Министерство образования и науки Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение**

**науки и высшего образования**

«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

**Кафедра информационных систем и программной инженерии**

Лабораторная работа №1

по дисциплине

«Распределённые программные системы»

Выполнил:

ст. гр. ПРИ-117

Дмитриев М.А.

Принял:

Трифонов Д.А.

Владимир, 2020

**Цель работы**

Познакомиться с принципами построения веб-приложений с использованием шаблона MVC и на базе принципов REST, получить практические навыки реализации MVC приложения на платформе Spring MVC.

**Ход работы**

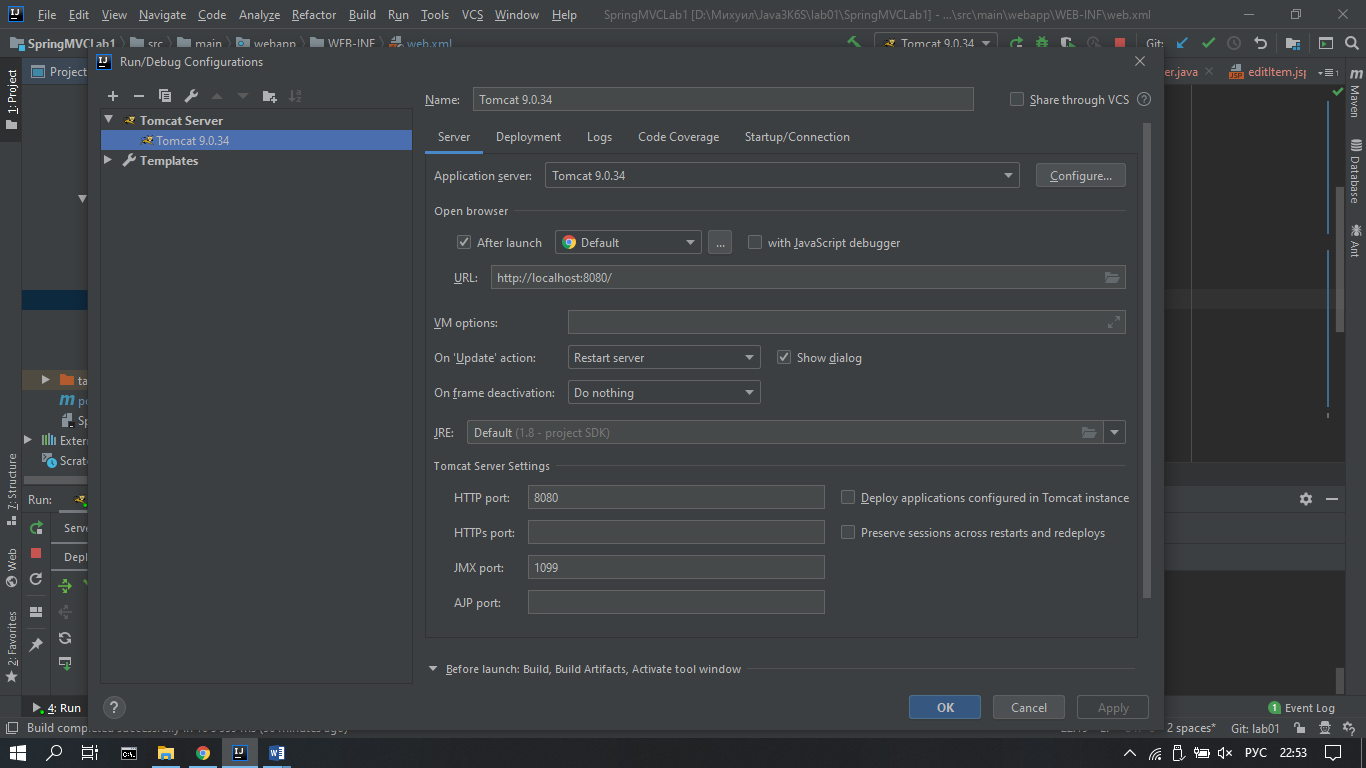
**Ссылка на github:** [https://github.com/myplusher/DSS3k6s.git](https://vk.com/away.php?to=https%3A%2F%2Fgithub.com%2Fmyplusher%2FDSS3k6s.git&cc_key=)

Создание maven проекта на основе webapp archetype.

Добавление необходимых зависимостей в pom.

<dependencies>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.11</version>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-jdbc</artifactId>  
 <version>5.2.3.RELEASE</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>5.2.3.RELEASE</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-core</artifactId>  
 <version>5.2.3.RELEASE</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.2.3.RELEASE</version>  
 </dependency>  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>8.0.11</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-web</artifactId>  
 <version>5.2.3.RELEASE</version>  
 </dependency>  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>servlet-api</artifactId>  
 <version>3.0-alpha-1</version>  
 </dependency>  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>jstl</artifactId>  
 <version>1.2</version>  
 </dependency>  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-api</artifactId>  
 <version>1.7.25</version>  
 </dependency>  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-log4j12</artifactId>  
 <version>1.7.25</version>  
 </dependency>  
</dependencies>

Подключение tomcat сервера.

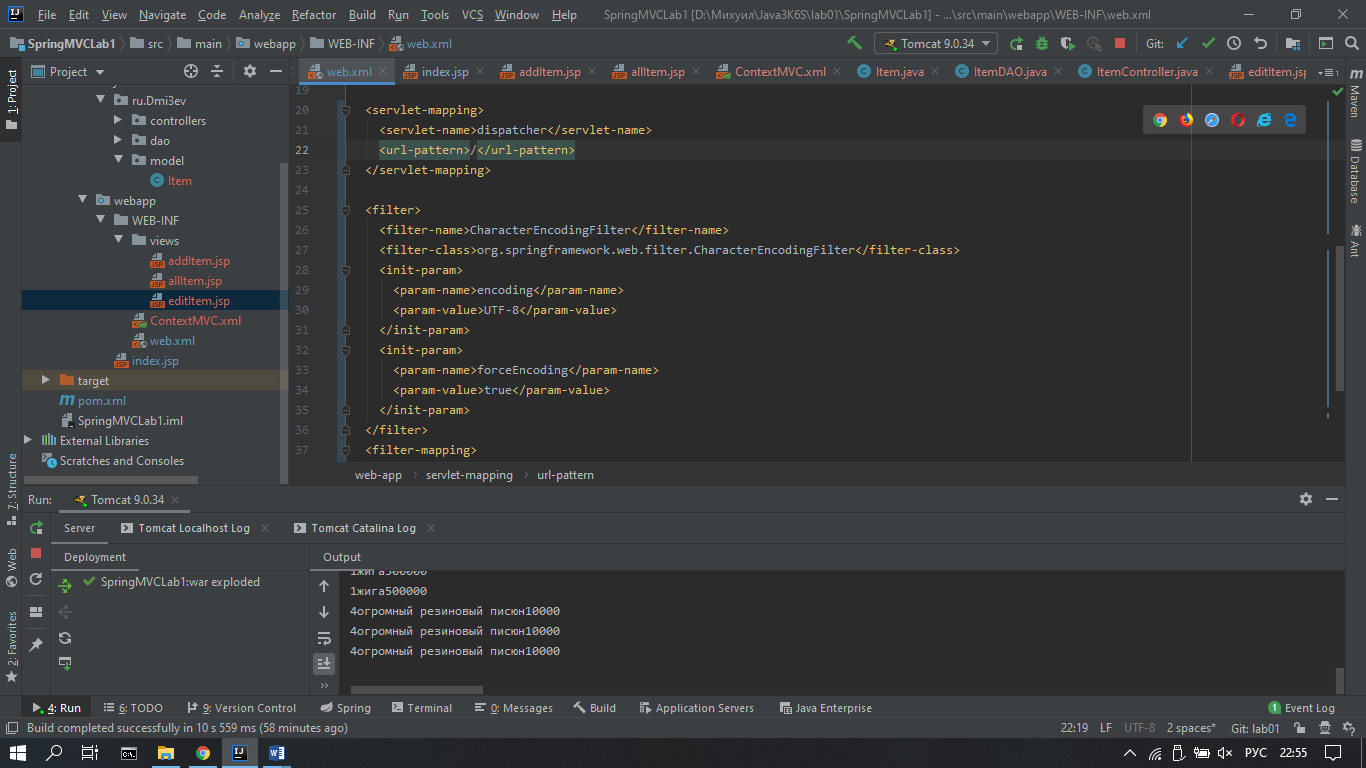


В web.xml создаю Dispatcher Servlet, который перенаправит запрос с сервера в Spring MVC приложение.

Название dispatcher. Класс, который обрабатывает запросы создаётся сам и находится в библиотеке org.springframework.web.servlet.DispatcherServlet. Также при создании нужно указать файл, где находится Spring конфигурация /WEB-INF/spring.xml.

Параметр <load-on-startup> для загрузки Dispatcher Servlet в первую очередь.

Затем в <servlet-mapping> указываем, что каждый запрос должен перенаправляться на Dispatcher Servlet. После Dispatcher Servlet отправит запрос в указанный контроллер.



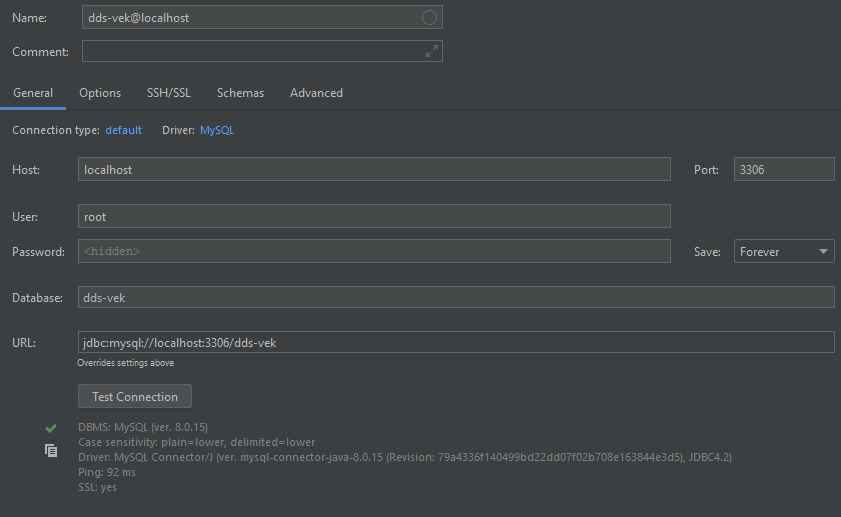
Создание файла spring.xml, в котором будет конфигурация Spring MVC приложения.

Component-scan отвечает за путь до компонента.

Далее описываем интерфейс ViewResolver, он будет определять какое представление необходимо использовать на основании полученного имени.  Описываем источник данных (ссылку на БД, логин и пароль) В конце настраиваем JdbcTemplate – класс, который будет управлять обработкой всех событий в БД.

<?xml version="1.0" encoding="UTF-8"?>  
<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"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-context.xsd">  
  
 <context:component-scan base-package="ru.Dmi3ev.controllers"/>  
  
 <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">  
 <property name="prefix" value="/WEB-INF/views/"/>  
 <property name="suffix" value=".jsp"/>  
 </bean>  
  
 <bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">  
 <property name="driverClassName" value="com.mysql.jdbc.Driver"></property>  
 <property name="url" value="jdbc:mysql://localhost:3306/dss\_lab-1?useSSL=false&amp;useUnicode=true&amp;useJDBCCompliantTimezoneShift=true&amp;useLegacyDatetimeCode=false&amp;serverTimezone=UTC"></property>  
 <property name="username" value="root"></property>  
 <property name="password" value="root"></property>  
 </bean>  
  
 <bean id="jt" class="org.springframework.jdbc.core.JdbcTemplate">  
 <property name="dataSource" ref="ds"></property>  
 </bean>  
  
 <bean id="ItemDao" class="ru.Dmi3ev.dao.ItemDAO">  
 <property name="template" ref="jt"></property>  
 </bean>  
  
  
</beans>

Подключение заранее подготовленной БД, таблица которой соответствует модели данных выбранной предметной области.



Создание модели данных

package ru.Dmi3ev.model;  
  
public class Item {  
 private int id;  
 private String name;  
 private int price;  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public int getPrice() {  
 return price;  
 }  
  
 public void setPrice(int price) {  
 this.price = price;  
 }  
}

ition;  
 }

Создание DAO класса для доступа к БД

@Repository  
public class ItemDAO {  
  
 JdbcTemplate template;  
 private static Logger *logger*= LoggerFactory.*getLogger*(ItemDAO.class);  
  
 public List<Item> getAllItem(){  
 *logger*.info("Вызов метода getAllItem");  
 try{  
 return template.query("select \* from item",new RowMapper<Item>(){  
  
 public Item mapRow(ResultSet rs, int row) throws SQLException {  
 Item item = new Item();  
 item.setId(rs.getInt("id"));  
 item.setName(rs.getString("name"));  
 item.setPrice(rs.getInt("price"));  
 return item;  
 }  
 });  
  
 }catch (Exception e){  
 *logger*.error("Ошибка при выполнении метода listAllTasks: ", e);  
 return null;  
 }  
 }  
  
 public Item getItemById(int id){  
 *logger*.info("Вызов метода getItemById");  
 String query="select \* from item where id=?";  
 try{  
 return template.queryForObject(query, new Object[]{id}, new BeanPropertyRowMapper<Item>(Item.class));  
 }catch (Exception e) {  
 *logger*.error("Ошибка при выполнении метода getItemById: ", e);  
 return null;  
 }  
 }  
  
 public int update(Item item){  
 *logger*.info("Вызов метода update");  
 String query="update item set name=?, price=? where id=?";  
 System.*out*.println(item.getId() + item.getName() + item.getPrice());  
 Object[] params = {item.getName(), item.getPrice(), item.getId() };  
 int[] types = {Types.*VARCHAR*, Types.*INTEGER*, Types.*INTEGER*};  
 try {  
 return template.update(query,params,types);  
 }catch (Exception e) {  
 *logger*.error("Ошибка при выполнении метода update: ", e);  
 return -1;  
 }  
 }  
  
 public int addItem(Item item){  
 *logger*.info("Вызов метода addItem");  
 String query="insert into item(name, price) values (?, ?)";  
 Object[] params = {item.getName(), item.getPrice()};  
 int[] types = {Types.*VARCHAR*, Types.*INTEGER*};  
 try {  
 return template.update(query,params,types);  
 }catch (Exception e) {  
 *logger*.error("Ошибка при выполнении метода insert: ", e);  
 return -1;  
 }  
 }  
  
 public int delete(int id){  
 *logger*.info("Вызов метода delete");  
 String query="delete from item where id=?";  
 Object[] params = {id};  
 int[] types = {Types.*INTEGER*};  
 try {  
 return template.update(query,params,types);  
 }catch (Exception e) {  
 *logger*.error("Ошибка при выполнении метода delete: ", e);  
 return -1;  
 }  
 }  
  
 public void setTemplate(JdbcTemplate template) {  
 this.template = template;  
 }

RowMapper – интерфейс для обработки выборки из БД. Этот класс обрабатывает отдельно каждую запись, полученную из БД, и возвращает уже готовый объект - модель данных.

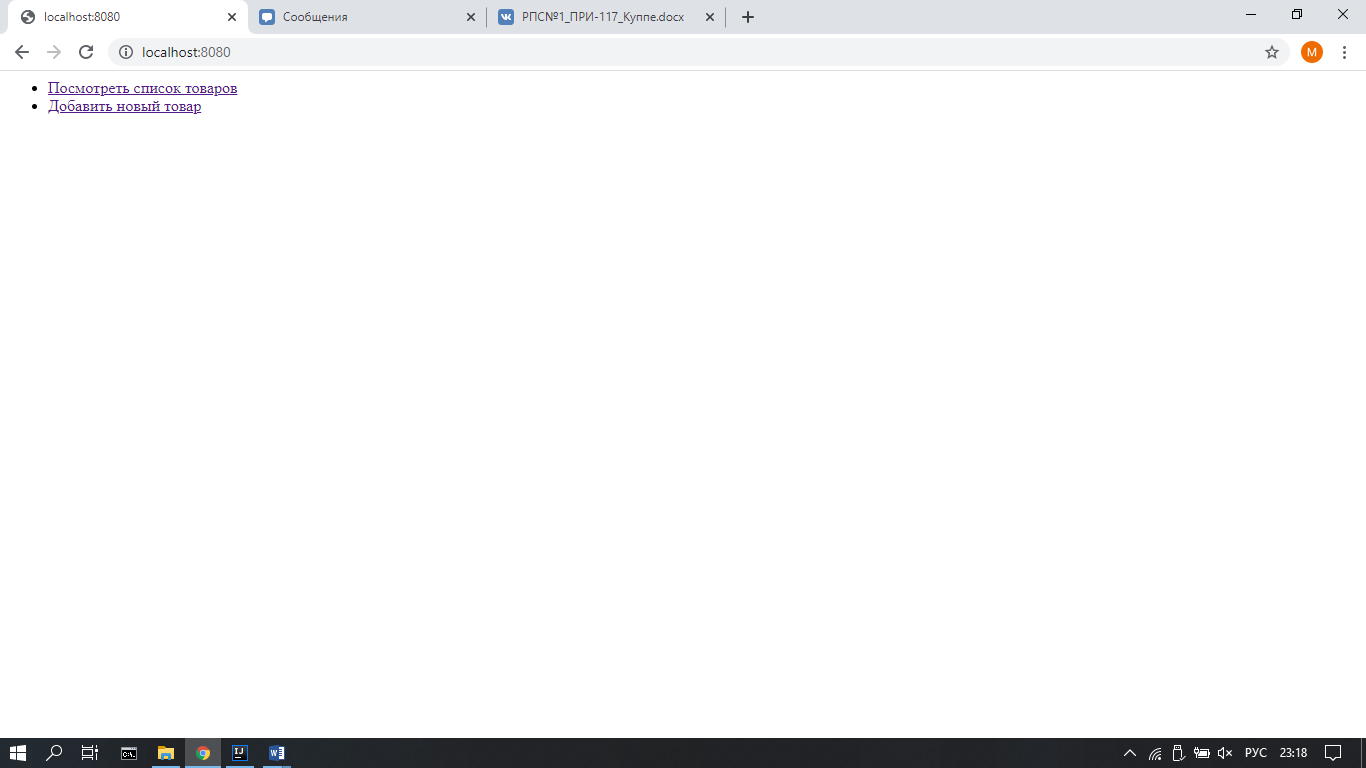
Создание контроллера

@Controller  
public class ItemController {  
  
 @Autowired  
 ItemDAO itemDAO;  
  
 @RequestMapping("index")  
 public String back(){  
 return "../../index";  
 }  
  
 @GetMapping("/addItem")  
 public String addItem(Model m){  
 m.addAttribute("command", new Item());  
 return "addItem";  
 }  
  
 @RequestMapping(value="/saveitem")  
 public String saveItem(@ModelAttribute("command") Item item){  
 int id = itemDAO.addItem(item);  
 if (id!=-1) return "redirect:/allItem";  
 else return "redirect:/error";  
 }  
  
 @GetMapping("allItem")  
 public String allItem(Model m){  
 List<Item> list = itemDAO.getAllItem();  
 m.addAttribute("list",list);  
 return "allItem";  
 }  
  
 @RequestMapping(value="/edit/{id}")  
 public String edit(@PathVariable int id, Model m){  
 Item item = itemDAO.getItemById(id);  
 m.addAttribute("command",item);  
 return "editItem";  
 }  
  
 @RequestMapping(value = "edit/apply")  
 public String editSave(@ModelAttribute("command") Item item){  
 System.*out*.println(item.getId() + item.getName() + item.getPrice());  
 int id = itemDAO.update(item);  
 System.*out*.println(item.getId() + item.getName() + item.getPrice());  
 if (id!=-1) return "redirect:../allItem";  
 else return "redirect:/error";  
 }  
  
 @RequestMapping(value = "/delete/{id}")  
 public String delete(@PathVariable int id){  
 int i = itemDAO.delete(id);  
 if (i!=-1) return "redirect:/allItem";  
 else return "redirect:/error";  
 }  
}

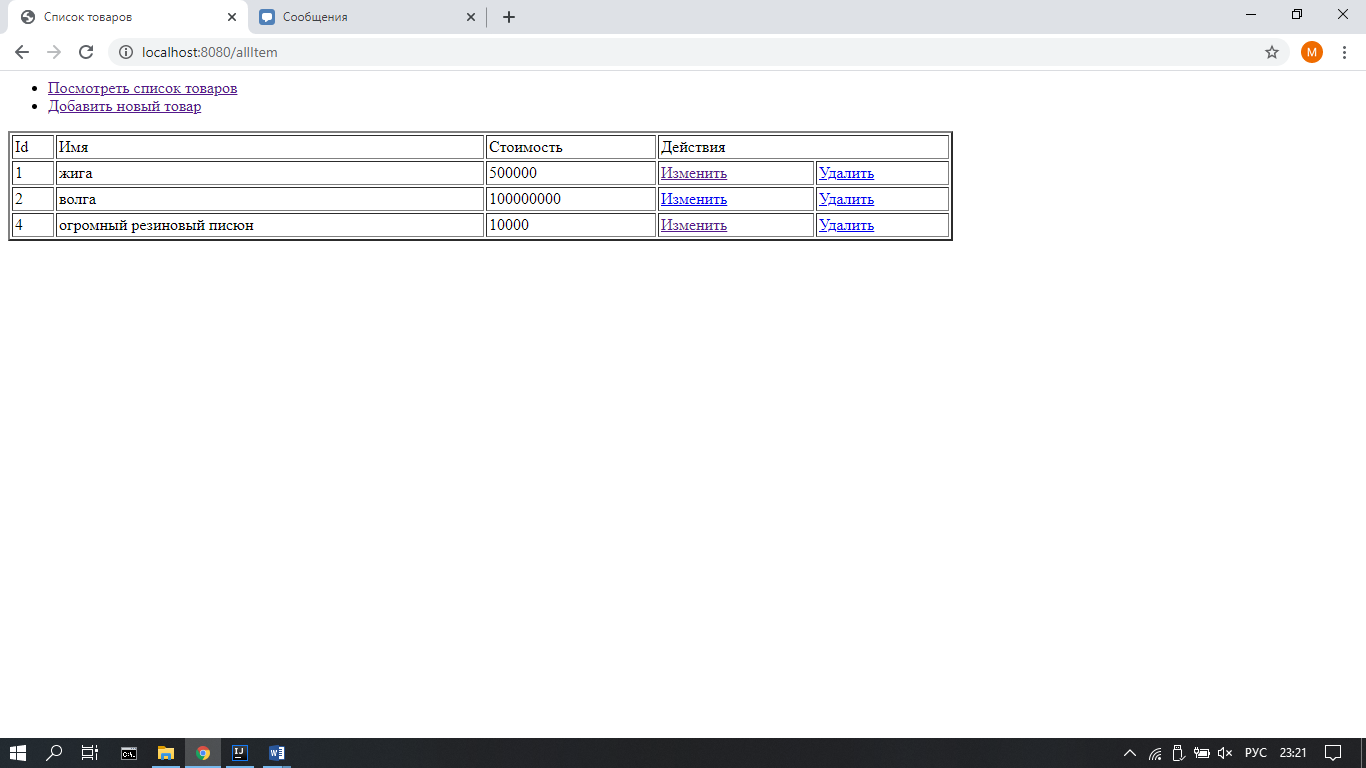
Используя эту аннотацию, не нужно заботиться о том, как лучше всего передать классу или bean'у экземпляр другого bean'a. Фреймворк Spring сам найдет нужный bean и подставит его значение в свойство, которое отмечено аннотацией @Autowired.

JSP – страницы

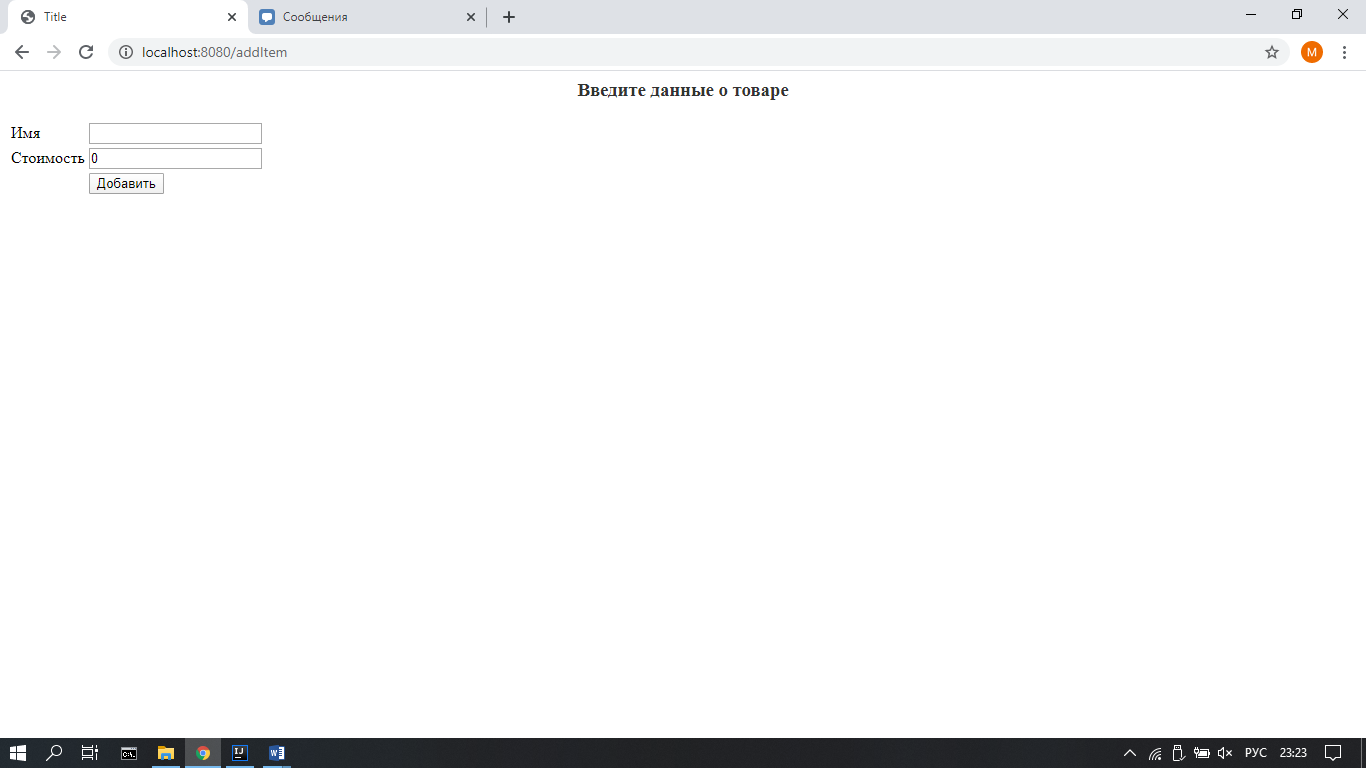
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<body>  
<ul>  
 <li><a href="allItem">Посмотреть список товаров</a>  
 <li><a href="addItem">Добавить новый товар</a>  
</ul>  
</body>  
</html>



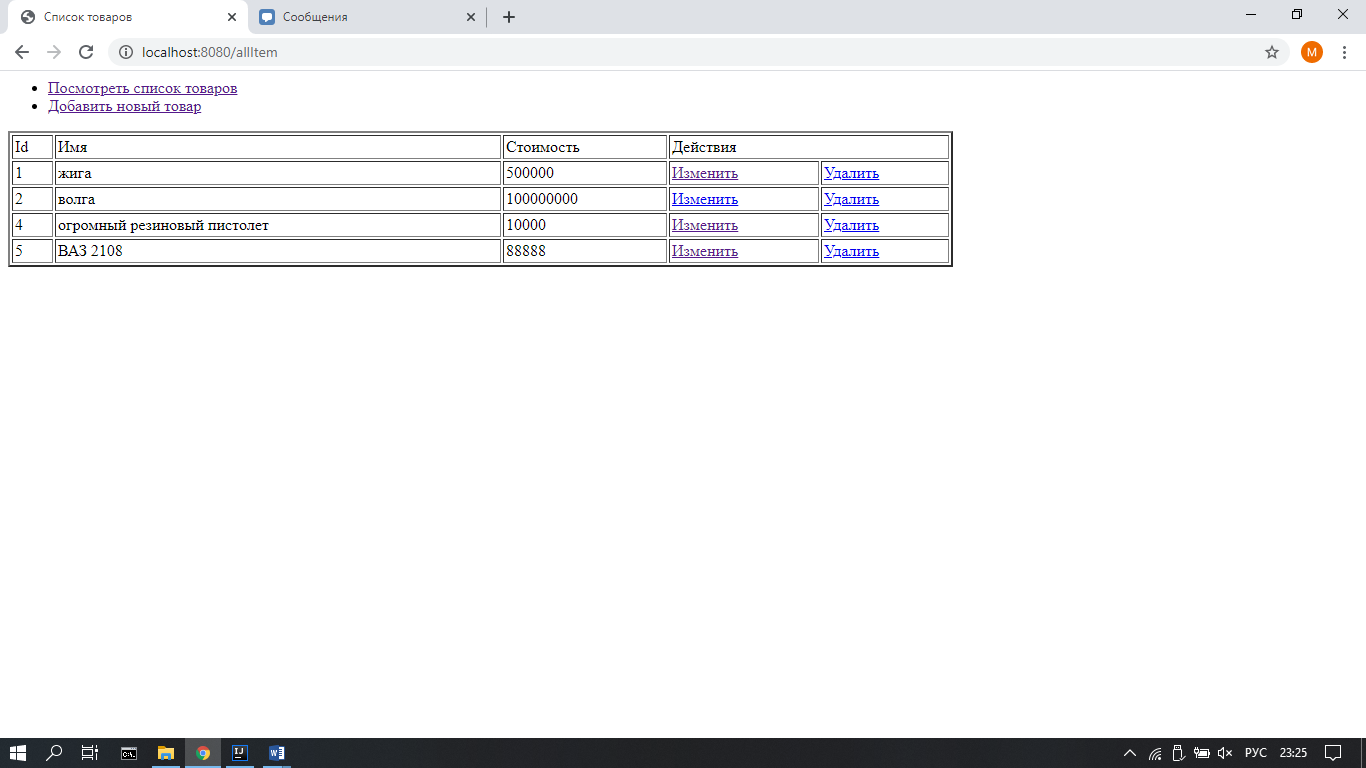
<%@ **taglib** uri="http://www.springframework.org/tags/form" prefix="form"%>  
<%@ **taglib** uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>  
<%@ **taglib** uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
  
<html>  
<head>  
 <title>Список товаров</title>  
</head>  
<body>  
<ul>  
 <li><a href="allItem">Посмотреть список товаров</a>  
 <li><a href="addItem">Добавить новый товар</a>  
</ul>  
<**c:choose**>  
 <**c:when** test="${fn:length(list) gt 0}">  
 <table border="2" width="70%" cellpadding="2">  
 <tr><td>Id</td><td>Имя</td><td>Стоимость</td><td colspan="2">Действия</td></tr>  
 <**c:forEach** var="item" items="${list}">  
 <tr>  
 <td>${item.id}</td>  
 <td>${item.name}</td>  
 <td>${item.price}</td>  
 <td><a href="edit/${item.id}">Изменить</a></td>  
 <td><a href="delete/${item.id}">Удалить</a></td>  
 </tr>  
 </**c:forEach**>  
 </table>  
 </**c:when**>  
 <**c:otherwise**>  
 <center><h3>Список пуст, добавте товар</h3></center>  
 </**c:otherwise**></**c:choose**>  
</body>  
</html>



<%--  
 Created by IntelliJ IDEA.  
 User: Михаил  
 Date: 07.05.2020  
 Time: 20:57  
 To change this template use File | Settings | File Templates.  
--%>  
<%@ **taglib** prefix="form" uri="http://www.springframework.org/tags/form" %>  
<%@ **taglib** uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>Title</title>  
</head>  
<body>  
<center><h3 style="color: #333;">Введите данные о товаре</h3></center>  
<**form:form** method="post" action="saveitem">  
 <table >  
 <tr>  
 <td>Имя</td>  
 <td><**form:input** path="name" /></td>  
 </tr>  
 <tr>  
 <td>Стоимость</td>  
 <td><**form:input** path="price" /></td>  
 </tr>  
 <tr>  
 <td> </td>  
 <td><input type="submit" value="Добавить" /></td>  
 </tr>  
 </table>  
</**form:form**>  
</body>  
</html>



<%--  
 Created by IntelliJ IDEA.  
 User: Роман  
 Date: 07.05.2020  
 Time: 21:14  
 To change this template use File | Settings | File Templates.  
--%>  
<%@ **taglib** prefix="form" uri="http://www.springframework.org/tags/form" %>  
<%@ **taglib** uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>Изменить товар</title>  
</head>  
<body>  
<**form:form** method="post" action="apply">  
 <table>  
 <**form:hidden** path="id"/>  
 <tr><td>Имя</td>  
 <td><**form:input** path="name"/></td>  
 </tr>  
 <tr>  
 <td>Стоимость</td>  
 <td><**form:input** path="price" /></td>  
 </tr>  
 <tr>  
 <td> </td>  
 <td><input type="submit" value="Принять изменения" /></td>  
 </tr>  
 </table>  
</**form:form**>  
  
</body>  
</html>



Вывод

В ходе лабораторной работы были изучены принципы построения веб приложений с использованием шаблона MVC и на базе принципов REST, получены практические навыки реализации MVC-приложения на платформе Spring MVC.