ORM. Hibernate

Užduotis

Susikurkime naują Maven projektą (Quickstart)

Nepamirškime nurodyti Maven Java versiją:

<maven.compiler.source>1.10</maven.compiler.source>

<maven.compiler.target>1.10</maven.compiler.target>

Instaliuokime paketą MySQL Connector/J

Užduotis

Sukurkime duomenų bazę kurią sudarys tokios lentelės ir laukai:

Tasks

id

name

task

status_id

Statuses

id

Name

CRUD interfeisai

Dažniausiai visiems objektams (ir releacinės lentelės įrašams) mes turime panašius veiksmus:

C – create

R – read

U – update

D – delete

Tai dažniausia vadinama CRUD

Tiesioginio kreipimosi į DB trūkumai

Trūkumai

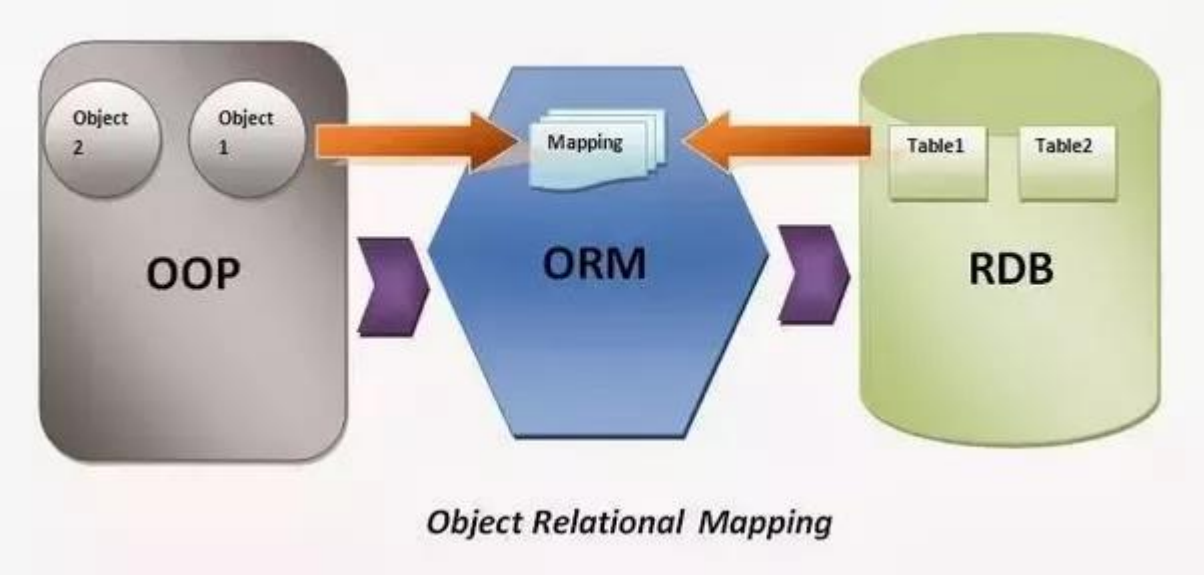
1. Papildomo kodo rašymas (SQL ir programinio kodo)
2. Kodo pasikartojimas
3. Pasikeitus DB gali tekti perrašyti SQL užklausas (DAO failą)
4. Rankinis transakcijų ir įrašymų (atnaujinimų) valdymas

Privalumai

1. Galima parašyti optimalesnes užklausas
2. Galima parašyti sudėtingesnes ir sudėtines užklausas
3. Galima naudotis DB strored procedūromis, įvykiais ir kt.

ORM – object-relational mapping

Tai programavimo technika (bibliotekos, karkasai) skirti konvertuoti duomenis iš objektiškai parašytos vietos į releacinių duomenų bazių įrašus ir atvirkščiai.



ORM

Paverčiant objektus į releacinės DB įrašus atsiranda daug nesuderinamumų, pagrindiniai išskiriami:

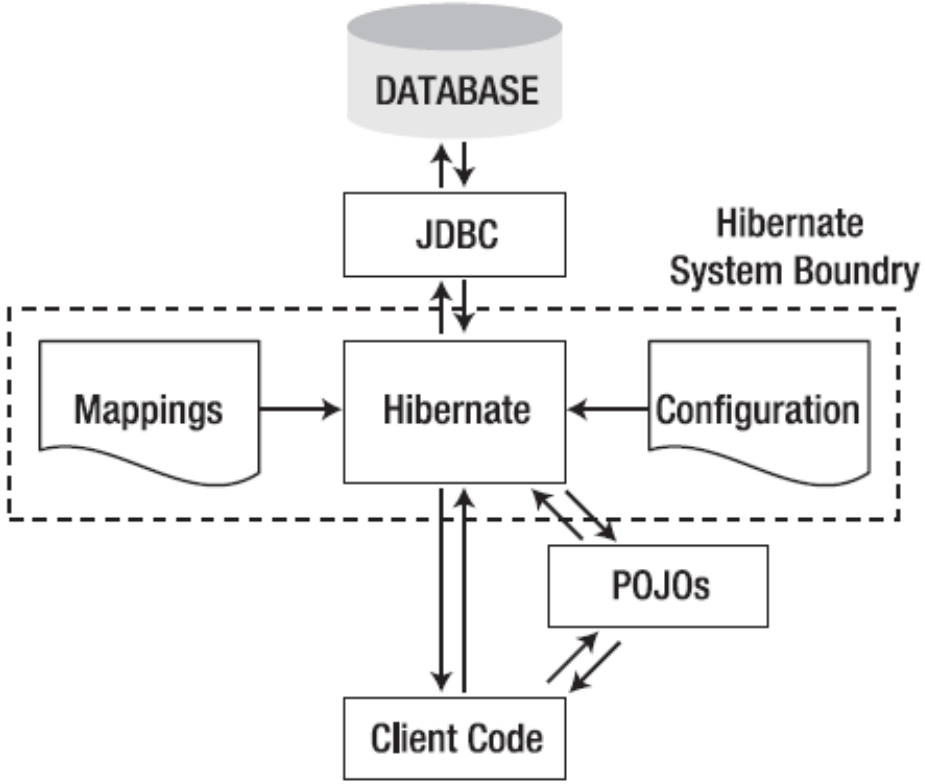
Granuliarumas – dažniausiai klasės yra žymiai mažesnio granuliarumo nei objektai

Paveldimumas – DB neturi paveldimumo

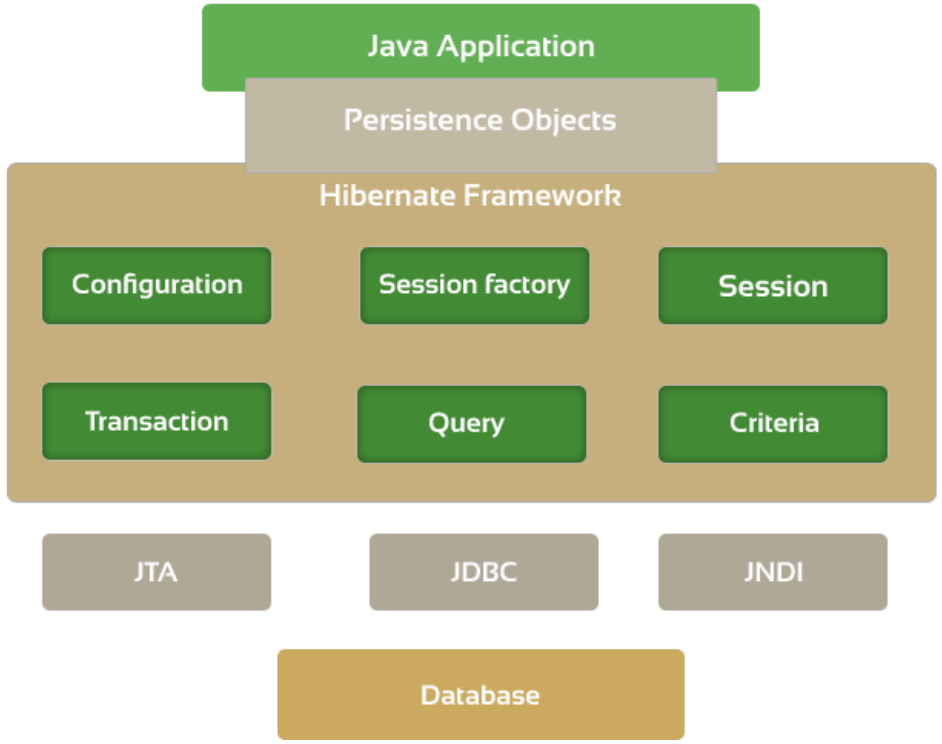
Identitetas – releacinėje DB įrašus apibrėžia pirminis raktas, objektiniame programavime įrašai gali būti tie patys arba identiški

Sąryšių – objektiniame programavime sąryšius atspindi adresų priskyrimas, tuo tarpu releacinėse duomenų bazėse tai aprašoma išoriniais ryšiais

Hibernate karkaso veikimo principai



Hibernate karkaso veikimo principai



Paruoškime projektą darbui

Psinaudodami Maven instaliuokime paketus:

- MySQL Connector/J (jį jau buvome instaliavę)
- Hibernate ORM Hibernate Core

XML paketų palaikymas

Jei JAVA versija > 1.8 tuomet mums papildomai reikės paketų XML failų skaitymui:

```
<dependency>
    <groupId>javax.xml.bind</groupId>
    <artifactId>jaxb-api</artifactId>
    <version>2.2.11</version>
</dependency>
<dependency>
    <groupId>com.sun.xml.bind</groupId>
    <artifactId>jaxb-core</artifactId>
    <version>2.2.11</version>
</dependency>
<dependency>
    <groupId>com.sun.xml.bind</groupId>
    <artifactId>jaxb-impl</artifactId>
    <version>2.2.11</version>
</dependency>
<dependency>
    <groupId>javax.activation</groupId>
    <artifactId>activation</artifactId>
    <version>1.1.1</version>
</dependency>
```

Hibernate konfigūracija

Sukurkime XML failą kuriame bus visa hibernate konfigūracija:

hibernate.cfg.xml (jis turi būti šakniniame kodo kataloge)

hibernate.cfg.xml

```
<hibernate-configuration>
  <session-factory>

   <!-- Database connection settings -->
   <property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
   <property name="connection.url">jdbc:mysql://localhost:3306/uzduotys?useSSL=false</property>
   <property name="connection.username">gediminas</property>
   <property name="connection.password">labas</property>

   <property name="dialect">org.hibernate.dialect.MySQL5InnoDBDialect</property>
   <property name="show_sql">true</property>

   <!-- Use Annotation-based mapping metadata -->
   <mapping class="com.poligonas.TasksList.Entities.Uzduotis"/>

  </session-factory>
</hibernate-configuration>
```

Session‘s factory

Norėdami dirbti su Hibernate, mes turėtumėme susikurti Sesijų fabriką:

```
StandardServiceRegistry serviceRegistry = new
                StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();
Metadata  metadata = new MetadataSources(serviceRegistry).getMetadataBuilder().build();
SessionFactory sessionFactory = metadata.getSessionFactoryBuilder().build();

//Baigus darbą
serviceRegistry.close();
```

Visa tai turėtų būti patalpina į Singletono klasę ir egzistuoti tik vienas egzempliorius sessionFactory objekto visoje sistemoje.

Užduotis

Pamėginkime paleisti Hibernate iš main() metodo ir pažiūrėkime ar visos konfigūracijos yra geros.

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.