



PERZISTENCIJA PODATKA U JAVI

**Ivan Senji
Mario Popović**

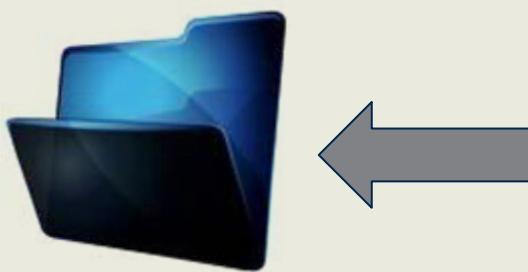
Rovinj, 13.-17. listopad 2009.

Sadržaj

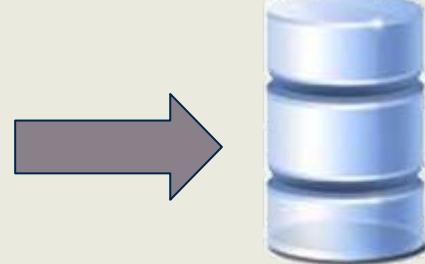
- ◆ Perzistencija podatka
- ◆ JDBC
- ◆ Objektno-relacijsko preslikavanje
- ◆ Alati za objektno-relacijsko preslikavanje
- ◆ JPA
- ◆ Primjer
- ◆ Pitanja

Perzistencija podataka

- ◆ Perzistencija – svojstvo podatka da nadživi izvođenje programa koji ga je stvorio



01101000 01110010
01101111 01110101
01100111 00110010
00110000 00110000
00111001 01101000
01110010 01101111
01110101 01100111
00110010 00110000
00110000 00111001



JDBC

- ◆ JDBC (*engl. Java Database Connectivity*) API je standard za povezivanje Java programskog jezika i baze podataka
- ◆ JDBC API omogućuje:
 - uspostavljanje veze sa bazom podataka
 - slanje SQL upita
 - procesiranje rezultata

JDBC

◆ Nedostaci JDBC:

- kod korištenja JDBC-a mogućnost greške je velika jer Java prevodilac ne može validirati upite koji se kreiraju spajanjem niza znakova
- potrebno je ručno transformirati podatke u objektni model koji se dalje koristi u programskoj logici

JDBC – primjer

```
package hroug2009model;

import java.util.ArrayList;
import java.util.List;

public class Korisnik {

    private Long id;
    private Ured ured;
    private String korisnickoIme;
    private String lozinka;
    private String prezime;
    private String ime;
    private String email;
    private DaNe aktivavan = DaNe.DA();
    private PodaciOUnosu podaciOUnosu = new PodaciOUnosu();
    private List<KorisnikRola> skupKorisnikRola =
        new ArrayList<KorisnikRola>();

}
```

JDBC – primjer

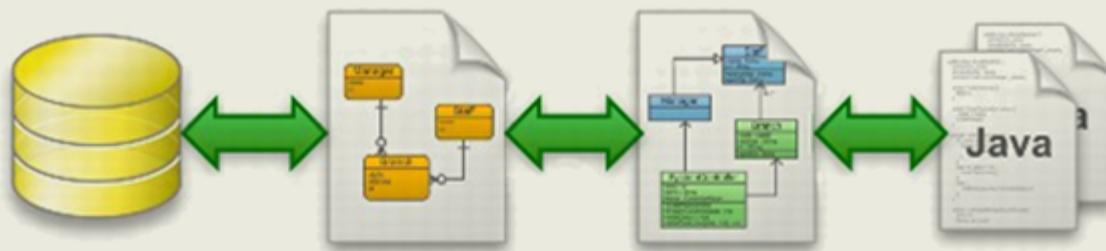
◆ Pozivanje upita:

```
public Korisnik findKorisnikById(Long id) {  
    Statement st = conn.createStatement();  
    ResultSet rs = st.executeQuery(  
        "select id, ime, prezime from korisnik where id = " + id);  
    while (rs.next()) {  
        Long id = rs.getLong(1);  
        String ime = rs.getString(2);  
        String prezime = rs.getString(3);  
        return new Korisnik(id, ime, prezime);  
    }  
}
```

Objektno-relacijsko preslikavanje

8

- ◆ ORM – tehnika za preslikavanje podataka između podataka u relacijskoj bazi podataka i objekata u objektno orijentiranim programskim jezicima



Objektno-relacijsko preslikavanje

◆ Prednosti ORM-a:

- smanjena količina koda kojeg je potrebno napisati čime se smanjuje mogućnost greške
- neovisnost aplikacije o izboru baze
- nije nužno detaljno poznavanje načina rada baze podataka

◆ Nedostaci ORM-a:

- neki ORM alati imaju slabije performanse prilikom rada s velikim količinama podataka u odnosu na ručno pisane SQL upite ili pohranjene procedure

Alati za objektno-relacijsko preslikavanje



Alati za objektno-relacijsko preslikavanje

11

iBATIS

- ◆ framework za perzistenciju podataka koji automatizira preslikavanje iz SQL baze podataka u *Plain Old Java Objects (POJO)*
- ◆ pravila za mapiranje se nalaze u zasebnim xml konfiguracijskim datotekama ili se zadaju korištenjem anotacija

Alati za objektno-relacijsko preslikavanje

12

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMap PUBLIC "-//iBATIS.com//DTD SQL Map 2.0//EN" "http://www.ibatis.com/dtd/sql-map-2.dtd">

<sqlMap namespace="Korisnik">

    <typeAlias alias="korisnik" type="Korisnik" />

    <resultMap id="result" class="korisnik">
        <result property="id" column="korisnikId" />
        <result property="korisnickoIme" column="korisnickoIme" />
        <result property="lozinka" column="lozinka" />
        <result property="ime" column="imeKorisnika" />
        <result property="prezime" column="prezimeKorisnika" />
        <result property="ured.id" column="uredId" />
    </resultMap>

    <sql id="select_korisnik">
        SELECT k.id AS korisnikId
        ,      k.korisnicko_ime AS korisnickoIme
        ,      k.lozinka AS lozinka
        ,      k.ime AS imeKorisnika
        ,      k.prezime AS prezimeKorisnika
        ,      k.ured_id AS uredId
        FROM   sif_korisnici k
    </sql>

    <select id="loadByPrimaryKey" resultMap="result" parameterClass="korisnik">
        <include refid="select_korisnik"/>
        WHERE k.id = #id#
    </select>
</sqlMap>
```

Alati za objektno-relacijsko preslikavanje

13

iBATIS

```
public Korisnik findKorisnikById(Long id) {  
    SqlSession session = sqlMapper.openSession();  
    try {  
        Korisnik korisnik = (Korisnik) session.select(  
            "Korisnik.loadByPrimaryKey", new Korisnik(id));  
    } finally {  
        session.close();  
    }  
}
```

Alati za objektno-relacijsko preslikavanje



- ◆ ORM framework za mapiranje objektnog modela u relacijsku bazu podataka
- ◆ rješava problem konceptualnih i tehničkih problema koji se javljaju između objektnog i relacijskog modela podatka (npr. ManyToMany veze)

Alati za objektno-relacijsko preslikavanje



◆ Primjena:

- dobro rješenje za aplikacije gdje se poslovna logika nalazi u srednjem sloju aplikacije
- nije dobro rješenje kada se poslovna logika nalazi u paketima, procedurama i triggerima spremljenim u bazi podataka

Alati za objektno-relacijsko preslikavanje



- ◆ HQL (eng. Hibernate query language)
 - sintaksa jezika slična SQL-u
 - potpuno objektno orientiran, podržava relaciju nasljeđivanje i agregacije

Alati za objektno-relacijsko preslikavanje

```
<hibernate-mapping>
    <class name="Korisnik" table="SIF_KORISNICI">
        <id name="id" type="long">
            <column name="ID" precision="15" scale="0" />
            <generator class="sequence">
                <param name="sequence">KRK_SEQ</param>
            </generator>
        </id>
        <many-to-one name="ured" class="Ured" fetch="select">
            <column name="URED_ID" precision="15" scale="0" />
        </many-to-one>
        <property name="korisnickoIme" type="string">
            <column name="KORISNICKO_IME" length="100" not-null="true" />
        </property>
        <property name="lozinka" type="string">
            <column name="LOZINKA" length="100" not-null="true" />
        </property>
        <property name="ime" type="string">
            <column name="IME" length="256" not-null="true" />
        </property>
        <property name="prezime" type="string">
            <column name="PREZIME" length="256" not-null="true" />
        </property>
        <bag name="skupKorisnikRola" inverse="true"
            cascade="save-update,delete,delete-orphan">
            <key><column name="KORISNIK_ID" precision="15" scale="0"
                not-null="true" /></key>
            <one-to-many class="KorisnikRola" />
        </bag>
    </class>
</hibernate-mapping>
```

Alati za objektno-relacijsko preslikavanje

18



```
List<Korisnik> sviKorisnici =  
        em.createQuery("from Korisnik").getResultList();  
  
List<Korisnik> korisniciUreda =  
        em.createQuery("from Korisnik where ured = :ured")  
        .setParameter("ured", trazeniUred)  
        .getResultList();
```

JPA

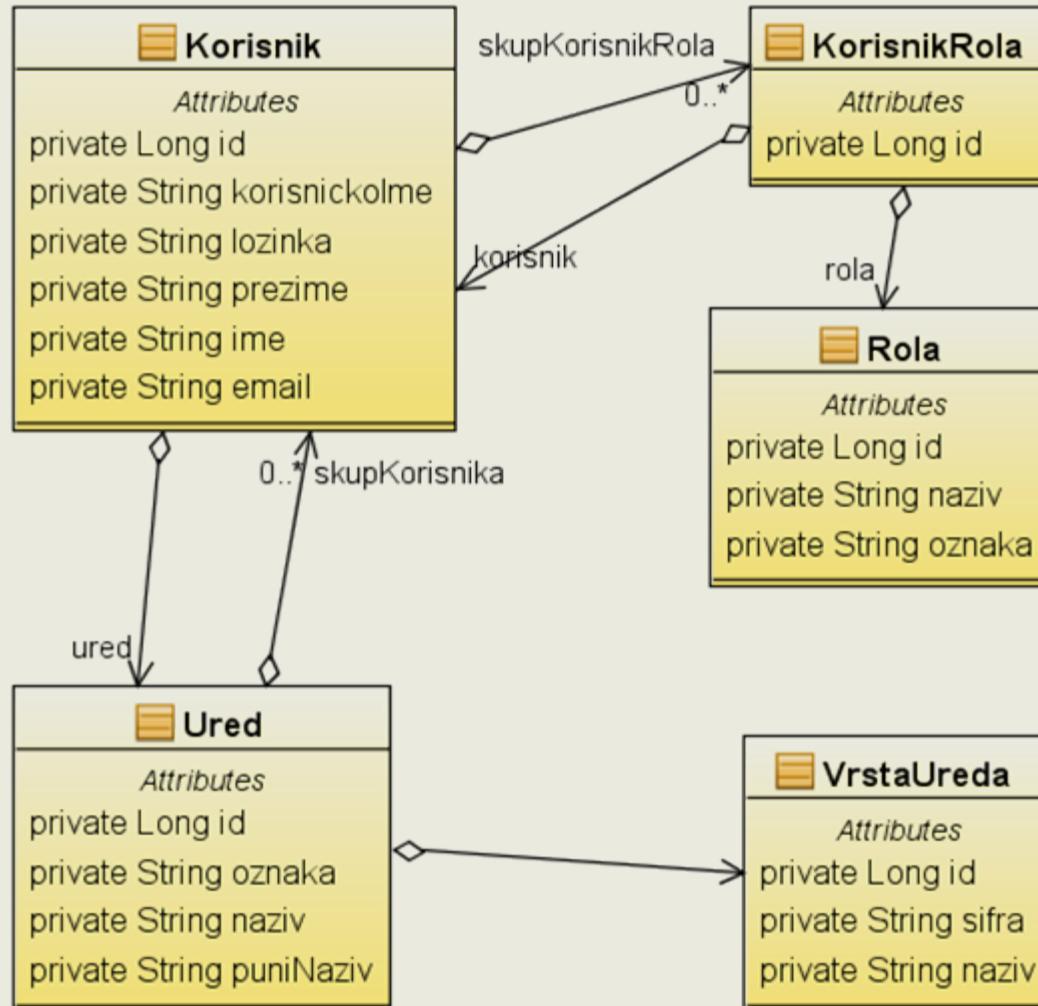
- ◆ JPA – Java Persistence API
- ◆ standardiziran kroz Java Community Process i dio je EJB 3.0 specifikacije
- ◆ nije ograničen samo na Java EE, već radi i sa Java SE

JPA

- ◆ nastao na temelju najboljih ideja iz ORM frameworka (Hibernate, Oracle TopLink i dr.)
- ◆ definira JPQL (eng. Java Persistence Query Language) jezik za kreiranje upita nad objektima

Primjer

21



Primjer

22

The screenshot shows the NetBeans IDE interface with the following details:

- Title Bar:** hroug2009 (jar) - NetBeans IDE 6.7 RC3
- Projects Tab:** Shows the project structure with Source Packages, Test Packages, Scala Packages, Libraries, and others.
- Files Tab:** The current file is Korisnik.java, which is an entity class.
- Code Editor:** Displays the Java code for the Korisnik class, including annotations for persistence and relationships.
- Navigator Tab:** Shows the members of the Korisnik class, including getters and setters for ID, name, login, email, and roles.
- Output Tab:** Shows the output of the build process.

```
16 import javax.persistence.SequenceGenerator;
17 import javax.persistence.Table;
18
19 @Entity
20 @Table(name = "SIF_KORISNICI")
21 public class Korisnik implements Serializable {
22
23     @Id
24     @Column(name = "ID", nullable = false, precision = 15, scale = 0)
25     @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "KKE_SEQ")
26     @SequenceGenerator(name = "KKE_SEQ", sequenceName = "KKE_SEQ", allocationSize = 1)
27     private Long id;
28
29     @Column(name = "KORISNIČKO_IME", unique = true, nullable = false, length = 32)
30     private String korisnickoIme;
31
32     @Column(name = "LOGINKA", nullable = false, length = 64)
33     private String loginka;
34
35     @Column(name = "PREZIME", nullable = false)
36     private String prezime;
37
38     @Column(name = "IME", nullable = false)
39     private String ime;
40
41     @Column(name = "E_MAIL", nullable = true, length = 320)
42     private String email;
43
44     @ManyToOne(fetch = FetchType.LAZY)
45     @JoinColumn(name = "URED_ID", unique = false, nullable = true)
46     private Ured ured;
47
48     @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.LAZY, mappedBy = "korisnik")
49     private List<KorisnikRole> skupKorisnikRole = new ArrayList<KorisnikRole>();
50
51
52
53     @Override
54     public String toString() {
55         return String.format("Korisnik[id=%s, ime=%s, prezime=%s]", id, ime, prezime);
56     }
57
58     public String getEmail() {
59         return email;
60     }
61 }
```

PRIMJER

PITANJA & ODGOVORI

Kontakt

24

IVAN SENJI

ivan.senji@in2.hr

MARIO POPOVIĆ

mario.popovic@in2.hr