ГУАП КАФЕДРА № 43

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ ПРЕПОДАВАТЕЛЬ

|  |  |  |
| --- | --- | --- |
| ассистент |  | П.А.Степанов |
| должность, уч. степень, звание | подпись, дата | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2 |
| «Разработка простого AJAX приложения Spring» |
| по дисциплине: ТЕХНОЛОГИЯ РАЗРАБОТКИ СЕРВЕРНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ |

РАБОТУ ВЫПОЛНИЛА

|  |  |  |
| --- | --- | --- |
| СТУДЕНТКА ГР. | 4232 | Е.А.Уткина |
|  | подпись, дата | инициалы, фамилия |

Санкт-Петербург

2024

1. Цель работы:

Разработать простое AJAX приложение Spring.

1. Задание:

1. Создайте одну или несколько форм и контроллеров для страниц приложения. Подключите библиотеку angularJS (желающие могут воспользоваться JQuery или иным аналогом).

2. Реализуйте задачи из лабораторной работы 1 средствами AJAX и REST-сервисов, разработанных в предыдущей лабораторной работе.

Вариант 14: Сдача недвижимости в аренду.

1. Листинг программы:

ControllersConfig.Java:

package info.config;  
  
import org.springframework.context.annotation.Configuration;  
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;  
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  
  
@Configuration  
public class ControllersConfig implements WebMvcConfigurer {  
  
 @Override  
 public void addViewControllers(ViewControllerRegistry registry) {  
 registry.addViewController("/apartments").setViewName("apartments");  
 registry.addViewController("/").setViewName("apartments");  
 }  
}

SecurityConfiguration.Java:

package info.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.web.SecurityFilterChain;  
  
@Configuration  
public class SecurityConfiguration {  
  
 @Bean  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http  
 .authorizeHttpRequests((authz) -> authz  
 .requestMatchers("/\*\*", "/css/\*\*", "/js/\*\*", "/images/\*\*", "/public/rest/\*\*").permitAll()  
 );  
  
 http.exceptionHandling((exception) -> exception.accessDeniedPage("/403"));  
 http.csrf().disable();  
  
 return http.build();  
 }  
}

SpringDocConfig.java:

package info.config;  
  
import io.swagger.v3.oas.models.OpenAPI;  
import io.swagger.v3.oas.models.info.Info;  
import org.springdoc.core.models.GroupedOpenApi;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
  
@Configuration  
  
public class SpringDocConfig {  
  
 @Bean  
 public OpenAPI apiInfo() {  
 return new OpenAPI().info(new Info().title("Lab 2 Rest API").version("1.0.0"));  
 }  
  
 @Bean  
 public GroupedOpenApi httpApi() {  
 return GroupedOpenApi.*builder*()  
 .group("http")  
 .pathsToMatch("/\*\*")  
 .build();  
 }  
}

ApartmentsRepository.java:

package info.db.dao;  
  
import info.db.model.ApartmentsPE;  
  
import java.util.List;  
import java.util.Optional;  
import org.springframework.data.repository.CrudRepository;  
  
public interface ApartmentsRepository extends CrudRepository<ApartmentsPE, Integer> {  
  
 Optional<ApartmentsPE> findByID(Integer ID);  
  
 List<ApartmentsPE> findAll();  
}

ApartmentsPE.java:

package info.db.model;  
  
import jakarta.persistence.Column;  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
import jakarta.persistence.Table;  
import java.io.Serializable;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
@Entity  
@Table(name = "APARTMENTS")  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
public class ApartmentsPE implements Serializable {  
  
 private static final long *serialVersionUID* = 1L;  
  
 public ApartmentsPE(String city, String address, Integer price) {  
 this.city = city;  
 this.address = address;  
 this.price = price;  
 }  
  
 @Id  
 @Column(name = "ID")  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Integer ID;  
  
 @Column(name = "CITY")  
 private String city;  
  
 @Column(name = "ADDRESS")  
 private String address;  
  
 @Column(name = "PRICE")  
 private Integer price;  
}

ApartmentsDTO.java:

package info.rest.model;  
  
import io.swagger.v3.oas.annotations.media.Schema;  
import lombok.Data;  
  
@Data  
@Schema(description = "Квартира")  
public class ApartmentsDTO {  
  
 @Schema(description = "Идентификатор квартиры")  
 private Integer ID;  
 @Schema(description = "Город")  
 private String city;  
 @Schema(description = "Адрес")  
 private String address;  
 @Schema(description = "Цена")  
 private Integer price;  
}

ApartmentsRestController.java:

package info.rest;  
  
import info.rest.model.ApartmentsDTO;  
import info.service.ApartmentsService;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestMethod;  
import org.springframework.web.bind.annotation.RestController;  
  
import io.swagger.v3.oas.annotations.Operation;  
import io.swagger.v3.oas.annotations.Parameter;  
import io.swagger.v3.oas.annotations.responses.ApiResponse;  
import java.util.List;  
import lombok.RequiredArgsConstructor;  
import lombok.extern.slf4j.Slf4j;  
  
@Slf4j  
@RestController  
@RequestMapping("/public/rest/apartments")  
@RequiredArgsConstructor  
public class ApartmentsRestController {  
  
 private final ApartmentsService apartmentsService;  
  
 @RequestMapping(value = "", method = RequestMethod.*GET*)  
  
 @Operation(summary = "Получить перечень квартир",  
 description = "Получить перечень квартир - расширенное описание",  
 responses = {  
 @ApiResponse(responseCode = "200",  
 description = "Успешное выполнение"),  
 @ApiResponse(responseCode = "404",  
 description = "Ресурс не найден")  
 })  
 public ResponseEntity<List<ApartmentsDTO>> browse() {  
 return ResponseEntity.*ok*(apartmentsService.listAll());  
 }  
  
 @Operation(summary = "Удаление квартиры",  
 description = "Удаление квартиры по ID",  
 responses = {  
 @ApiResponse(responseCode = "200",  
 description = "Успешное выполнение"),  
 @ApiResponse(responseCode = "404",  
 description = "Ресурс не найден")  
 })  
 @RequestMapping(value = "/{ID}", method = RequestMethod.*DELETE*)  
 public void delete(@PathVariable("ID")  
 @Parameter(description = "Идентификатор квартиры") Integer ID) {  
 apartmentsService.delete(ID);  
 }  
  
 @Operation(summary = "Создать новую квартиру",  
 description = "Создать новую квартиру - расширенное описание",  
 responses = {  
 @ApiResponse(responseCode = "200",  
 description = "Успешное выполнение"),  
 @ApiResponse(responseCode = "404",  
 description = "Ресурс не найден")  
 })  
 @RequestMapping(value = "/{city}/{address}/{price}", method = RequestMethod.*POST*)  
 public ResponseEntity<ApartmentsDTO> add(  
 @PathVariable("city")  
 @Parameter(description = "Город") String city,  
 @PathVariable("address")  
 @Parameter(description = "Адрес") String address,  
 @PathVariable("price")  
 @Parameter(description = "Цена, руб/мес") Integer price) {  
 return ResponseEntity.*ok*(apartmentsService.add(city, address, price));  
 }  
  
 @Operation(summary = "Поиск квартиры по ID",  
 description = "Поиск квартиры по ID - расширенное описание",  
 responses = {  
 @ApiResponse(responseCode = "200",  
 description = "Успешное выполнение"),  
 @ApiResponse(responseCode = "404",  
 description = "Ресурс не найден")  
 })  
 @RequestMapping(value = "/{ID}", method = RequestMethod.*GET*)  
 public ResponseEntity<ApartmentsDTO> findByNumber(@PathVariable("ID")  
 @Parameter(description = "Идентификатор квартиры") Integer ID) {  
 return ResponseEntity.*ok*(apartmentsService.findByID(ID));  
 }  
}

ApartmentsService.java:

package info.service;  
  
import info.rest.model.ApartmentsDTO;  
  
import java.util.List;  
  
public interface ApartmentsService {  
  
 List<ApartmentsDTO> listAll();  
  
 void delete(Integer ID);  
  
 ApartmentsDTO add(String city, String address, Integer price);  
  
 ApartmentsDTO findByID(Integer ID);  
  
}

ApartmentsServiceImpl.java:

package info.service;  
  
import com.fasterxml.jackson.databind.ObjectMapper;  
import info.db.dao.ApartmentsRepository;  
import info.db.model.ApartmentsPE;  
import info.rest.model.ApartmentsDTO;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.stream.Collectors;  
import lombok.RequiredArgsConstructor;  
import lombok.extern.slf4j.Slf4j;  
  
@Slf4j  
@Service  
@RequiredArgsConstructor  
public class ApartmentsServiceImpl implements ApartmentsService {  
  
 private final ApartmentsRepository apartmentsRepository;  
  
 private final ObjectMapper objectMapper;  
  
 @Override  
 public List<ApartmentsDTO> listAll() {  
 return apartmentsRepository.findAll().stream()  
 .map(apartmentsPE -> objectMapper.convertValue(apartmentsPE, ApartmentsDTO.class))  
 .collect(Collectors.*toList*());  
  
 }  
  
 @Override  
 public void delete(Integer ID) {  
 apartmentsRepository.deleteById(ID);  
 }  
  
 @Override  
 public ApartmentsDTO add(String city, String address, Integer price) {  
 return objectMapper.convertValue(apartmentsRepository.save(new ApartmentsPE(city, address, price)), ApartmentsDTO.class);  
 }  
  
 @Override  
 public ApartmentsDTO findByID(Integer ID) {  
 var ApartmentsPE = apartmentsRepository.findByID(ID);  
 return ApartmentsPE.map(apartmentsPE -> objectMapper.convertValue(apartmentsPE, ApartmentsDTO.class)).orElse(null);  
 }  
}

pom.xml:

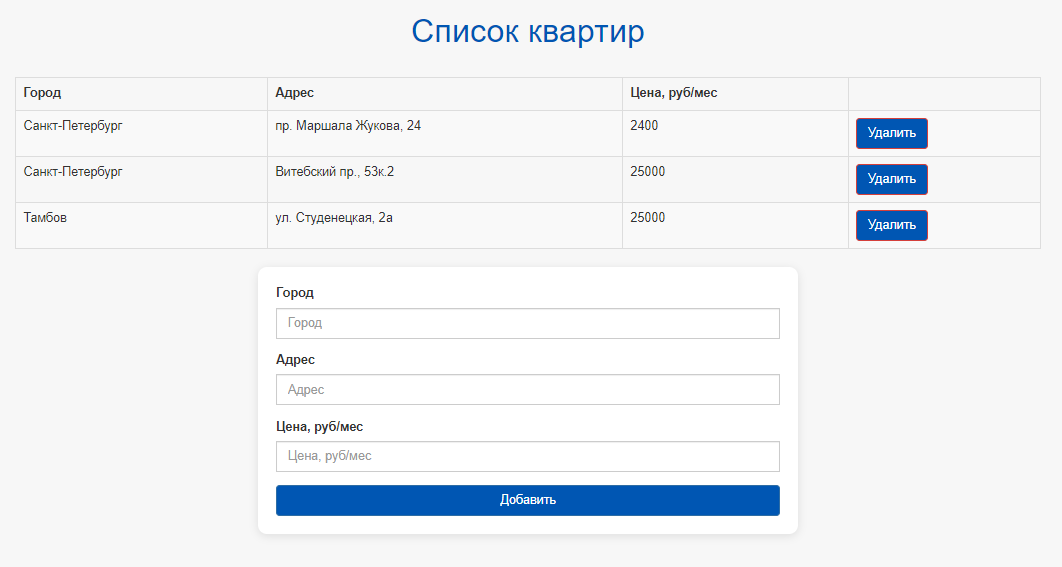
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <groupId>info.stepanoff.trsis.samples.jsp</groupId>  
 <artifactId>Lab3</artifactId>  
 <packaging>jar</packaging>  
 <version>1.0-SNAPSHOT</version>  
 <name>AJAX and REST - based webapp</name>  
 <url>https://maven.apache.org</url>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.3.0</version>  
 </parent>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-thymeleaf</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-security</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springdoc</groupId>  
 <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>  
 <version>2.6.0</version>  
 </dependency>  
  
 <dependency>  
 <groupId>com.github.joschi.jackson</groupId>  
 <artifactId>jackson-datatype-threetenbp</artifactId>  
 <version>2.15.2</version>  
 </dependency>  
  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <scope>test</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <type>jar</type>  
 </dependency>  
  
 <dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <version>2.3.230</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.liquibase</groupId>  
 <artifactId>liquibase-core</artifactId>  
 <version>4.29.1</version>  
 </dependency>  
  
  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-api</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-log4j12</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
 <build>  
 <finalName>lab3</finalName>  
  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 <version>3.3.1</version>  
 <executions>  
 <execution>  
 <goals>  
 <goal>repackage</goal>  
 </goals>  
 </execution>  
 </executions>  
 </plugin>  
 <plugin>   
 <groupId>org.liquibase</groupId>   
 <artifactId>liquibase-maven-plugin</artifactId>   
 <version>4.29.1</version>   
 <configuration>   
 <propertyFile>liquibase.properties</propertyFile>   
 </configuration>   
 <dependencies>   
 <dependency>   
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <version>2.3.230</version>  
 </dependency>   
 </dependencies>   
 </plugin>   
 </plugins>  
 </build>  
 <properties>  
 <maven.compiler.source>17</maven.compiler.source>  
 <maven.compiler.target>17</maven.compiler.target>  
 </properties>  
</project>

web.xml:

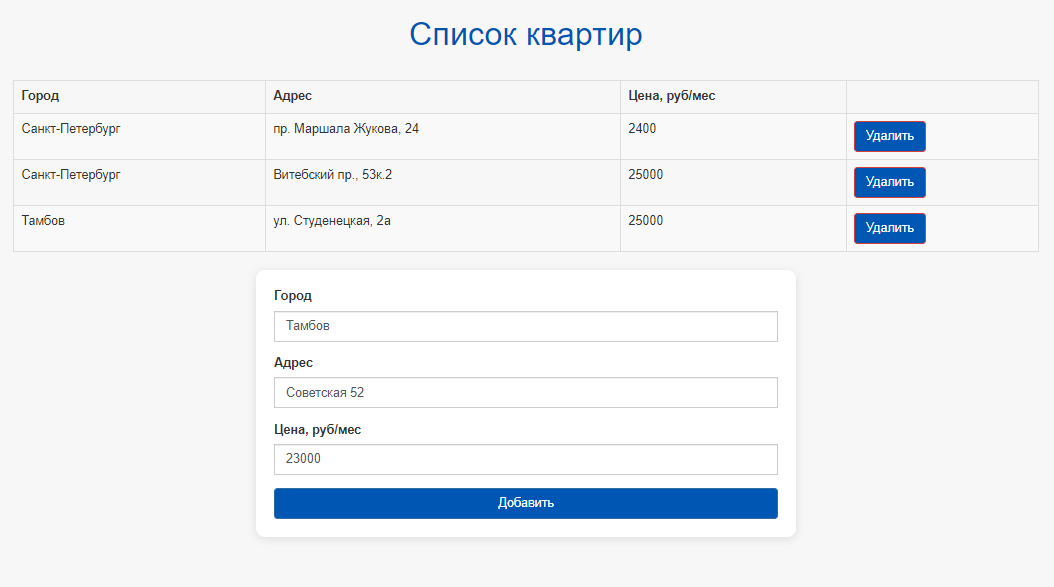
<!DOCTYPE web-app PUBLIC  
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"  
 "http://java.sun.com/dtd/web-app\_2\_3.dtd" >  
  
<web-app>  
 <display-name>Archetype Created Web Application</display-name>  
</web-app>

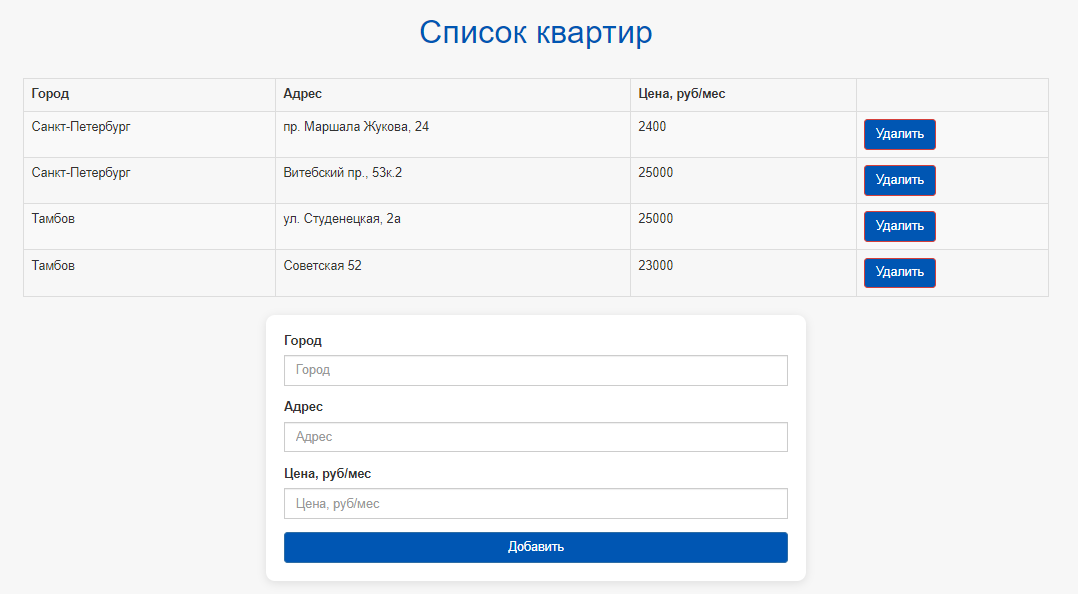
1. Пример выполнения программы:

Список квартир:

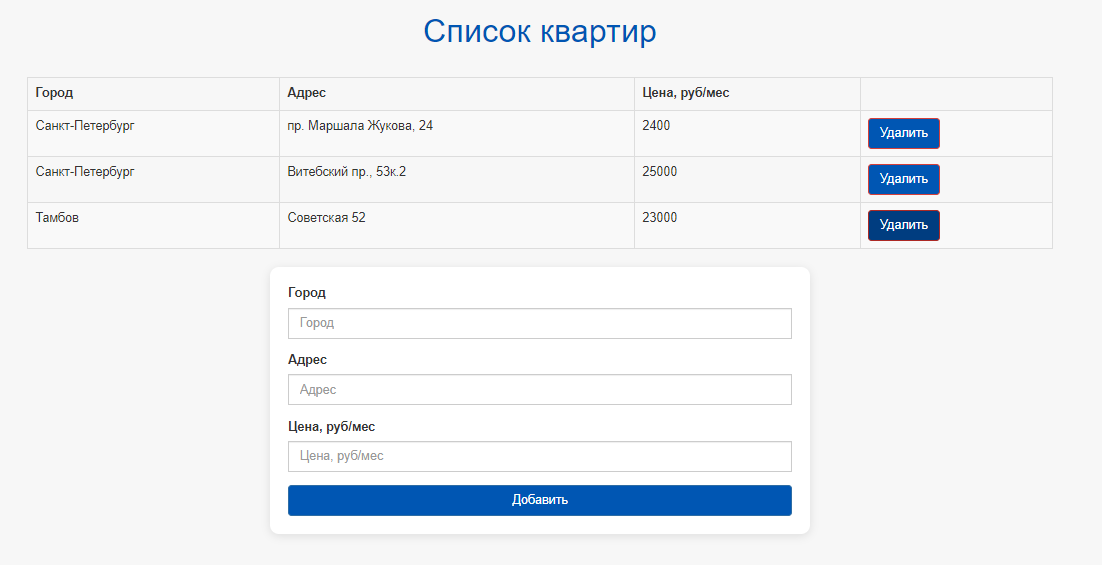


Добавление квартиры:





Удаление квартиры:



1. Вывод:

Разработано простое AJAX приложение Spring.