Experiment-9

Student Name: Deepu Jose
UID: 22BCS15707
Branch: BE-CSE
Section/Group: IOT-618/B
Semester: 6th
Date of Performance: 11/04/25
Subject Name: PBLJ-Lab
Subject Code: 22CSH-359

1. Aim:

Easy Level:
Create a Spring application using Java-based configuration to inject a Course into a Student using @Configuration and @Bean.

Medium Level:
Build a Hibernate app to perform CRUD on a Student entity using MySQL.

Hard Level:
Create a Spring-based application integrated with Hibernate ORM that:

- Transfers funds between accounts.
- Rolls back in case of failure.
- Demonstrates both successful and failed transactions.

2. Objective:

a.) Understand the Servlet Lifecycle
Learn how servlets are created, executed, and destroyed in a web application.

b.) Learn HTTP Servlet and Request-Response Mechanism
Understand how HTTP requests (GET and POST) work and how servlets handle user inputs and responses.

c.) Implement Form Handling using Servlets
Develop a servlet to accept user credentials via an HTML form and display a response.

3. Implementation/Code:

a.) Easy Problem:

1. HTML Code:

```
public class Course {
    private String courseName;
    private int duration;
```

```
    public Course(String courseName, int duration) {
        this.courseName = courseName;
        this.duration = duration;
    }

    public String getCourseName() {
        return courseName;
    }

    public int getDuration() {
        return duration;
    }

    @Override
    public String toString() {
        return "Course{name='" + courseName + "', duration=" + duration + " weeks}";
    }
}
```

1. Student:

```
public class Student {

    private String name;

    private Course course;



    public Student(String name, Course course) {

        this.name = name;

        this.course = course;

    }



    public void printDetails() {

        System.out.println("Student Name: " + name);
```

```
        System.out.println("Enrolled Course: " + course);
    }
}
```

App config:

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class AppConfig {

    @Bean
    public Course course() {
        return new Course("Spring Framework", 6);
    }

    @Bean
    public Student student() {
        return new Student("Alice", course());
    }
}
import org.springframework.context.ApplicationContext;
```

```
import org.springframework.context.annotation.AnnotationConfigApplicationContext;


public class Main {
    public static void main(String[] args) {
        ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);
        Student student = context.getBean(Student.class);
        student.printDetails();
    }
}
```

b. Medium Problem:

XML

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
    <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/testdb</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password">yourpassword</property>
```

```
    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
    <property name="hibernate.hbm2ddl.auto">update</property>
    <property name="show_sql">true</property>
    <mapping class="Student"/>
  </session-factory>
</hibernate-configuration>
import javax.persistence.*;


@Entity
public class Student {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;


    private String name;
    private int age;


    public Student() {}


    public Student(String name, int age) {
        this.name = name;
```

```
        this.age = age;
    }

    // Getters and Setters
}
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {
    private static final SessionFactory sessionFactory = buildSessionFactory();

    private static SessionFactory buildSessionFactory() {
        try {
            return new Configuration().configure("hibernate.cfg.xml").buildSessionFactory();
        } catch (Exception e) {
            throw new ExceptionInInitializerError(e);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
```

```
    }
}
import org.hibernate.Session;
import org.hibernate.Transaction;

public class Main {
    public static void main(String[] args) {
        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();

        // Create
        Student s1 = new Student("Bob", 22);
        session.save(s1);

        // Read
        Student readStudent = session.get(Student.class, 1);
        System.out.println("Read Student: " + readStudent.getName());

        // Update
        readStudent.setAge(23);
        session.update(readStudent);
```

```
        // Delete
        session.delete(readStudent);

        tx.commit();
        session.close();
    }
}
```

c) Hard Problem:

```
CREATE DATABASE bankdb;

USE bankdb;

CREATE TABLE Account (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50),
    balance DOUBLE
);
import javax.persistence.*;
```

```
@Entity
public class Account {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;
    private double balance;

    public Account() {}
    public Account(String name, double balance) {
        this.name = name;
        this.balance = balance;
    }

    // Getters and Setters
}
import javax.persistence.*;
import java.util.Date;

@Entity
```

```
public class TransactionLog {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String description;
    private Date date;

    public TransactionLog() {}
    public TransactionLog(String description) {
        this.description = description;
        this.date = new Date();
    }

    // Getters and Setters
}
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/bankdb</property>
```

```
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">yourpassword</property>
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
        <property name="hibernate.hbm2ddl.auto">update</property>
        <property name="show_sql">true</property>

        <mapping class="Account"/>
        <mapping class="TransactionLog"/>
    </session-factory>
</hibernate-configuration>
import org.springframework.context.annotation.*;
import org.springframework.orm.hibernate5.HibernateTransactionManager;
import org.springframework.orm.hibernate5.LocalSessionFactoryBean;

import javax.sql.DataSource;
import org.springframework.jdbc.datasource.DriverManagerDataSource;
import java.util.Properties;

@Configuration
@EnableTransactionManagement
public class AppConfig {
```

```
@Bean
public DataSource dataSource() {
    DriverManagerDataSource ds = new DriverManagerDataSource();
    ds.setDriverClassName("com.mysql.cj.jdbc.Driver");
    ds.setUrl("jdbc:mysql://localhost:3306/bankdb");
    ds.setUsername("root");
    ds.setPassword("yourpassword");
    return ds;
}

@Bean
public LocalSessionFactoryBean sessionFactory() {
    LocalSessionFactoryBean factory = new LocalSessionFactoryBean();
    factory.setDataSource(dataSource());
    factory.setPackagesToScan("your.package"); // replace with actual package
    Properties props = new Properties();
    props.setProperty("hibernate.dialect", "org.hibernate.dialect.MySQLDialect");
    props.setProperty("hibernate.hbm2ddl.auto", "update");
    props.setProperty("show_sql", "true");
    factory.setHibernateProperties(props);
```

```
        return factory;
    }

    @Bean
    public HibernateTransactionManager transactionManager() {
        HibernateTransactionManager txManager = new HibernateTransactionManager();
        txManager.setSessionFactory(sessionFactory().getObject());
        return txManager;
    }
}
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.transaction.annotation.Transactional;

public class BankingService {

    private SessionFactory sessionFactory;

    public BankingService(SessionFactory sessionFactory) {
        this.sessionFactory = sessionFactory;
    }
```

```
@Transactional
public void transferMoney(int fromId, int toId, double amount) {
    Session session = sessionFactory.getCurrentSession();

    Account from = session.get(Account.class, fromId);
    Account to = session.get(Account.class, toId);

    if (from.getBalance() < amount) {
        throw new RuntimeException("Insufficient balance!");
    }

    from.setBalance(from.getBalance() - amount);
    to.setBalance(to.getBalance() + amount);

    session.update(from);
    session.update(to);

    // Optional: Add transaction log
    session.save(new TransactionLog("Transferred ₹" + amount + " from " + from.getName() +
" to " + to.getName()));
```

```
    }
}
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;


public class Main {
    public static void main(String[] args) {
        ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);
        BankingService service = new BankingService(context.getBean(SessionFactory.class));


        try {
            service.transferMoney(1, 2, 1000.0);
            System.out.println("Transaction successful.");
        } catch (Exception e) {
            System.out.println("Transaction failed: " + e.getMessage());
        }
    }
}
```

4. Output:

a.)

```
Student Name: Alice
Enrolled Course: Course{name='Spring Framework', duration=6 weeks}
```

b.)

Hibernate: insert into Student (age, name) values (?, ?)
Hibernate: select student0_.id as id1_0_0_, student0_.age as age2_0_0_, student0_.name as name3_0_0_ from Student student0_ where student0_.id=?
Read Student: Bob
Hibernate: update Student set age=? where id=?
Hibernate: delete from Student where id=?

c.)

```
Transaction successful.
Hibernate: update Account set balance=? where id=?
Hibernate: update Account set balance=? where id=?
Hibernate: insert into TransactionLog (description, date) values (?, ?)
```

```
Transaction failed: Insufficient balance!
(no changes committed, rollback done)
```

5. Learning Outcome:

1. Understand and Implement Servlets in Web Applications
2. Handle User Input and Process Requests Dynamically
3. Connect a Web Application to a Database using JDBC