Ex10_LuisZüttel_GionRubitschung_D1P

May 9, 2024

1 Exercise 1 (JDBC)

1. Understand the code in module `jdbc-assignment`.
2. Run the main function in class `JdbcExample`, it should output a list of instructors.
3. Adapt the code to output a list of departments with the average salary of their instructors.

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

        // just check if we can load the JDBC driver
        try {
            Class.forName("org.sqlite.JDBC");
        } catch (ClassNotFoundException e) {
            throw new Error("Cannot find JDBC Driver", e);
        }

        // establish connection and fetch and print some data
        try (Connection conn =
                     DriverManager.getConnection(
                             "jdbc:sqlite:../../../data/university.db")) {
            try (Statement stmt = conn.createStatement()) {
                ResultSet rs = stmt.executeQuery("select * from instructor");
                while (rs.next()) {
                    String name = rs.getString("name");
                    System.out.println(name);
                }
            } catch (SQLException e) {
                throw new Error("Problem executing query", e);
            }

        } catch (SQLException e) {
            throw new Error("Cannot establish database connection", e);
        }
    }
}
```

2 Exercise 2 (JPA)

1. Understand the code in module `jpa-assignment`.
2. Run the main function in class `JPAExample`, it should output a list of instructors.
3. Adapt the code to output each department (not just the `dept_name`). For that, you'll have to write an appropiate `Department`class with a `toString` method and add it to `persistence.xml`.

```
@Entity
public class Department {
    @Id
    private String dept_name;

    private String building;

    private float budget;

    public Department() {
        super();
    }

    @Override
    public String toString() {
        return "Department [dept_name=" + dept_name +
                ", building=" + building +
                ", budget=" + budget + "]";
    }
}
```

4. Adapt the code to output each department together wit the names of the instructors working for that department. For that, add a bidirectional relationship between `Instructor` and `Department`. In particular, the `Instructor` class will need a `department` member and the `Department` class will need an `instructors` member and a public `getInstructors` method. You will also need to tell JPA the name of the foreign key column in `Instructor` that references `Department`. You can do this with the `@JoinColumn(name = "dept_name")` annotation on the `department` member.

```
@Entity
public class Department {
    @Id
    private String dept_name;

    private String building;

    private float budget;

    private List<Instructor> instructors;

    public Department() {
```

```java
        super();
    }

    public List<Instructor> getInstructors() {
        return instructors;
    }

    @Override
    public String toString() {
        return "Department [dept_name=" + dept_name +
                ", building=" + building +
                ", budget=" + budget + "]";
    }
}

@Entity
public class Instructor {
    @Id
    private Integer id;

    private String name;

    @ManyToOne
    @JoinColumn(name = "dept_name")
    private Department department;

    private float salary;

    public Instructor() {
        super();
    }

    @Override
    public String toString() {
        return "Instructor [id=" + id +
                ", name=" + name +
                ", dept_name=" + dept_name +
                ", salary=" + salary + "]";
    }
}

public class JPAExample {

    public static void main(String[] args) {

        EntityManagerFactory factory =
                Persistence.createEntityManagerFactory("university");
        EntityManager em = factory.createEntityManager();
        TypedQuery<Instructor> q1 = em.createQuery("SELECT e FROM Department e", Department.cla
        List<Department> departments = q1.getResultList();

        for (Department department : departments) {
            System.out.println(department);
            List<Instructor> instructors = department.getInstructors();
            for (Instructor instructor : instructors) {
                System.out.println(instructor);
            }
        }

        em.close();
        factory.close();
    }
}
```