**ЛАБОРАТОРНАЯ РАБОТА №8**

**РАЗРАБОТКА ЭФФЕКТИВНЫХ ВЕБ-ПРИЛОЖЕНИЙ (SPRING MVC)**

Цель работы: разработка веб-приложений с реализацией шаблона MVC.

Задание:

1. Создать дополнительный контроллер и представление (JSP) с отображением полей объекта класса из лабораторной работы №7.

2. Прокомментировать код программы.

На основании классов из лабораторной работы №7 были разработаны контроллер RateController и JSP-файл rate.jsp.

package com;  
  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.ModelMap;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestMethod;  
  
import java.util.ArrayList;  
import java.util.List;  
  
@Controller  
@RequestMapping("/rate")  
public class RateController {  
  
 // Для примера создаем статический список тарифов.  
 private static final List<Rate> *rates* = new ArrayList<>();  
  
 static {  
 // Создаем компании  
 Company company1 = new Company("Life");  
 Company company2 = new Company("МТС");  
  
 // Создаем тарифы и привязываем их к компаниям  
 Rate rate1 = new Rate("Линейный тариф", company1);  
 Rate rate2 = new Rate("Все за копейку", company2);  
  
 // Добавляем тарифы к компаниям  
 company1.addRate(rate1);  
 company2.addRate(rate2);  
  
 // Создаем абонентов и привязываем их к тарифам  
 Subscriber subscriber1 = new Subscriber("Иванов Иван", "+375253456789", rate1);  
 Subscriber subscriber2 = new Subscriber("Петров Петр", "+375257654321", rate1);  
 Subscriber subscriber3 = new Subscriber("Сидоров Сидор", "+375335555555", rate2);  
  
 // Добавляем абонентов к тарифам  
 rate1.addSubscriber(subscriber1);  
 rate1.addSubscriber(subscriber2);  
 rate2.addSubscriber(subscriber3);  
  
 // Добавляем тарифы в список  
 *rates*.add(rate1);  
 *rates*.add(rate2);  
 }  
  
 // Обработчик HTTP GET-запросов на URL "/rate"  
 @RequestMapping(method = RequestMethod.*GET*)  
 public String listRates(ModelMap model) {  
 // Передаем список тарифов в модель  
 model.addAttribute("rates", *rates*);  
 // Возвращаем имя представления, которое будет отображать данные  
 return "rate";  
 }  
}

<%@ page contentType="text/html;charset=UTF-8" language="java" %>  
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  
<html>  
<head>  
 <title>Rate List</title>  
</head>  
<body>  
<h2>Тарифы и абоненты:</h2>  
<%-- Перебираем тарифы и отображаем их детали вместе с абонентами --%>  
<c:forEach var="rate" items="${rates}">  
 <h3>Тариф: ${rate.name} (Компания: ${rate.company.name})</h3>  
 <p>Абоненты:</p>  
 <ul>  
 <%-- Перебираем абонентов, привязанных к текущему тарифу, и отображаем их данные --%>  
 <c:forEach var="subscriber" items="${rate.subscribers}">  
 <li>${subscriber.data} - ${subscriber.number} (Balance: ${subscriber.balance})</li>  
 </c:forEach>  
 </ul>  
</c:forEach>  
</body>  
</html>

Работа приложения, запускаемого на localhost:8080, представлена на рисунке 1.

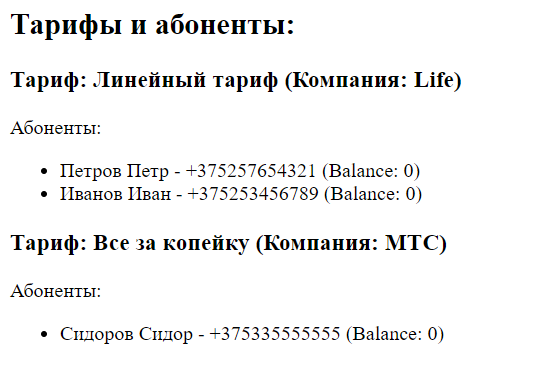


Рисунок 1 – Работа приложения в браузере

Вывод: В рамках выполнения лабораторной работы была изучена структура проектов на Spring, получены знания о паттерне MVC.