Lab Exercise: Lazy Fetching

Task 1: As explained in the lecture on Lazy Fetching, program for the condition which will throw an org.hibernate.LazyInitializationException exception.

Task 2: Program for the condition when associated collection of students is also fetched at the same time when a Guide is retrieved by its id value.

Task 3: Retrieve a Student by its id value and find out if its associated Guide is also fetched with it.

***The source code files for the lecture on "Lazy Fetching" 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: Lazy Fetching

Schema for hello-world database

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

persistence.xml

(to be placed inside META-INF folder; the META-INF folder should be in the root of the classpath of your application)

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
   version="2.0" xmlns="http://java.sun.com/xml/ns/persistence">

   <persistence-unit name="hello-world" transaction-type="RESOURCE_LOCAL">
      <properties>

         <!-- Database connection settings -->
         <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
         <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/hello-
world" />
         <property name="javax.persistence.jdbc.user" value="root" />
         <property name="javax.persistence.jdbc.password" value="password" />

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

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

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

   </persistence-unit>
</persistence>
```

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
```

HelloWorldClient.java

```
package client;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

import entity.Guide;
import entity.Student;

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

      EntityManagerFactory emf = Persistence.createEntityManagerFactory("hello-world");
      EntityManager em = emf.createEntityManager();
      EntityTransaction txn = em.getTransaction();
            try {
               txn.begin();

               //Lazy Collection Fetching with default settings(fetch=FetchType.LAZY for
collection associations)
               Guide guide = em.find(Guide.class, 2L);
               Set<Student> students = guide.getStudents();
               int numberOfStudents = students.size();

               //Eager Fetching with default settings (fetch=FetchType.EAGER for single point
associations)
               //Student student = em.find(Student.class, 2L);

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

```
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;
   }
}
```