Experiment -9

Student Name: Aniket yadav **UID:22BCS11646**

Branch: BE-CSE **Section/Group:903/A**

Semester:6th **Date of Performance:17/03/2025**

Subject Name: Project-Based Learning in Java with Lab **Subject Code: 22CSH-359**

9.1.1.Aim: To demonstrate dependency injection using Spring Framework with Java-based configuration.

9.1.2 Objective:

Define Course and Student classes.
Use Configuration and Bean annotations to inject dependencies.
Load Spring context and print student details.

9.1.3 Code:

```
// Course.java
public class Course {
    private String courseName;
    private String duration;

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

    public String getCourseName() { return courseName; }
    public String getDuration() { return duration; }

    @Override
    public String toString() {
        return "Course: " + courseName + ", Duration: " + duration;
    }
}

// Student.java
public class Student {
    private String name;
    private Course course;
```

```
    public Student(String name, Course course) {
        this.name = name;
        this.course = course;
    }

    public void showDetails() {
        System.out.println("Student: " + name);
        System.out.println(course);
    }
}// AppConfig.java
import org.springframework.context.annotation.*;

@Configuration
public class AppConfig {
    @Bean
    public Course course() {
        return new Course("Java", "3 months");
    }

    @Bean
    public Student student() {
        return new Student("Aman", course());
    }
}// MainApp.java
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

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

Output:

```
Student: Sarthak
Course: Java, Duration: 3 months
```

9.2.1 Aim: To perform CRUD operations on a Student entity using Hibernate ORM with MySQL.

Objective: Define Course and Student classes.

Use Configuration and Bean annotations to inject dependencies.

Load Spring context and print student details.

9.2.2 Code:

```
        <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">password</property>
            <property
name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>
            <property name="hibernate.hbm2ddl.auto">update</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, setters, toString
}
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {
    private static final SessionFactory sessionFactory;

    static {
        sessionFactory = new Configuration().configure().buildSessionFactory();
    }

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

import org.hibernate.*;

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

        // Create
        Transaction tx = session.beginTransaction();
        Student s1 = new Student("Aman", 22);
        session.save(s1);
        tx.commit();

        // Read
        Student student = session.get(Student.class, 1);
        System.out.println(student);

        // Update
        tx = session.beginTransaction();
        student.setAge(23);
        session.update(student);
        tx.commit();

        // Delete
        tx = session.beginTransaction();
        session.delete(student);
```

```
        tx.commit();

        session.close();
    }
}
```

9.2.3 Output:

```
Student{id=1, name='Sallu', age=22}
Updated age to 23
Deleted student with id 1
```

9.3.1 Aim: To implement a banking system using Spring and Hibernate that ensures transaction consistency during fund transfers.

Objective:

Integrate Spring + Hibernate.

Handle transactions atomically (rollback on failure).

Demonstrate success and failure cases.

Code:

```
import javax.persistence.*;

Entity
public class Account {
   @Id
   private int accountId;
   private String holderName;
   private double balance;

   // Constructors, getters, setters
}

import javax.persistence.*;
import java.util.Date;

@Entity
public class BankTransaction {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private int txnId;
   private int fromAcc;
   private int toAcc;
   private double amount;
   private Date txnDate = new Date();

   // Constructors, getters, setters
}
import org.hibernate.*;
import org.springframework.transaction.annotation.Transactional;
```

```
public class BankService {
    private SessionFactory sessionFactory;

    public BankService(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);

        BankTransaction txn = new BankTransaction(fromId, toId, amount);
        session.save(txn);
    }
}
@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/testdb");
        ds.setUsername("root");
        ds.setPassword("password");
```

```
        return ds;
    }

    @Bean
    public LocalSessionFactoryBean sessionFactory() {
        LocalSessionFactoryBean lsf = new LocalSessionFactoryBean();
        lsf.setDataSource(dataSource());
        lsf.setPackagesToScan("your.package");
        Properties props = new Properties();
        props.put("hibernate.dialect", "org.hibernate.dialect.MySQL8Dialect");
        props.put("hibernate.hbm2ddl.auto", "update");
        lsf.setHibernateProperties(props);
        return lsf;
    }

    @Bean
    public HibernateTransactionManager transactionManager(SessionFactory sf) {
        return new HibernateTransactionManager(sf);
    }

    @Bean
    public BankService bankService(SessionFactory sf) {
        return new BankService(sf);
    }
}

public class MainApp {
    public static void main(String[] args) {
        AnnotationConfigApplicationContext ctx = new
AnnotationConfigApplicationContext(AppConfig.class);
        BankService service = ctx.getBean(BankService.class);

        try {
            service.transferMoney(101, 102, 500);
            System.out.println("Transaction Successful!");
        } catch (Exception e) {
            System.out.println("Transaction Failed: " + e.getMessage());
        }
```

```
        ctx.close();
    }
}
```

OUTPUT

```
Transaction Successful!
OR
Transaction Failed: Insufficient Balance
```

Experiment -8

Student Name: Arpit Sharma **UID:22BCS16900**

Branch: BE-CSE **Section/Group:IOT_615-A**

Semester:6th **Date of Performance:17/03/2025**

Subject Name: Project-Based Learning in Java with Lab **Subject Code: 22CSH-359**

7.1.1.Aim: To develop a servlet that accepts user credentials from an HTML form and displays a personalized welcome message on successful login.

7.1.2 Objective: Learn form handling with Servlets
Understand HTTP request/response handling
Practice doPost() method

7.1.3 Code:

```
<!DOCTYPE html>
<html>
<head><title>Login</title></head>
<body>
  <form action="LoginServlet" method="post">
    Username: <input type="text" name="username"><br>
    Password: <input type="password" name="password"><br>
    <input type="submit" value="Login">
  </form>
</body>
</html>
```

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class LoginServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String user = request.getParameter("username");
        String pass = request.getParameter("password");

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        if ("admin".equals(user) && "1234".equals(pass)) {
```

```
            out.println("<h2>Welcome, " + user + "!</h2>");
        } else {
            out.println("<h2>Login Failed. Invalid credentials.</h2>");
        }
    }
}
```

```
<web-app>
  <servlet>
    <servlet-name>LoginServlet</servlet-name>
    <servlet-class>LoginServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>LoginServlet</servlet-name>
    <url-pattern>/LoginServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

Output:

1) On correct login: Welcome, Sarthak!
2) On failure: Login Failed. Invalid credentials.

7.2.1 Aim: To build a servlet integrated with JDBC that displays all employees and enables search by employee ID.

Objective: 1) Use JDBC with Servlet

2) Fetch and display records

3) Implement search functionality

7.2.2 Code:

```
<!DOCTYPE html>
<html>
<head><title>Search Employee</title></head>
<body>
  <form action="EmployeeServlet" method="post">
    Enter Employee ID: <input type="text" name="empId">
    <input type="submit" value="Search">
  </form>
</body>
</html>
```

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class EmployeeServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        String empId = request.getParameter("empId");
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/company", "root", "password");

            String query = "SELECT * FROM employees WHERE emp_id=?";
            PreparedStatement ps = con.prepareStatement(query);
```

```
            ps.setString(1, empId);
            ResultSet rs = ps.executeQuery();

            if (rs.next()) {
                out.println("<h2>Employee Details</h2>");
                out.println("ID: " + rs.getInt(1) + "<br>");
                out.println("Name: " + rs.getString(2) + "<br>");
                out.println("Department: " + rs.getString(3));
            } else {
                out.println("No employee found with ID " + empId);
            }

            con.close();
        } catch (Exception e) {
            out.println("Error: " + e.getMessage());
        }
    }
}
```

7.2.3 Output:

1) Enter an employee ID → Shows details if found.

2) Not found → "No employee found with ID X

7.3.1 Aim: To develop a JSP-based student portal that accepts attendance data and saves it to the database using a servlet.

Objective: 1) Combine JSP for UI and Servlets for logic
2) Perform INSERT using JDBC
3) Build a real-world web flow

Code:

```
<%@ page language="java" %>
<html>
<head><title>Student Attendance</title></head>
<body>
  <h2>Mark Attendance</h2>
  <form action="AttendanceServlet" method="post">
    Roll No: <input type="text" name="roll"><br>
    Name: <input type="text" name="name"><br>
    Status: <select name="status">
       <option>Present</option>
       <option>Absent</option>
    </select><br>
    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class AttendanceServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String roll = request.getParameter("roll");
        String name = request.getParameter("name");
        String status = request.getParameter("status");

        response.setContentType("text/html");
```

```
        PrintWriter out = response.getWriter();

        try {
            Class.forName("com.mysql.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/student_portal", "root",
"password");

            String query = "INSERT INTO attendance (roll_no, name, status) VALUES (?, ?,
?)";
            PreparedStatement ps = con.prepareStatement(query);
            ps.setString(1, roll);
            ps.setString(2, name);
            ps.setString(3, status);

            int i = ps.executeUpdate();
            if (i > 0) {
                out.println("<h3>Attendance marked successfully for " + name + "!</h3>");
            }

            con.close();
        } catch (Exception e) {
            out.println("Error: " + e.getMessage());
        }
    }
}

CREATE TABLE attendance (
    id INT AUTO_INCREMENT PRIMARY KEY,
    roll_no VARCHAR(20),
    name VARCHAR(100),
    status VARCHAR(10)
);
```

OUTPUT

Form submission → "Attendance marked successfully for John!"
And the data is stored in the database.