ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| кандидат технических наук |  |  |  | П. А. Степанов |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4 |
| Разработка формы логина |
| по курсу: Технология разработки серверных информационных систем |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4134к |  |  |  | Столяров Н.С. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2023

**Текст и вариант задания:**

19 вариант

База данных с рецептами блюд

**Описание разрабатываемого продукта:**

В данной программе создан сервис по работе с рецептами блюд (вывод, добавление, удаление и поиск)

**Текст основных фрагментов кода :**

package info.stepanoff.trsis.samples.config;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.builders.WebSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@Configuration

public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

@Autowired

private UserDetailsService userDetailsService;

@Bean

@Override

public AuthenticationManager authenticationManagerBean() throws Exception {

return super.authenticationManagerBean();

}

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.inMemoryAuthentication()

.withUser("100LAR").password("{noop}hello")

.authorities("ROLE\_USER");

}

@Override

public void configure(WebSecurity web) throws Exception {

web.ignoring().antMatchers("/resources/\*\*");

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.authorizeRequests().antMatchers("/\*\*", "/css/\*\*", "/js/\*\*", "/img/\*\*", "/public/rest/\*\*")

.permitAll().and().formLogin().loginPage("/login").failureUrl("/login?error").usernameParameter("login").passwordParameter("pass")

.permitAll().and().logout().logoutUrl("/logout").logoutSuccessUrl("/login?logout").permitAll().and()

.exceptionHandling().accessDeniedPage("/forbidden");

}

@Autowired

public void configAuthentication(AuthenticationManagerBuilder auth) throws Exception {

auth.userDetailsService(userDetailsService).passwordEncoder(new BCryptPasswordEncoder());

}

}

package info.stepanoff.trsis.samples.rest;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import io.swagger.annotations.ApiResponse;

import io.swagger.annotations.ApiResponses;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import info.stepanoff.trsis.samples.service.ReceptService;

import lombok.extern.slf4j.Slf4j;

import java.security.Principal;

@Slf4j

@RestController

@RequestMapping("/public/rest/Recept")

@Api(value = "ReceptAPI", description = "API for accessing Recept")

public class ReceptRestService {

@Autowired

private ReceptService receptService;

@RequestMapping(value = "", method = RequestMethod.GET)

@ApiOperation(value = "View a list of all available recept", response = Iterable.class)

@ApiResponses(value = {

@ApiResponse(code = 200, message = "Successfully retrieved list"),

@ApiResponse(code = 401, message = "You are not authorized to view the resource"),

@ApiResponse(code = 403, message = "Accessing the resource you were trying to reach is forbidden"),

@ApiResponse(code = 404, message = "The resource you were trying to reach is not found")

}

)

public ResponseEntity<Object> browse() {

return ResponseEntity.ok(receptService.listAll());

}

@RequestMapping(value = "/{id}", method = RequestMethod.DELETE)

@ApiOperation(value = "Remove a recept by ID", response = Iterable.class)

@ApiResponses(value = {

@ApiResponse(code = 200, message = "Successfully removed"),

@ApiResponse(code = 401, message = "You are not authorized to view the resource"),

@ApiResponse(code = 403, message = "Accessing the resource you were trying to reach is forbidden"),

@ApiResponse(code = 404, message = "The resource you were trying to reach is not found")

}

)

public void delete(@PathVariable("id") Integer id, Principal principal) {

receptService.delete(id);

}

@RequestMapping(value = "/{\_quantity}/{name}/{\_recept}", method = RequestMethod.POST)

public ResponseEntity<Object> add(

@PathVariable("\_quantity") int \_quantity,

@PathVariable("name") String name,

@PathVariable("\_recept") String \_recept) {

return ResponseEntity.ok(receptService.add(\_quantity, name, \_recept));

}

//int \_quantity, String \_name, String \_recept

@ApiOperation(value = "Find a recept by name", response = Iterable.class)

@ApiResponses(value = {

@ApiResponse(code = 200, message = "Successfully found"),

@ApiResponse(code = 401, message = "You are not authorized to view the resource"),

@ApiResponse(code = 403, message = "Accessing the resource you were trying to reach is forbidden"),

@ApiResponse(code = 404, message = "The resource you were trying to reach is not found")

}

)

@RequestMapping(value = "/{name}", method = RequestMethod.GET)

public ResponseEntity<Object> findBy\_name(@PathVariable("name") String name) {

return ResponseEntity.ok(receptService.findBy\_name(name));

}

}

package info.stepanoff.trsis.samples.db.model;

import lombok.Data;

import javax.persistence.\*;

@Entity

@Table(name = "User")

@Data

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id", nullable = false, updatable = false)

private Long id;

@Column(nullable = false)

private String login;

@Column(name="pass\_hash", nullable = false)

private String passHash;

}