ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доцент, к.т.н. |  |  |  | П.А. Степанов |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

­­

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| РАЗРАБОТКА ФОРМЫ ЛОГИНА |
| по дисциплине: ТЕХНОЛОГИИ РАЗРАБОТКИ СЕРВЕРНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4032 |  |  |  | М.В. Мухортов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

**Задание**

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

Вариант7: Ведение списка литературы согласно последнему актуальному ГОСТу

**Описание разрабатываемого продукта**

В ходе выполнения работы необходимо разработать серверное приложение средствами Spring-boot, с формой логина для выполнения операций записи и удаления литературы в списке литературы.

**Текст основных фрагментов кода**

**WebSecurityConfig.java**

@Configuration

public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.inMemoryAuthentication()

.withUser("guest").password("{noop}hello")

.authorities("ROLE\_USER");

}

@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());

}

}

**literature.js**

var app = angular.module('literature', []).config(function ($httpProvider) {

csrftoken = $("meta[name='\_csrf']").attr("content");

csrfheader = $("meta[name='\_csrf\_header']").attr("content");

$httpProvider.defaults.headers.common["X-CSRF-TOKEN"] = csrftoken;

$(document).ajaxSend(function(e, xhr, options) {

xhr.setRequestHeader(csrfheader, csrftoken);

});

});

app.controller("LiteratureController", function ($scope, $http) {

$scope.successGetLiteraturesCallback = function (response) {

$scope.literatureList = response.data;

$scope.errormessage = "";

};

$scope.errorGetLiteraturesCallback = function (error) {

console.log(error);

$scope.errormessage = "Unable to get list of literature";

};

$scope.getLiterature = function () {

$http.get('/public/rest/literature').then($scope.successGetLiteraturesCallback, $scope.errorGetLiteraturesCallback);

};

$scope.successDeleteLiteratureCallback = function (response) {

for (var i = 0; i < $scope.literatureList.length; i++) {

var j = $scope.literatureList[i];

if (j.id === $scope.deletedId) {

$scope.literatureList.splice(i, 1);

break;

}

}

$scope.errormessage = "";

};

$scope.errorDeleteLiteratureCallback = function (error) {

console.log(error);

if (error.status === 405) {

$scope.errormessage = "You do not have permissions to do that";

} else

if (error.status === 403) {

$scope.errormessage = "You do not have permissions to do that";

} else

$scope.errormessage = "Unable to delete literature; check if there are any related records exist. Such records should be removed first";

};

$scope.deleteLiterature = function (id) {

$scope.deletedId = id;

$http.delete('/public/rest/literature/' + id).then($scope.successDeleteLiteratureCallback, $scope.errorDeleteLiteratureCallback);

};

$scope.successGetLiteratureCallback = function (response) {

$scope.literatureList.splice(0, 0, response.data);

$scope.errormessage = "";

};

$scope.errorGetLiteratureCallback = function (error) {

console.log(error);

$scope.errormessage = "Unable to get information on literature number " + $scope.inputnumber;

};

$scope.successAddLiteratureCallback = function (response) {

$http.get('/public/rest/literature/' + $scope.inputnumber).then($scope.successGetLiteratureCallback, $scope.errorGetLiteratureCallback);

$scope.errormessage = "";

};

$scope.errorAddLiteratureCallback = function (error) {

console.log(error);

if (error.status === 405) {

$scope.errormessage = "You do not have permissions to do that";

} else

if (error.status === 403) {

$scope.errormessage = "You do not have permissions to do that";

} else

$scope.errormessage = "Impossible to add new literature; check if it's number is unique";

};

$scope.addLiterature = function () {

$http.post('/public/rest/literature/' + $scope.inputnumber + "/" + $scope.inputname).then($scope.successAddLiteratureCallback, $scope.errorAddLiteratureCallback);

};

});