
  - skup interfejsa implementiram korišćenjem COM tehnologije
- Zamena i naslednik ODBC-a
  - unapredio funkcionalnosti ODBC-a
  - proširio skup funkcionalnosti za potrebe rada sa objektnim bazama i izvorima podataka koji ne podržavaju SQL.
- Nudi sledeće koncepte:
  - DataSource
  - Session
  - Command
  - RowSet
- OLE DB je deo grupe Majkrosoft tehnologija koje se nazivaju Microsoft Data Access Components (MDAC)

Microsoft Data Access Components (MDAC)



**Note*: the Microsoft SQL Server Network Library (Net-Lib) is used specifically by SQL Server but is still counted as an official part of MDAC

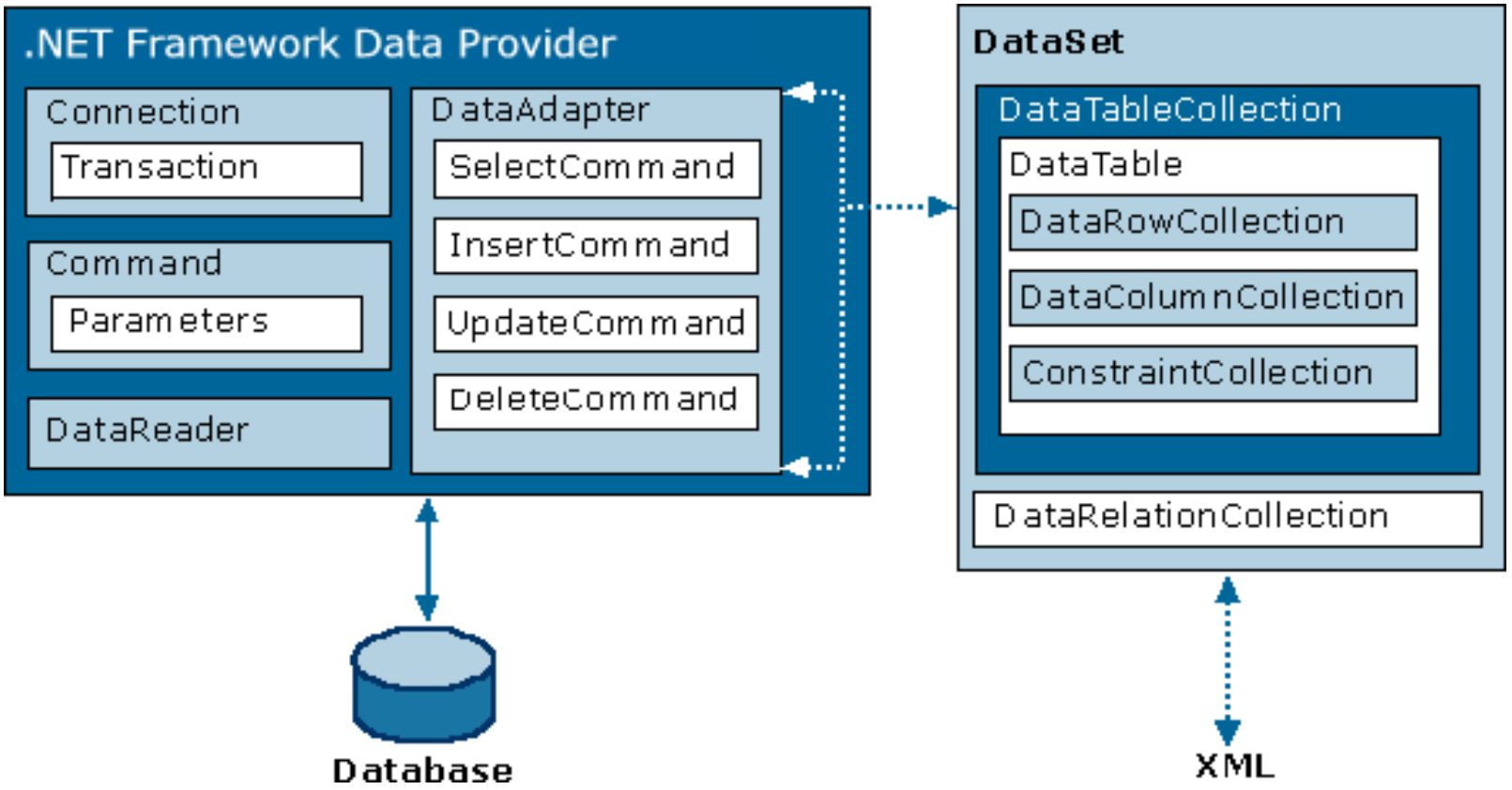
ADO (tehnologija)
ActiveX DataAccess Objects

- skup COM objekata za pristup izvorima podataka
- sloj iznad OLE DB
- čine ga kolekcije i objekti
- Izvori podataka:
  - baza podataka
  - tekstualni fajl
  - excel dokumenti
  - XML fajlovi
- ADO .NET
  - značajno unapredjenje date tehnologije
  - objektno orijentisan skup biblioteka koje omogućavaju interakciju za različitim izvorima podataka (*data sources*)
  - smatra se čak i potpuno novim proizvodom

ADO .NET arhitektura

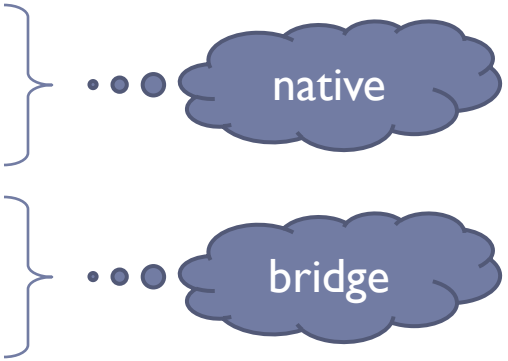
- pristup podacima se oslanja na dve komponente:
  - DataSet
  - DataProvider
- DataSet
  - lokalna kopija podataka iz baze podataka u memoriji
  - ne zahteva direktnu povezanost za bazom podataka
  - omogućava i perzistenciju podataka
- DataProvider
  - upravlja vezama ka bazi podataka (connections)
  - izvršava SQL upite
  - za bilo koji izvor podataka moguće je implementirati odgovarajući DataProvider

ADO .NET arhitektura

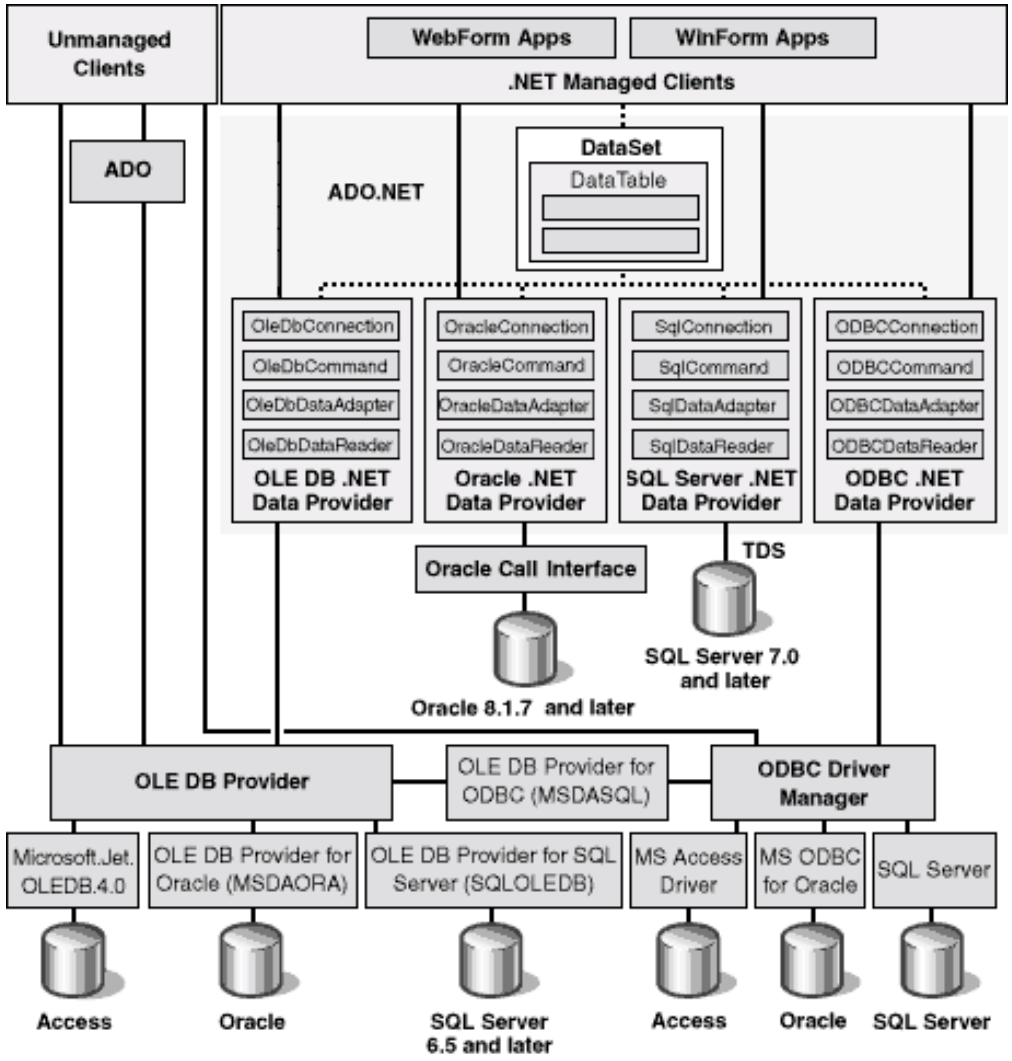


Provajderi

- Bridge provajderi
  - za pristup OLE DB i ODBC izvorištima podataka
  - omogućavaju korišćenje biblioteka dizajniranih za ranije tehnologije pristupa podacima
- Native provajderi
  - jedan nivo apstrakcije manje
  - donose poboljšanja u pogledu performansi
  - specijalno napisani za konkretnu bazu podataka
- SQL Server .NET Data Provider
- Oracle .NET Data Provider
- OLE DB .NET Data Provider
- ODBC .NET Data Provider



Arhitektura provajdera podataka



DataProvider

- Ključne komponente svakog od provajdera koje se koriste na aplikativnom nivou su:
  - Connection
    - objekat preko kojeg se dobijaju konekcije ka bazi
  - Command
    - izvršavanje SQL komandi,
      ExecuteReader, ExecuteNonQuery, ExecuteScalar
  - DataReader
    - rezultatima upita u obliku skupa zapisa,
      samo čitanje,
      jednosmerni pristup zapisima,
      zahteva postojanje aktivne veze sa bazom
  - DataAdapter
    - popunjavanje objekta DataSet,
      nije potrebna aktivna veza sa bazom podataka (dissconnected),
      omogućava manipulisanje podacima u okviru DataSet-a

Pristup bazi kroz ADO .NET

- Connection uspostavlja vezu aplikacije sa bazom podataka
- Command omogućava izvršavanja SQL upita nad bazom podataka
- Ako rezultat upita ima više od jedan zapis vraća se DataReader
- Alternativno, DataAdapter može biti korišćena za punjenje objekta DataSet
- Podaci u bazi se mogu ažurirati kroz objekat
  - Command ili
  - DataAdapter

ADO .NET primer

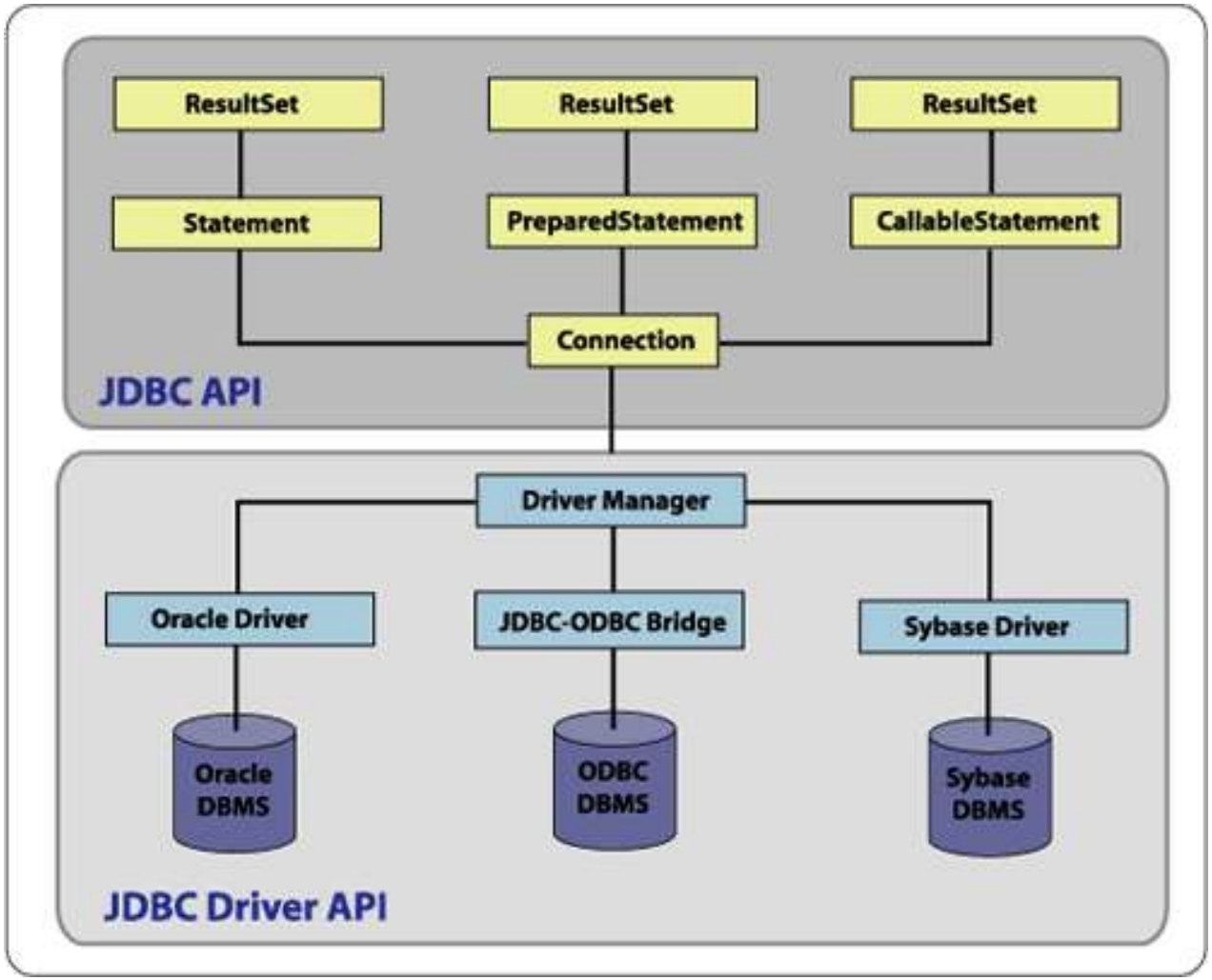
```
IDbConnection conn = new MySql.Data.MySqlClient.MySqlConnection("mysql
connection string");
conn.Open();
IDbTransaction trans = null;
try {
    trans = conn.BeginTransaction(IsolationLevel.Serializable);
    IDbCommand cmd = conn.CreateCommand();
    cmd.CommandText = "insert into person(name) values('John');";
    cmd.Transaction = trans;

    cmd.ExecuteNonQuery();
    trans.Commit();
    cmd.Dispose();
}
catch (Exception e) {
     Console.WriteLine(e.Message);
     if (trans != null) trans.Rollback();
}
finally {
    conn.Close();
}
```

JDBC API

- Java API za pristup bazama podataka
- Čini pristup različitim bazama podatak potpuno transparentim za programera i aplikaciju
- Prva verzija je izdata 1996, SUN
- Pristup bazi korišćenjem SQL-a
- Aplikacije i programi su:
  - nezavsni od platforme (Java)
  - nezavisni od baze podataka (JDBC API)

JDBC arhitektura



Osnovni koncepti JDBC programiranja

- URL za bazu podataka
  - jdbc:*ime protokola:drugi parametri (jdbc:postgresql:COREJAVA)*
- Učitavanje drajvera
  - Class.forName("org.postgresql.Driver")
  - Prosledjivanjem kao argumenta komandne linije (*java –jdbc.drivers=org.postgresql.Driver MyProg*)
- Veze i njihovo kreiranje (*java.sql.Connection*)
- SQL naredbe i njihvo izvršavanje (*java.sql.Statement*)
- Skupovi rezultata (*java.sql.ResultSet*)

Povezivanje sa bazom podataka

- DriverManager
  - stariji koncept
  - upravlja učitanim drajverima
    - zahtev za povezivanje prosleđuje odgovarajućem drajveru
  - nema recikliranja konekcija
    - loše performanse!!
  - ne postoji podrška za distribuiranje transakcije
- DataSource
  - noviji (superiorniji) koncept
  - poboljsava portabilnost programa
    - mogu se koristiti logička imena za izvor podatak umesto zadavanja informacija specifičnih za konkretni drajver
  - podrška za distribuirane transakcije
  - recikliranje konekcija

Upravljač drajverima i pravljenje veza (*DriverManager)*

```
String url = "jdbc:postgresql:COREJAVA";
String user = "john";
String pass = "smith";
Connection conn =
DriverManager.getConnection(url,user,pass);
```

- Pored "hard" kododovanih parametara u programskom kodu, mogu se koristiti i fajlovi svojstava (*properties* fajlovi)
- Linije u formatu *ime_parametra=vrednost*
- Primer properties fajla za prethodni primer:

```
jdbc.drivers=org.postgresql.Driver
jdbc.url=jdbc:postgresql:COREJAVA
jdbc.username=user
jdbc.password=pass
```

JDBC primer (1): Uspostavljanje veze sa bazom podataka

* Povezivanje na bazu pomoću klase DriverManager

```
// Inicijalizacija
Class.forName("com.mysql.jdbc.Driver")
String url = "jdbc:mysql://localhost/coffeebreak";

try {
   conn = DriverManager.getConnection(url, "username", "password");
   doSomeJob(conn);
   conn.commit();
   conn.close();
} catch (SQLException ex) {...}
```

* Povezivanje na bazu pomoću klase DataSource

```
DataSource ds = ...;
init(ds);
try {
   conn = ds.getConnection();
   doSomeJob(conn);
   conn.commit
   conn.close();
} catch (SQLException ex) {...}
```

JDBC primer (2): Čitanje iz baze podataka

* Čitanje podataka iz baze podataka

```
String query = "SELECT COF_NAME, PRICE FROM COFFEES";
try {
   Statement st = conn.createStatement();
   ResultSet rs = st.executeQuery(query);
   while (rs.next()) {
      String s = rs.getString("COF_NAME");
      float n = rs.getFloat("PRICE");
   }
}
catch (SQLException ex) {...}
```

JDBC primer (3)
Ažuriranje podataka u bazi

* Unos podataka u bazu podataka

```
try {
   Statement st = conn.createStatement();
   st.executeUpdate("INSERT INTO COFFEES VALUES('BLEND',200,7.99,0,0)");
   conn.commit();
} catch (SQLException ex) {...}
```

* Ažuriranje podataka u bazi podataka

```
try {
   Statement st = conn.createStatement();
   st.executeUpdate("UPDATE COFFEES SET PRICE=4.9 WHERE COF_NAME='BLEND'");
   conn.commit();
} catch (SQLException ex) {...}
```

* Brisanje podataka iz baze podataka

```
try {
   Statement st = conn.createStatement();
   st.executeUpdate("DELETE FROM COFFEES WHERE COF_NAME='BLEND'");
   conn.commit();
} catch (SQLException ex) { ... }
```

Napredne tehnike i detalji

Napredne tehnike i detalji (industrijska primena)

- Kodiranje nezavisno od provajdera podataka
  - Fabričke klase/metode
- Parametrizovanje upita
  - "SQL injection attack"!
- Korišćenje transakcija
- Obrada izuzetaka i grešaka
  - upravljanje resursima
- Upravljanje "skupim" resursima
  - recikliranje konekcija (connection pool)