ORM. Hibernate

Duomenų paėmimas iš DB

Norėdami paimti duomenis turėtumėme susikurti sesiją ir paimti įrašą:

```
Session session=sessionFactory.openSession();


Uzduotis uzduotis= (Uzduotis) session.get(Uzduotis.class, 1);
System.out.println(uzduotis.getName());


session.close();
```

Duomenų paėmimas iš DB

Tačiau mūsų klasė „Uzduotis“ neturi nurodymų kur yra duomenys ir kaip jie surišti, todėl modifikuokime klasę:

```
@Entity
@Table(name="tasks")
public class Uzduotis {
  @Id
  public int id;
  ....
}
```

Dabar mūsų sukurta programa turėtų paimti duomenis iš DB.

Užduotis

Sukurkime Singleton klasę SessionFactory, kuri prisijungs prie Hibernate ir kurios pagalba galėsime gauti Hibernate sesijas.

Duomenų įterpimas

Mūsų id laukui taip pat nurodykime jog jis yra automatiškai generuojamas:

```
@GeneratedValue(strategy=GenerationType.IDENTITY)
```

Tuomet įterpti naują įrašą galėsime:

```
Session session=sessionFactory.openSession();
session.beginTransaction();

Uzduotis uzduotis=new Uzduotis("Pasimokinti SQL",  new Date());
session.save(uzduotis);

session.getTransaction().commit();
session.close();
```

Užduotis

Sukurkime DAO metodus (dirbančius su Hibernate) duomenų įterpimui, atnaujinimui ir ištrynimui.

Visų įrašų atspausdinimas

```
Session session=sessionFactory.openSession();
session.beginTransaction();

List<Uzduotis> uzduotys= session.createCriteria(Uzduotis.class).list();
// arba
// List<Uzduotis> uzduotys= session.createQuery("from Uzduotis").list();
for (Uzduotis uzduotis:  uzduotys) {
    System.out.println(uzduotis.getName()+" "+uzduotis.getDate());
}



session.getTransaction().commit();
session.close();
```

Įrašų trynimas

```
Session session=sessionFactory.openSession();
session.beginTransaction();

Uzduotis uzduotis= (Uzduotis) session.get(Uzduotis.class, 1);
session.delete(uzduotis);

session.getTransaction().commit();
session.close();
```

Maping anotacijos

Kiekvieno lauko pavadinimas turi sutapti su DB įrašo pavadinimu, arba turi būti aprašytas:

```
@Column(name="TEXT")
```

Norint nurodyti pagal nutylėjimą lauko reikšmę:

```
@ColumnDefault("'N/A'")
```

Norint nurodyti jog laukas negali būti null:

```
@Column(nullable=false)
```

Hibernate automatinis lentelių kūrimas

Hibernate taip pat gali automatiškai sukurti ir DB lenteles:

Tereikia nurodyti hibernate.cfg.xml faile parametrą:

<property name="hbm2ddl.auto">update</property>

Užduotis

Sukurkite klasę:

Status kurią sudarytų šie laukai

id - auto number

name - varchar

Sukurkime JPA mapping anotaciją, nurodykime jog Hibernate ją mappint‘ų iš DB ir migruokite duomenis į MySQL

Sukurkite DAO ir pamėginkime paimti ir atvaizduoti įrašus

Named querys - HQL

Vardinės užklausos gali būti parašytos arba SQL kalba arba HQL kalba. Rašant užklausas HQL kalba dirbama su objektais, SQL su duomenų baze

```
@NamedQueries({
   @NamedQuery(
      name = "taskById",
      query = "from Task t where t.id = :id"
   )
})
```

Native name querys

Naudojant SQL kalbą, mes turėtumėme nurodyti kokio tipo objekto norėsime gauti.

```
@NamedNativeQueries({
  @NamedNativeQuery(
    name = "taskByIdNative",
    query = "select * from tasks t where t.id = :id",
    resultClass = Task.class
  )
})
```

Užduotis

Sukurkime named query kuri paimtų visus darbus kurie yra dar neatlikti (jų status – 1)

Sukurkime kitą named qury kuri gražintų visus darbus kurių name laukelyje yra tam tikras žodis