Hibernate

Hibernate

- Är ett ramverk för att hantera persistent data i en objektorienterad värld.
- Hanterar hur man sparar, skapar och letar efter objekt.
- hibernate.org
- Framtaget av Jboss corperation, jboss är en opensource applikationsserver som ägs av Redhat.

- Ni ska ladda ner hibernate-core
- hibernate annotation är bra att ha
- hibernate EntityManager är knepigare att sätta upp och är endast för den som vill fördjupa sig mer.

- Du kan använda sql frågor direkt med objekt retur eller det speciella HQL (hibernate query language
-

HQL

- Exempel
  - select from Student
  - select from Student as student inner join student.courses as course where course.name = ”blaha”

Hibernate - core

- Vi startar med att endast använda core paketet.
-

Student.java

```
package entity;

public class Student {

    private long id;

    private String name;

    public Student() {

    }

    public long getId() {

        return id;

    }
```

```
public void setId(long id) {
    this.id = id;
}

public void setName(String name) {
    this.name = name;
}
```

Student-hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-mapping PUBLIC

        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"

        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

     <class name="bo.Student" table="T_STUDENT">

        <id name="id" column="ID">

            <generator class="native"/>

        </id>

        <property name="name"/>

    </class>

</hibernate-mapping>
```

- <set name="courses" table=”T_STUDENT_COURSES">
- <key column="STUDENT_ID"/>
- <many-to-many column="COURSE_ID" class="bo.Course"/>
- </set>

hibernate.properties

- Ska innehålla detta och läggas i top katalogen i projektet.
- original filen finns i $HIBERNATE/etc

hibernate.dialect org.hibernate.dialect.MySQLDialect

hibernate.dialect org.hibernate.dialect.MySQLInnoDBDialect

hibernate.dialect org.hibernate.dialect.MySQLMyISAMDialect

hibernate.connection.driver_class com.mysql.jdbc.Driver

hibernate.connection.url jdbc:mysql:reine

hibernate.connection.username reine

hibernate.connection.password tjo

hibernate.cfg.xml

- Ska även den ligga i top katalogen i projektet

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.url">jdbc:mysql://localhost/reine</property>
        <property name="connection.username">reine</property>
        <property name="connection.password">tjo</property>
```

```
        <property name="connection.pool_size">2</property>

        <property name="dialect"> org.hibernate.dialect.MySQLDialect</property>

    <property
      name="current_session_context_class">org.hibernate.context.ManagedSessionContext</prope
      rty>

  <property name="cache.provider_class">org.hibernate.cache.NoCacheProvider</property>

        <property name="show_sql">true</property>

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

        <mapping resource="entity/Student-hbm.xml"/>

    </session-factory>

</hibernate-configuration>
```

Använda POJOn

- Plain Old Java Objekt
- Sedan sk aman använda den som ett persistent objekt.

```
package utilk;

import org.hibernate.*;

import org.hibernate.cfg.*;

public class HibernateUtil {


    private static SessionFactory sessionFactory = null;

    static {

        try {

            // Create the SessionFactory from hibernate.cfg.xml

            sessionFactory = new Configuration().configure().buildSessionFactory();
```

```
    } catch (Throwable ex) {
            // Make sure you log the exception, as it might be swallowed
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }
    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }

}
```

Skapa objekt

```
import ht.HibernateUtil;

import org.hibernate.Session;

import org.hibernate.*;

import org.hibernate.cfg.*;

-------

    Session session = (new Configuration().configure().buildSessionFactory()).openSession();

    session.beginTransaction();

    Test test = new Test();

    test.setName(name);

    session.save(test);

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

Leta efter objekt

```
session.beginTransaction();

List result = session.createQuery("from
Student").list();

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

Men annotation ändras bönan till

- Skippa cfg filen för varje böna och gör detta....

```
package bo;

import java.io.Serializable;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;
```

```
@Entity
@Table(name="T_TEST")
public class NewEntity implements Serializable {
    private static final long serialVersionUID = 1L;
    private Long id;
    public void setId(Long id) {
        this.id = id;
    }
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public Long getId() {
        return id;
```

```
@Override
 public int hashCode() {
     int hash = 0;
     hash += (id != null ? id.hashCode() : 0);
     return hash;
 }

 @Override
 public boolean equals(Object object) {
     if (!(object instanceof NewEntity)) {
```

```
        return false;
    }
    NewEntity other = (NewEntity) object;
    if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {
        return false;
    }
    return true;
}
@Override
public String toString() {
    return "ent.NewEntity[id=" + id + "]";
}
```