Lab Exercise: One-To-Many Relationship

Task 1: As explained in the lecture on One-To-Many Relationship, create a bi-directional One-To-Many Guide-To-Student relationship. After that, create a collection of Student objects (use the Set Collection), associate it with a Guide object, and then persist the Guide object to the database.

And also, cascade the persist operation when you're persisting the Guide object, so that when you persist the Guide object, you get its collection of students persisted with it as well.

Task 2: By default, *only* the Many side (owner of the relationship) in a bi-directional One-To-Many relationship is responsible for the relationship. Write a helper method (Guide.addStudent()) for making the One side (inverse-end) responsible for the relationship as well.

***The source code files for the lecture on "One-To-Many Relationship" 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: One-To-Many Relationship

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
<?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.Guide;
import entity.Student;
```

```
package client;

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

import util.HibernateUtil;
import entity.Guide;
import entity.Student;
```

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

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

                Guide guide1 = new Guide("2000MO10789", "Mike Lawson", 1000);
                Guide guide2 = new Guide("2000IM10901", "Ian Lamb", 2000);

                Student student1 = new Student("2014JT50123", "John Smith", guide1);
                Student student2 = new Student("2014AL50456", "Amy Gill", guide1);

                guide1.getStudents().add(student1);
                guide1.getStudents().add(student2);

                session.persist(guide1);
                session.persist(guide2);

                //Updating inverse end
                /*
                Guide guide = (Guide) session.get(Guide.class, 2L);
                Student student = (Student) session.get(Student.class, 2L);
                guide.getStudents().add(student);
                */

                //Updating owner
                /*
                Guide guide = (Guide) session.get(Guide.class, 2L);
                Student student = (Student) session.get(Student.class, 2L);
                student.setGuide(guide);
                */

                //Updating inverse end (after adding addStudent(Student) in Guide entity)
                /*
                Guide guide = (Guide) session.get(Guide.class, 2L);
                Student student = (Student) session.get(Student.class, 1L);
                guide.addStudent(student);
                */

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

    }
}
```

Guide.java

```
package entity;

import java.util.HashSet;
import java.util.Set;

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

@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;

   @OneToMany(mappedBy="guide", cascade={CascadeType.PERSIST})
   private Set<Student> students = new HashSet<Student>();

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

   public Set<Student> getStudents() {
      return students;
   }
   public void setSalary(Integer salary) {
      this.salary = salary;
   }

   public void addStudent(Student student) {
      students.add(student);
      student.setGuide(this);
   }

}
```

Student.java

```java
package entity;

import javax.persistence.CascadeType;
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(cascade={CascadeType.PERSIST, CascadeType.REMOVE})
   @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;
   }

   public Guide getGuide() {
      return guide;
   }
   public void setGuide(Guide guide) {
      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
```