Lab Exercise: Mapping Associations

Task 1: As explained in the lecture on Mapping Associations, create a Student-To-Guide Many-To-One uni-directional relationship where each Student has a Guide for his/her guidance. And then do the necessary mapping between them for associating them in the database with the help of a *foreign key column*.

After that, create an object of each of them, associate them, and then persist the relationship between them to the database.

Task 2: Once you've mapped the association between the Student and Guide entities, retrieve a Student by its id, and get its associated Guide object using a Student.getGuide() method.

***The source code files for the lecture on "Mapping Associations" are available to be downloaded with this lab exercise. You could use them to complete the given tasks successfully.

You could also look at the source code inside all the downloadable files below.

Source Code: Mapping Associations

Schema for hello-world database

```
CREATE DATABASE `hello-world`;
```

HibernateUtil.java

```
package util;

import org.hibernate.SessionFactory;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {

    private static final SessionFactory sessionFactory = buildSessionFactory();

    private static SessionFactory buildSessionFactory() {
        try {
            // Create the SessionFactory from hibernate.cfg.xml
            Configuration configuration = new Configuration().configure("hibernate.cfg.xml");
            return configuration.buildSessionFactory( new
StandardServiceRegistryBuilder().applySettings( configuration.getProperties() ).build() );
        }
        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;
    }

}
```

hibernate.cfg.xml

(to be placed in the classpath of your application)

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>

        <!-- Database connection settings -->
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.url">jdbc:mysql://localhost:3306/hello-world</property>
        <property name="connection.username">root</property>
        <property name="connection.password">password</property>

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

        <!-- Pretty print the SQL in the log file and console -->
        <property name="format_sql">true</property>

        <!-- Create/update tables automatically using mapping metadata -->
        <property name="hbm2ddl.auto">update</property>

        <mapping class="entity.Guide" />
        <mapping class="entity.Student" />

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

HelloWorldClient.java

```
package client;

import org.hibernate.Session;
import org.hibernate.Transaction;

import util.HibernateUtil;
import entity.Address;
import entity.Person;


public class HelloWorldClient {
   public static void main(String[] args) {

            Session session = HibernateUtil.getSessionFactory().openSession();
            Transaction txn = session.getTransaction();
            try {
               txn.begin();

                Guide guide = new Guide("2000MO10789", "Mike Lawson", 1000);
               Student student = new Student("2014JT50123", "John Smith", guide);

               session.save(guide);
               session.save(student);

               txn.commit();
            }  catch(Exception e) {
                  if(txn != null) { txn.rollback(); }
                  e.printStackTrace();
            }  finally {
                  if(session != null) { session.close(); }
            }

   }
}
```

Guide.java

```
package entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Guide {

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Long id;

    @Column(name="staff_id", nullable=false)
    private String staffId;

    private String name;
    private Integer salary;

    public Guide() {}
    public Guide(String staffId, String name, Integer salary) {
        this.staffId = staffId;
        this.name = name;
        this.salary = salary;
    }

}
```

Student.java

```
package entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;

@Entity
public class Student {

   @Id
   @GeneratedValue(strategy=GenerationType.AUTO)
   private Long id;

   @Column(name="enrollment_id", nullable=false)
   private String enrollmentId;

   private String name;

   @ManyToOne
   @JoinColumn(name="guide_id")
   private Guide guide;

   public Student() {}
   public Student(String enrollmentId, String name, Guide guide) {
      this.enrollmentId = enrollmentId;
      this.name = name;
      this.guide = guide;
   }

}
```

log4j.properties

```
# Direct to file
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=hello-world.log
log4j.appender.file.MaxFileSize=2MB
log4j.appender.file.MaxBackupIndex=1
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n

# Direct to stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n

# Root logger option
log4j.rootLogger=OFF, stdout, file

#Log everything (this will also include the logging information configured by
"log4j.logger.org.hibernate.SQL=ALL" and
"log4j.logger.org.hibernate.type.descriptor.sql.BasicBinder=TRACE")
#log4j.logger.org.hibernate=ALL

# Show SQL statements
log4j.logger.org.hibernate.SQL=ALL

# Show the bind parameter values
log4j.logger.org.hibernate.type.descriptor.sql.BasicBinder=TRACE
```