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 дисциплине: Технология разработки серверных информационных систем |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4134к |  |  |  | Костяков Н.А. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург

2023

Задание на лабораторную работу.

# 1 Переведите вебсервер на использование SSL 2 Реализуйте form-based authentication 3 Защитите ресурсы, требующие запись и введите аудит.

# Вариант 4

Расписание поездов, самолетов, кораблей

# Листинг

<!DOCTYPE html>  
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org"  
 xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity3">  
<head>  
  
 <meta charset="utf-8"></meta>  
 <meta http-equiv="X-UA-Compatible" content="IE=edge"></meta>  
 <meta name="viewport" content="width=device-width, initial-scale=1"></meta>  
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8"></meta>  
  
 <title>TimeTable</title>  
  
 <link href="../static/css/bootstrap.min.css" th:href="@{/css/bootstrap.min.css}" rel="stylesheet"></link>  
  
  
 <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.3/angular.min.js"></script>  
 <script>  
 var *app* = angular.module('time', []);  
 *app*.controller("TtApiController", function( $scope, $http){  
 $scope.getTt = function (){  
 $http.get('/tt').success(function (data, status, headers, config){  
 $scope.ttList = data;  
  
 *console*.log($scope.ttList)  
 }).error(function (data, status, headers, config){  
 if (data.message === 'Time is out'){  
 $scope.finishByTimeout();  
 *console*.log('Errorrrr')  
 }  
 });  
 };  
  
  
 $scope.delTime = function (id){  
 $http.delete('/tt/'+id).success(function (data, status, headers, config){  
 for (var i = 0; i <$scope.ttList.length;i++){  
 var j = $scope.ttList[i];  
 if (j.id===id){  
 $scope.ttList.splice(i, 1);  
 break;  
 }  
 }  
 }).error(function (data, status, headers, config){  
 if (data.message === 'Time is out'){  
 *console*.log('Errorrrr')  
 }  
 });  
 };  
  
 $scope.createTime = function (){  
 $http.post('/tt', JSON.stringify({  
 'name': $scope.name,  
 'type': $scope.type  
 })).success(function (data, status, headers, config){  
 console.log("successsssssssssss");  
 console.log(data);  
 $scope.ttList.splice(0,0,data);  
  
 }).error(function (data, status, headers, config){  
 if (data.message === 'Time is out'){  
 *console*.log('Errorrrr')  
 }  
 });  
 };  
  
 });  
 </script>  
</head>  
<body ng-app="time" ng-controller="TtApiController">  
  
 <div sec:authorize="isAnonymous()" >  
 <h1><a href="/login">Login</a></h1>  
 </div>  
  
 <div sec:authorize="isAuthenticated()">  
 <form name="logoutForm" th:action="@{/logout}" method="post">  
 <input type="submit" class="btn btn-primary" value="Sign Out"/>  
 </form>  
 </div>  
  
 <h1>Расписание</h1>  
 <div ng-controller="getTt" ng-show="ttList.length>0">  
 <table id ="Tttable">  
 <thead>  
 <tr>  
 <th>Номер</th>  
 <th>название</th>  
 <th>Тип</th>  
 <th>Действие</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr ng-repeat="Time in ttList" id ="{{Time.id}}">  
 <td>{{Time.time\_id}}</td>  
 <td>{{Time.name}}</td>  
 <td>{{Time.type}}</td>  
 <td><input type="button" value="remove" ng-click="delTime(Time.time\_id-1)"></td>  
  
 </tr>  
 </tbody>  
 </table>  
  
 </div>  
 <div>  
 <form role="form" ng-submit="createTime()">  
 <div>  
 <input placeholder="Название роута" type="text" name ="name" ng-model="name" />  
 <input placeholder="Тип маршрута (Автобус, корабль, самолет)" type="text" name="type" ng-model="type" />  
 </div>  
 <div><input type="submit" value="Отпарвить"></div>  
 </form>  
 </div>  
  
  
  
</body>  
</html>

package io.swagger.configuration;  
  
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.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("guest").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").and().csrf().disable();  
  
 }  
  
 @Autowired  
 public void configAuthentication(AuthenticationManagerBuilder auth) throws Exception {  
 auth.userDetailsService(userDetailsService).passwordEncoder(new BCryptPasswordEncoder());  
 }  
  
}

package io.swagger.api;  
  
import io.swagger.model.Time;  
import io.swagger.model.TimeTable;  
import com.fasterxml.jackson.databind.ObjectMapper;  
import io.swagger.v3.oas.annotations.Parameter;  
import io.swagger.v3.oas.annotations.enums.ParameterIn;  
import io.swagger.v3.oas.annotations.media.Schema;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.security.core.annotation.AuthenticationPrincipal;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.RequestBody;  
  
import javax.validation.Valid;  
import javax.servlet.http.HttpServletRequest;  
import java.security.Principal;  
import java.util.concurrent.atomic.AtomicInteger;  
  
@javax.annotation.Generated(value = "io.swagger.codegen.v3.generators.java.SpringCodegen", date = "2023-10-19T16:54:06.552045592Z[GMT]")  
@RestController  
@RequestMapping("/")  
public class TtApiController implements TtApi {  
  
 private static final Logger *log* = LoggerFactory.*getLogger*(TtApiController.class);  
  
 private final ObjectMapper objectMapper;  
  
 private final HttpServletRequest request;  
  
 @org.springframework.beans.factory.annotation.Autowired  
 public TtApiController(ObjectMapper objectMapper, HttpServletRequest request) {  
 this.objectMapper = objectMapper;  
 this.request = request;  
 }  
  
 private AtomicInteger key = new AtomicInteger();  
 public TimeTable TTable = new TimeTable();  
  
 public ResponseEntity<Time> createTime(@Parameter(in = ParameterIn.*DEFAULT*, description = "", required=true, schema=@Schema()) @Valid @RequestBody Time body, @AuthenticationPrincipal Principal principal) {  
 String accept = request.getHeader("Accept");  
 if (principal == null){  
 throw new ForbiddenException();  
 }  
 if (accept != null && accept.contains("application/json")) {  
 synchronized (TTable){  
 body.setTimeId(key.incrementAndGet()) ;  
 TTable.add(body);  
 }  
 return new ResponseEntity<Time>(body, HttpStatus.*CREATED*);  
 }  
  
 return new ResponseEntity<Time>(HttpStatus.*NOT\_IMPLEMENTED*);  
 }  
  
 public ResponseEntity<Void> delTime(@Parameter(in = ParameterIn.*PATH*, description = "Id of Time", required=true, schema=@Schema()) @PathVariable("time\_id") int timeId , @AuthenticationPrincipal Principal principal) {  
  
 if (principal == null){  
 throw new ForbiddenException();  
 }  
 TTable.remove(timeId);  
 return new ResponseEntity<Void>(HttpStatus.*OK*);  
 }  
  
 public ResponseEntity<TimeTable> getTt() {  
 String accept = request.getHeader("Accept");  
  
 if (accept != null && accept.contains("application/json")) {  
  
 synchronized (TTable){  
 return new ResponseEntity<TimeTable>(TTable,HttpStatus.*OK*);  
 }  
  
  
 }  
  
 return new ResponseEntity<TimeTable>(HttpStatus.*NOT\_IMPLEMENTED*);  
 }  
  
 public ResponseEntity<Time> getTtById(@Parameter(in = ParameterIn.*PATH*, description = "Id of Time", required=true, schema=@Schema()) @PathVariable("time\_id") int timeId, @AuthenticationPrincipal Principal principal) {  
 String accept = request.getHeader("Accept");  
 if (principal == null){  
 throw new ForbiddenException();  
 }  
 if (accept != null && accept.contains("application/json")) {  
 synchronized (TTable){  
 return new ResponseEntity<Time>(TTable.get(timeId),HttpStatus.*OK*);  
 }  
 }  
  
 return new ResponseEntity<Time>(HttpStatus.*NOT\_IMPLEMENTED*);  
 }  
  
}