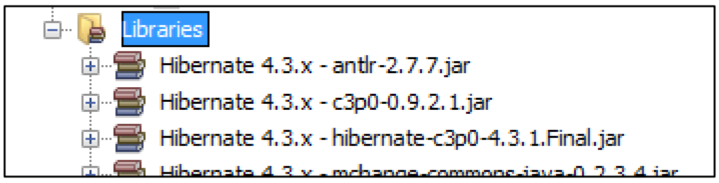
Programing Technologies
Lesson 9

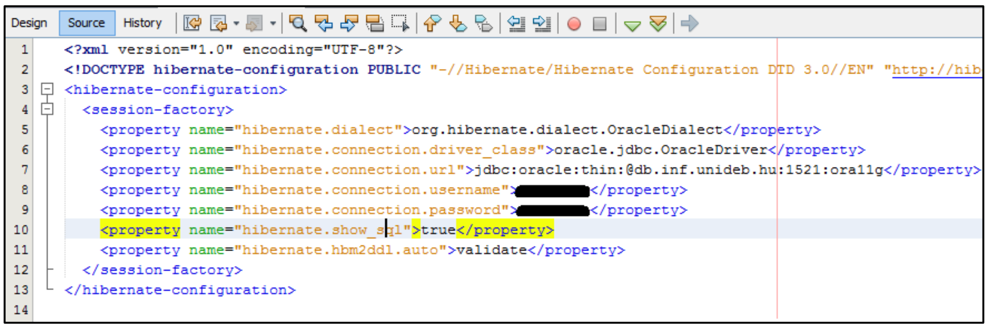
In this project the basic use of Hibernate will be presented

1. Start a new NetBeans project with a preferred name, package name and main class.
2. Include the required hibernate 4.x libraries
    - Expand your project in the Projects pane, and right click on Libraries.
    - Select add Library and add Hibernate 4.3.x
    - Result:



```
Libraries
  Hibernate 4.3.x - antlr-2.7.7.jar
  Hibernate 4.3.x - c3p0-0.9.2.1.jar
  Hibernate 4.3.x - hibernate-c3p0-4.3.1.Final.jar
  Hibernate 4.3.x - mchange-commons-java-0.2.3.4.jar
```

3. Using the Hibernate Configuration Wizard add hibernate.cfg.xml to you rproject
    - Right click on your project
    - Select New -> Other ->
      (Category: Hibernate, File Type: Hibernate Configuration Wizard)
    - Leave the name as it is and on the next page create a new database connection (by default Derby is selected).
    - The configuration for the new connection is:
        - – Driver: Oracle thin, `ojdbc6.jar` (available on the page of the class)
        - – Host: `db.inf.unideb.hu`, Port `1521`, SID: `ora11g`
        - – Use your own login information
        - – Finish the setup (Next-next-finish)
    - In the Design view of hibernate.cfg.xml under Optional properties, Configuration properties add `hibernate.show_sql: true`
    - Under Miscellaneous Properties add hibernate.hbm2ddl.auto create (Later you can set this property to validate in order not to recreate the DB, just use the existing.)
    - Check the Source view of your config file:



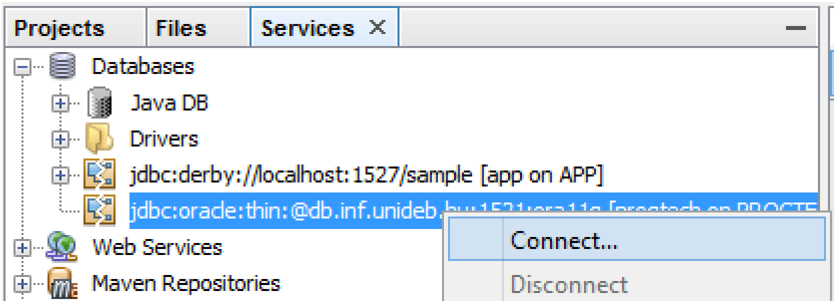
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hib
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.OracleDialect</property>
    <property name="hibernate.connection.driver_class">oracle.jdbc.OracleDriver</property>
    <property name="hibernate.connection.url">jdbc:oracle:thin:@db.inf.unideb.hu:1521:ora11g</property>
    <property name="hibernate.connection.username">[redacted]</property>
    <property name="hibernate.connection.password">[redacted]</property>
    <property name="hibernate.show_sql">true</property>
    <property name="hibernate.hbm2ddl.auto">validate</property>
  </session-factory>
</hibernate-configuration>
```

4. Add HibernateUtil.java file to your project. Under a new package hibernate.db.
   Use Right click on the project -> New -> (Category: Hibernate)

   Add an extra method to the class:
   ```
   public static void closeSessionFactory(){
       sessionFactory.close();
   }
   ```
5. Create a new Class under the `hibernate.model` package (the subpackage is not yet created). The name of it is Animal and it has the following properties:
   - gender (`enum` MALE/FEMALE),
   - age (`int`),
   - name (`String`),
   - id (`int`)

   Add getters and setters to these fields. Also add a default constructor.
6. Use persistence annotations to create the animal entity
   - Use `@Entity` and `@Table(name = “animal”)` on the `Animal` class
   - Use `@Id`, and `@GeneratedValue(strategy = GenerationType.AUTO)` on the `id` field
   - Use `@Column(name = "…", unique = … , nullable = false)` on all fields. Set unique to true for the id, and for false for the other fields.
7. In the hibernate.cfg.xml file under Mappings add the Animal class. The following line is added in the source: `<mapping class="hibernatetest.model.Animal"/>`
8. In the main method create a new hibernate session using the `HibernateUtil` class
9. Instantiate a new `Animal` and save it to the database
   - Instantiate and initialize the animal object.
   - Save the animal using the code below:

   ```
   session.beginTransaction();
   session.save(elephant);
   session.getTransaction().commit();
   session.close();
   HibernateUtil.closeSessionFactory();
   ```
10. Check the new data appearing in the database


Projects
Files
Services
Databases
Java DB
Drivers
jdbc:derby://localhost:1527/sample [app on APP]
jdbc:oracle:thin:@db.inf.unideb.hu:1521:ora11g [progtech on PROGTE
Web Services
Maven Repositories
Connect...
Disconnect

Advanced project:

1. Create a DAO (Data Access Object) class for the Animal class
    - Create a new class under the `hibernate.db` package name: `AnimalDAO`
    - Make the class implement the `AutoCloseable` interface
    - Open a new Session in the constructor, and close it in the `close()` method
    - Add simple transaction methods to add, delete and update `Animal` objects
    - Add a method to get all the saved animals. Use the code below:

```
public List<Animal> getAnimals(){
    String hql = "FROM hibernatetest.model.Animal";
    Query query = session.createQuery(hql);
    return query.list();
}
```

    Read this for more about Hibernate Query Language: http://www.tutorialspoint.com/hibernate/hibernate_query_language.htm

2. Modify the main method so that it uses the new DAO class for DB operations

Advanced project 2 - Mapping collections:

1. Add a `Zoo` class to your project. A zoo has the following fields: `id`, `name`, `animals`. This latter field is a `Set` of animals held by the zoo.
2. Annotate the `Zoo` class to be the zoo entity. Use the following annotations on the animals

```
@OneToMany(cascade = CascadeType.ALL)
@JoinColumn(name = "zoo_id")
private Set<Animal> animals;
```

field:

3. In the main method instantiate a new `Zoo` class and add an animal to it. Save the `Zoo`. Note that the `Animal` table also has been updated.

----

4. Modify the `Zoo` class by removing the animal field and the depending methods from it.
5. Add a new field to the `Animal` class. Name it `owner_zoo`. The type is `Zoo`.

1. Use these annotations on the field:

```
@ManyToOne
@Cascade(CascadeType.ALL)
@JoinColumn(name = "zoo_id")
private Zoo owner_zoo;
```

2. Create an animal and a zoo. Set the `owner_zoo` field of the animal. Persist the animal.

See more about Hibernate associations:

http://viralpatel.net/blogs/hibernate-one-to-many-annotation-tutorial/