

JPA

Ukratko o označavanju entiteta

Entitet?

- Entitet je razred čiji se objekti pohranjuju u bazu podataka i koji ima vlastiti identitet (ima primarni ključ po kojem je dohvativ)
- Pri anotiranju razreda "obična" svojstva te veze prema drugim entitetima označavaju se različitim anotacijama

Veze prema više entiteta

- Ukoliko između jedne vrste entiteta i druge vrste entiteta postoji veza tipa *xToMany* (npr. Korisnik ima više Posudba), JPA dozvoljava da se ta veza u kodu iskaže uporabom raznih sučelja (npr. Set, List, Map)
- Međutim, ne poštuje se semantika tih sučelja (primjerice, List ne garantira poredak!)

Primarni ključ entiteta

- Svaki entitet mora imati primarni ključ prema kojem se primjerci tog entiteta razlikuju
- Ključ može biti jednostavan (jedan Long, String, ...) ili složen (kompozitan)
- Preporuča se uporaba jednostavnih ključeva
- Mi ovdje nećemo govoriti o kompozitnim ključevima i načinima njihovog definiranja (opisano je u specifikaciji)

Označavanje entiteta

@Entity

@Table(name="customer_records")

public class CustomerRecord {

@Id @GeneratedValue

private Long id;

@Column(nullable=false, length=30)

private String firstName;

@Column(nullable=false, length=50)

private String lastName;

@Column(nullable=true, length=20, unique=true)

private String nickName;

@Temporal(TemporalType.TIMESTAMP)

private Date lastVisit;

}

Dvosmjerna 1-na-1

```
@Entity
public class Employee {
    private Cubicle assignedCubicle;
```

Vlasnik relacije
Ima strani ključ na Cubicle

```
    @OneToOne
    public Cubicle getAssignedCubicle() {
        return assignedCubicle;
    }
    public void setAssignedCubicle(Cubicle cubicle) {
        this.assignedCubicle = cubicle;
    }
    ...
}
```

```
@Entity
public class Cubicle {
    private Employee residentEmployee;
```

```
    @OneToOne(mappedBy="assignedCubicle")
    public Employee getResidentEmployee() {
        return residentEmployee;
    }
    public void setResidentEmployee(Employee employee) {
        this.residentEmployee = employee;
    }
    ...
}
```

Inverzni kraj

Dvosmjerna n-na-1 / 1-na-n

```
@Entity
public class Employee {
    private Department department;
```

Vlasnik relacije
Ima strani ključ na Department

```
    @ManyToMany
    public Department getDepartment() {
        return department;
    }
    public void setDepartment(Department department) {
        this.department = department;
    }
    ...
}
```

```
@Entity
public class Department {
    private Collection<Employee> employees = new HashSet();

    @OneToMany(mappedBy="department")
    public Collection<Employee> getEmployees() {
        return employees;
    }

    public void setEmployees(Collection<Employee> employees) {
        this.employees = employees;
    }
    ...
}
```

Inverzni kraj



Jednosmjerna 1-na-1

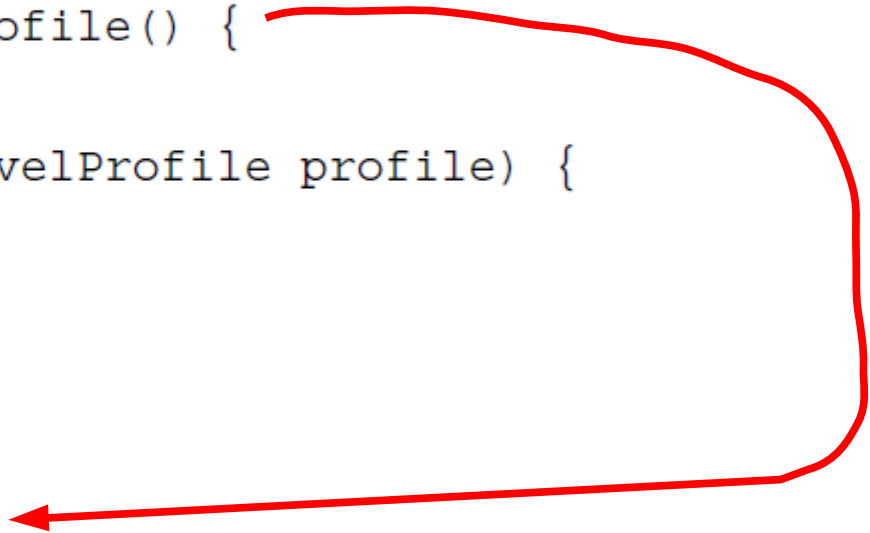
```
@Entity
public class Employee {
    private TravelProfile profile;
```

Vlasnik relacije

Ima strani ključ na TravelProfile

```
    @OneToOne
    public TravelProfile getProfile() {
        return profile;
    }
    public void setProfile(TravelProfile profile) {
        this.profile = profile;
    }
    ...
}
```

```
@Entity
public class TravelProfile {
    ...
}
```



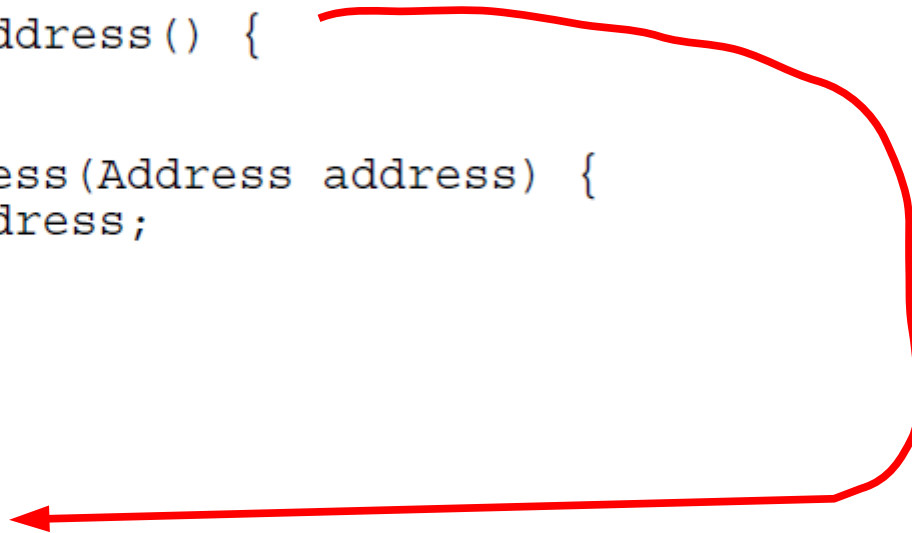
Jednosmjerna n-na-1

```
@Entity
public class Employee {
    private Address address;
```

Vlasnik relacije
Ima strani ključ na Address

```
    @ManyToMany
    public Address getAddress() {
        return address;
    }
    public void setAddress(Address address) {
        this.address = address;
    }
    ...
}
```

```
@Entity
public class Address {
    ...
}
```



Dvosmjerna n-na-n

```
@Entity
public class Project {
    private Collection<Employee> employees; Vlasnik relacije
```

```
    @ManyToMany
    public Collection<Employee> getEmployees() {
        return employees;
    }

    public void setEmployees(Collection<Employee> employees) {
        this.employees = employees;
    }
    ...
}
```

```
@Entity
public class Employee {
    private Collection<Project> projects;
```

```
    @ManyToMany(mappedBy="employees")
    public Collection<Project> getProjects() {
        return projects;
    }
```

```
    public void setProjects(Collection<Project> projects) {
        this.projects = projects;
    }
    ...
}
```

**Par stranih
ključeva ide u
zasebnu
(spojnu) tablicu**

**Inverzni
kraj**

Ograničenja na relaciju

```
@Entity
@Table(name="customer_records")
public class CustomerRecord {

    ...

    @ManyToOne
    @JoinColumn(nullable=false, unique=false)
    private Product favoriteProduct;

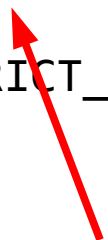
    ...

}
```

- x “Obična” svojstva (engl. *Property*): @Column
- x Relacija xToOne: @JoinColumn

Ograničenja na razini entiteta

```
@NamedQueries({
    @NamedQuery(name="Group.findAllSemUsers",query="select distinct ug.user from Group as g,
    IN(g.users) ug where g.compositeCourseID LIKE :compositeCourseID AND relativePath LIKE '0/%'"),
    @NamedQuery(name="Group.findForUser",query="select g from Group as g, IN(g.users) ug where
    g.compositeCourseID LIKE :compositeCourseID AND relativePath LIKE '0/%' AND ug.user=:user")
})
@Entity
@Table(name="groups",uniqueConstraints={
    // Ne mogu postojati dvije grupe s istim compositeCourseID i relativePath
    @UniqueConstraint(columnNames={"compositeCourseID","relativePath"}),
    // Roditelj ne može imati dva djeteta koja se zovu isto
    @UniqueConstraint(columnNames={"parent_id","name"})
})
@Cache(usage=CacheConcurrencyStrategy.NONSTRICT_READ_WRITE)
public class Group implements Serializable {
    ...
}
```



Od svojstva "User parent", gdje je primarni ključ od razreda User nazvan "id" (hibernate sam generira ovakav naziv – trebamo ga stoga koristiti)

Gdje pisati anotacije

- Anotacije mogu ići:
 - Nad definicijom članskih varijabli
 - Nad getterima
- Jednom kada ste odlučili, NE smijete ih miješati: u razredu sve mora biti anotirano na isti način

Više...

- Pogledati specifikaciju JPA, posebice sekcije 2.9 i 2.10