Datenmanagement - Datenbankanwendungen mit JDBC

Seminar 3, am 14.12.2017

Hibernate installieren (schon erledigt) und Projekt strukturieren

- Hibernate und JDBC Dependencies sind heruntergeladen und im Build-Path eures Projektes
- Euer Modell ist entweder in einem Package in diesem Projekt oder in einem separaten Projekt
- Ihr habt bereits eine (eventuell nicht vollständige) Persistenzschicht mit JDBC implementiert, die hoffentlich auch in einem separaten Package liegt!

Hibernate konfigurieren

- hibernate.cfg.xmlwird im src-Ordner erstellt und beinhaltet Informationen über
    - Datenbank
    - Benutzername
    - Passwort
    - Treiber
    - SQL-Dialekte
    - Caching
- log4j.properties wird im srcOrdner erstellt und konfiguriert den Logger
- Xml-Mapping Dateien werden üblicherweise im gleichen Ordner wie die Klassen angelegt
- Annotations-Mapping wird direkt in der Klasse angewendet **Wir verwenden Annotations!** Ihr müsst also eure Klassen wie in den LV-Unterlagen beschrieben annotieren - arbeitet euch dabei schrittweise vor, ihr müsst nicht alle Domainklassen sofort machen!

src/hibernate.cfg.xml

```
<!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.autocommit">true</property>
      <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
      <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
      <property name="connection.username">fhv</property>
      <property name="connection.password">fhv</property>
      <property name="connection.url">jdbc:mysql://data.infeo.at/fhv</property>

       <mapping class="domain.Person" />

     <!-- <mapping resource="domain/Person.hbm.xml"/>-->
   </session-factory>
</hibernate-configuration>
```

src/log4j.properties

```
log4j.rootLogger=DEBUG, A1
log4j.appender.A1=org.apache.log4j.ConsoleAppender
log4j.appender.A1.layout=org.apache.log4j.PatternLayout
log4j.appender.A1.layout.ConversionPattern=\
 %-4r [%t] %-5p %c %x - %m%n
```

Hibernate verwenden

Main.java

```
Configuration configuration = new Configuration();
configuration.configure("hibernate.cfg.xml");
StandardServiceRegistryserviceRegistry = new
                                        StandardServiceRegistryBuilder().applySettings(
                                        configuration.getProperties()).build();

factory= configuration.buildSessionFactory(serviceRegistry);
Session session = null;
Transaction tx= null;
try{
       session= sessionFactory.openSession();
       tx= session.beginTransaction();
       session.Save(myObject);
       tx.commit();
} catch(Exception ex) {
       if(tx != null) tx.Rollback();
} finally{
       if(session!= null) session.close();
}
```

Datenbankfassade mit Hibernate implementieren

Wenn die obige Konfiguration geklappt hat und ihr es geschafft habt mit Hibernate Objekte zu materialisieren und dematerialisieren, ist der nächste Schritt wieder eine entsprechende Datenbankfassade zu bauen (siehe vorherige LVs und Seminare) und die Persistenzfunktionalitäten dort zu kapseln.