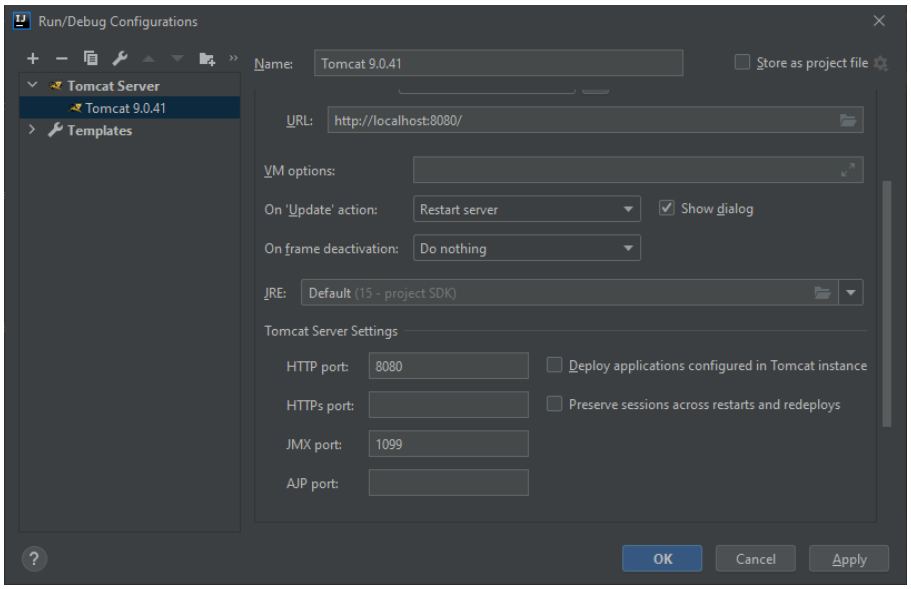
21. Spring Framework (MVC + AOP + Hibernate + JSP)

- Пример mvc приложения, для изменения данных в базе
- Отображение страниц в формате jsp

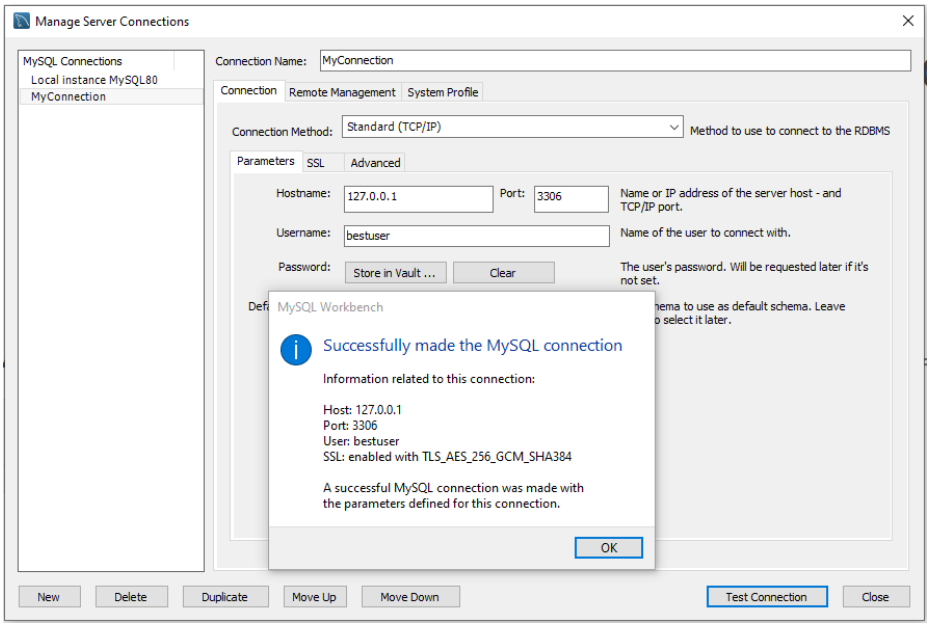
Pom.xml (добавление зависимостей):

```
<dependencies>

  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>5.2.12.RELEASE</version>
  </dependency>

  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
  </dependency>

  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.4.27.Final</version>
  </dependency>

  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.22</version>
  </dependency>

  <dependency>
    <groupId>com.mchange</groupId>
    <artifactId>c3p0</artifactId>
    <version>0.9.5.2</version>
  </dependency>

  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-orm</artifactId>
    <version>5.2.9.RELEASE</version>
  </dependency>

  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-orm</artifactId>
    <version>5.2.12.RELEASE</version>
  </dependency>

  <dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjweaver</artifactId>
    <version>1.9.6</version>
  </dependency>

</dependencies>
```

Добавление tomcat-сервера:

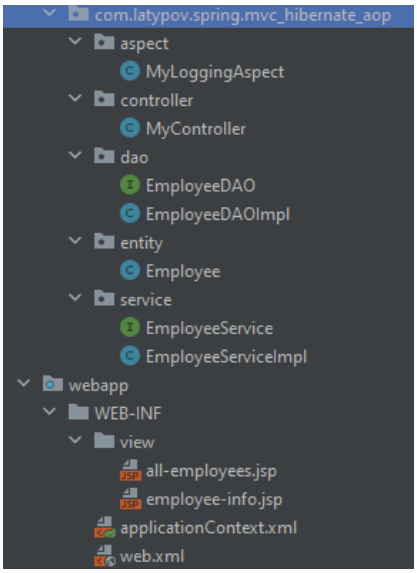


Создание соединения MySQLServer:



Сценарий: создание веб-страниц, посредством которых будет осуществляться модификация данных в базе средствами Hibernate и их отображение. Добавление логгирования для основных методов

Структура проекта:



Web.xml (аналог класса DispatcherInitializer, конфигурация DispatcherServlet параметров):

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         id="WebApp_ID" version="3.1">

  <display-name>spring-cource-mvc-hibernate-aop</display-name>

  <absolute-ordering />

  <servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>/WEB-INF/applicationContext.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

</web-app>
```

applicationContext.xml (основная конфигурация, добавление схемы, определение бинов, объекты ViewResolver, DataSource, SessionFactory, TransactionalManager):

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xmlns:tx="http://www.springframework.org/schema/tx"

       xmlns:aop="http://www.springframework.org/schema/aop"

       xsi:schemaLocation="
      http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context.xsd
       http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd
      http://www.springframework.org/schema/tx
      http://www.springframework.org/schema/tx/spring-tx.xsd

        http://www.springframework.org/schema/aop
      http://www.springframework.org/schema/aop/spring-aop.xsd">

    <context:component-scan base-
package="com.latypov.spring.mvc_hibernate_aop" />

    <mvc:annotation-driven/>
    <aop:aspectj-autoproxy/>

    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/view/" />
        <property name="suffix" value=".jsp" />
    </bean>

    <!--Создание C3po pool connection -->
    <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"
          destroy-method="close">
        <property name="driverClass" value="com.mysql.cj.jdbc.Driver" />
        <property name="jdbcUrl"
value="jdbc:mysql://localhost:3306/my_db?useSSL=false&amp;serverTimezone=UTC"
/>
        <property name="user" value="bestuser" />
        <property name="password" value="bestuser" />
    </bean>


    <bean id="sessionFactory"
          class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="packagesToScan"
value="com.latypov.spring.mvc_hibernate_aop.entity" />
        <property name="hibernateProperties">
            <props>
                <prop
key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>
                <prop key="hibernate.show_sql">true</prop>
            </props>
        </property>
    </bean>

    <!--Управление транзакциями -->
    <bean id="transactionManager"

class="org.springframework.orm.hibernate5.HibernateTransactionManager">
        <property name="sessionFactory" ref="sessionFactory"/>
```

```
    </bean>

    <!--Аннотация Transactional -->
    <tx:annotation-driven transaction-manager="transactionManager" />


</beans>
```

Создание таблицы Employees, и добавление данных:

```
USE my_db;

CREATE TABLE employees (
  id int NOT NULL AUTO_INCREMENT,
  name varchar(15),
  surname varchar(25),
  department varchar(20),
  salary int,
  PRIMARY KEY (id)
) ;

INSERT INTO my_db.employees (name, surname, department, salary)
VALUES
   ('Linar', 'Latypov', 'IT', 500),
   ('Oleg', 'Ivanov', 'Sales', 700),
   ('Nina', 'Sidorova', 'HR', 850);
```

entity.Employee(связь с таблицей Employees в БД):

```
@Entity
@Table(name = "employees")
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;
    @Column(name = "name")
    private String name;
    @Column(name = "surname")
    private String surname;
    @Column(name = "department")
    private String department;
    @Column(name = "salary")
    private int salary;
```

dao.EmployeeDAO (интерфейс с методами для обработки данных):

```
public interface EmployeeDAO {
    public List<Employee> getAllEmployees();
    public void saveEmployee(Employee employee);
    public Employee getEmployee(int id);
    public void deleteEmployee(int id);
}
```

dao.EmployeeDAOImpl (имплементация методов из интерфейса EmployeeDAO, аннотация @Repository):

```
@Repository
public class EmployeeDAOImpl implements EmployeeDAO {
    @Autowired
    private SessionFactory sessionFactory;
    @Override

    public List<Employee> getAllEmployees() {
        Session session = sessionFactory.getCurrentSession();
        Query<Employee> query = session.createQuery("from Employee",
Employee.class);
        List<Employee> allEmployees = query.getResultList();
        return allEmployees;
    }

    @Override
    public void saveEmployee(Employee employee) {
        Session session = sessionFactory.getCurrentSession();
        session.saveOrUpdate(employee);

    }

    @Override
    public Employee getEmployee(int id) {
        Session session = sessionFactory.getCurrentSession();
        Employee employee = session.get(Employee.class, id);
        return employee ;
    }

    @Override
    public void deleteEmployee(int id) {
        Session session = sessionFactory.getCurrentSession();
        Query<Employee> query = session.createQuery("delete from Employee " +
                "where id =: employeeId");
        query.setParameter("employeeId", id);
        query.executeUpdate();
    }
```

service.EmployeeService (интерфейс сервисной части, методы обработки данных):

```
public interface EmployeeService {
    public List<Employee> getAllEmployees();
    public void saveEmployee(Employee employee);
    public Employee getEmployee(int id);
    public void deleteEmployee(int id);
}
```

service.EmployeeServiceImpl (имплементация методов, связь с DAO, аннотация @Service, @Transactional):

```
@Service
public class EmployeeServiceImpl implements EmployeeService {
```

```
    @Autowired
    private EmployeeDAO employeeDAO;

    @Override
    @Transactional
    public List<Employee> getAllEmployees() {
        return employeeDAO.getAllEmployees();
    }

    @Override
    @Transactional
    public void saveEmployee(Employee employee) {
        employeeDAO.saveEmployee(employee);
    }

    @Override
    @Transactional
    public Employee getEmployee(int id) {
        return employeeDAO.getEmployee(id);
    }

    @Override
    @Transactional
    public void deleteEmployee(int id) {
        employeeDAO.deleteEmployee(id);
    }
}
```

controller.MyController (определение маппингов для основной логики, добавление данных в модель и вывод результата, @Controller, @RequestMapping, @RequestParam):

```
@Controller
public class MyController {
    @Autowired
    private EmployeeService employeeService;

    @RequestMapping("/")
    public String showAllEmployees(Model model){
        List<Employee> allEmployees = employeeService.getAllEmployees();
        model.addAttribute("allEmps", allEmployees);
        return "all-employees";
    }

    @RequestMapping("/addNewEmployee")
    public String addNewEmployee(Model model){
        Employee employee = new Employee();
        model.addAttribute("employee", employee);
        return "employee-info";
    }
    @RequestMapping("/saveEmployee")
    public String saveEmployee(@ModelAttribute("employee") Employee
employee){
        employeeService.saveEmployee(employee);
        return "redirect:/";
    }

    @RequestMapping("/updateInfo")
    public String updateEmployee(@RequestParam("empId") int id, Model model){
        Employee employee = employeeService.getEmployee(id);
```

```
        model.addAttribute("employee", employee);
        return "employee-info";
    }

    @RequestMapping("/deleteEmployee")
    public String deleteEmployee(@RequestParam("empId") int id){
        employeeService.deleteEmployee(id);

    return "redirect:/";
    }
```

aspect.MyLoggingAspect (добавление логгирования для каждого метода, @Aspect, @Around, ProceedingJoinPoint):

```
@Component
@Aspect
public class MyLoggingAspect {
    @Around("execution(* com.latypov.spring.mvc_hibernate_aop.dao.*.*(..))")
    public Object aroundAllRepositoryMethodsAdvice(ProceedingJoinPoint
proceedingJoinPoint) throws Throwable{
        MethodSignature methodSignature = (MethodSignature)
proceedingJoinPoint.getSignature();
        String methodName = methodSignature.getName();
        System.out.println("Begin of " + methodName);
        Object targetMethpodResult = proceedingJoinPoint.proceed();
        System.out.println("End of " + methodName);
        return targetMethpodResult;
    }
}
```

view.all-employees.jsp (вывод списка Employee в форме таблицы, с кнопками редактирования и добавления, использование цикла for):

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
<h2>All employees</h2>
<br>
<table>
    <tr>
        <th>Name</th>
        <th>Surname</th>
        <th>Department</th>
        <th>Salary</th>
        <th>Operations</th>
    </tr>
    <c:forEach var="emp" items="${allEmps}">
        <c:url var="updateButton" value="/updateInfo">
            <c:param name="empId" value="${emp.id}"/>
        </c:url>
        <c:url var="deleteButton" value="/deleteEmployee">
            <c:param name="empId" value="${emp.id}"/>
        </c:url>
```

```
        <tr>
            <td>${emp.name}</td>
            <td>${emp.surname}</td>
            <td>${emp.department}</td>
            <td>${emp.salary}</td>
            <td><input type="button"
                       value="Update"
                       onclick="window.location.href='${updateButton}'"/>
                <input type="button"
                       value="Delete"
                       onclick="window.location.href='${deleteButton}'"/>
            </td>
        </tr>
    </c:forEach>
    <br>
    <br>
    <input type="button" value="Add"
           onclick="window.location.href ='addNewEmployee'"/>
</table>
</body>
</html>
```

view.employee-info.jsp (отображение данных конкретного работника, использование форм):

```
<body>
<h2>Employee info</h2>
<br>
<form:form action="saveEmployee" modelAttribute="employee">
    <form:hidden path="id"/>

    Name<form:input path="name"/>
    <br><br>
    Surname<form:input path="surname"/>
    <br><br>
    Department<form:input path="department"/>
    <br><br>
    Salary<form:input path="salary"/>
    <br><br>
    <input type="submit" value="OK">

</form:form>
</body>
```

Интерфейс

Основное окно:

Информация о конкретном работнике: