МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И ПРОГРАММИРОВАНИЯ

КАФЕДРА компьютерных технологий и программной инженерии

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | Кочин Д.А. |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

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

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4932 |  | 29.11.21 |  | Н.С. Иванов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

## Цель работы

Познакомится с основами написания форм авторизации и защиты ресурсов.

## Задание:

1. Добавьте в приложение поддержку одной из моделей безопасности средствами spring security.
2. Защитите ресурсы, требующие запись и введите аудит.

|  |
| --- |
| Вариант: 8. Учет трат в бюджете семьи. |

## Описание разрабатываемого продукта:

#### addCost

@PostMapping("add")  
public ResponseEntity<String> addCost(@RequestBody CostEntity cost){  
 try {  
 if(costService.addCost(cost)) {  
 return ResponseEntity.*status*(HttpStatus.*CREATED*).body(*gson*.toJson("Cost added successfully"));  
 }  
 throw new CostNotFoundException(*gson*.toJson("Cost don't found"));  
 } catch (CostNotFoundException e){  
 return ResponseEntity.*status*(HttpStatus.*NOT\_FOUND*).body(*gson*.toJson(e.getMessage()));  
 } catch (Exception e){  
 e.printStackTrace();  
 return ResponseEntity.*status*(HttpStatus.*INTERNAL\_SERVER\_ERROR*).body("CostController error -> " + e.getMessage() +  
 "\n\n" + Arrays.*toString*(e.getStackTrace()));  
 }  
}

#### deleteCost

@DeleteMapping("delete/{id}")  
public ResponseEntity<String> deleteCost(@PathVariable Long id){  
 try {  
 if (costService.deleteCost(id)){  
 return ResponseEntity.*status*(HttpStatus.*OK*).body(*gson*.toJson("Cost delete successfully"));  
 }  
 throw new CostNotFoundException("Cost don't found");  
 } catch (CostNotFoundException e){  
 return ResponseEntity.*status*(HttpStatus.*NOT\_FOUND*).body(*gson*.toJson(e.getMessage()));  
 }catch (Exception e){  
 return ResponseEntity.*status*(HttpStatus.*INTERNAL\_SERVER\_ERROR*).body(*gson*.toJson("CostController error"));  
 }  
}

#### putCost

@PutMapping("put/{id}")  
public ResponseEntity<String> putCost(@RequestBody CostEntity cost){  
 try {  
 if(costService.putCost(cost)) {  
 return ResponseEntity.*status*(HttpStatus.*ACCEPTED*).body(*gson*.toJson("Cost updated successfully"));  
 }  
 throw new CostNotFoundException(*gson*.toJson("Cost don't found"));  
 } catch (CostNotFoundException e){  
 return ResponseEntity.*status*(HttpStatus.*NOT\_FOUND*).body(*gson*.toJson(e.getMessage()));  
 }catch (Exception e){  
 return ResponseEntity.*status*(HttpStatus.*INTERNAL\_SERVER\_ERROR*).body(*gson*.toJson("CostController error"));  
 }  
}

#### getCostList

@GetMapping("get/all")  
public ResponseEntity<Object> getCostList(){  
 try {  
 return ResponseEntity.*status*(HttpStatus.*OK*).body(costService.getCosts());  
 }catch (Exception e){  
 return ResponseEntity.*status*(HttpStatus.*INTERNAL\_SERVER\_ERROR*).body(*gson*.toJson("CostController error"));  
 }  
}

#### getCost

@GetMapping("get/{id}")  
public ResponseEntity<String> getCost(@PathVariable Long id){  
 try {  
 return ResponseEntity.*status*(HttpStatus.*FOUND*).body(*gson*.toJson(costService.getCost(id)));  
 }catch (CostNotFoundException e){  
 return ResponseEntity.*status*(HttpStatus.*NOT\_FOUND*).body(*gson*.toJson(e.getMessage()));  
 }catch (Exception e){  
 return ResponseEntity.*status*(HttpStatus.*INTERNAL\_SERVER\_ERROR*).body("CostController error");  
 }  
}

### Form

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Costs Management (lab3)</title>  
 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>  
  
 <style>  
 .input\_form{  
 width: 40%;  
 }  
  
 input{  
 width: 97%;  
 }  
  
 button.btn{  
 width:100%;  
 height:100%;  
 min-width:30px;  
 max-width:200px;  
 min-height:30px;  
 max-height:30px;  
  
 color: black;  
 background-color: white;  
 border-color: black;  
 border-width: 1px;  
 border-radius: 7px;  
 font-size: 20px;  
 }  
  
 button.green\_btn{  
 width: 10%;  
 height: 10%;  
 color: black;  
 background-color: palegreen;  
 border-color: seagreen;  
 }  
  
 button.green\_btn:hover{  
 border: none;  
 color: gray;  
 }  
  
 button.red\_btn{  
 width:100%;  
 height:10%;  
  
 color: white;  
 background-color: palevioletred;  
 border-color: darkred;  
 }  
  
 button.red\_btn:hover{  
 border: none;  
 color: black;  
 }  
  
 button.orange\_btn{  
 width:100%;  
 height:10%;  
  
 color: white;  
 background-color: darkorange;  
 border-color: orangered;  
 }  
  
 button.orange\_btn:hover{  
 border: none;  
 color: black;  
 }  
  
 table {  
 background: #f5f5f5;  
 border-collapse: separate;  
 box-shadow: inset 0 1px 0 #fff;  
 font-size: 20px;  
 /\*line-height: 24px;\*/  
 margin: 30px auto;  
 text-align: left;  
 width: 90%;  
 }  
  
 th {  
 background: lightgray;  
 border-left: 1px solid #555;  
 border-right: 1px solid #777;  
 border-top: 1px solid #555;  
 border-bottom: 1px solid #333;  
 box-shadow: inset 0 1px 0 #999;  
 color: #000;  
 font-weight: bold;  
 padding: 10px 15px;  
 position: relative;  
 }  
  
 th:after {  
 background: linear-gradient(rgba(255,255,255,0), rgba(255,255,255,.08));  
 content: '';  
 display: block;  
 height: 25%;  
 left: 0;  
 margin: 1px 0 0 0;  
 position: absolute;  
 top: 25%;  
 width: 100%;  
 }  
  
 th:first-child {  
 border-left: 1px solid #777;  
 box-shadow: inset 1px 1px 0 #999;  
 }  
  
 th:last-child {  
 box-shadow: inset -1px 1px 0 #999;  
 }  
  
 td {  
 border-right: 1px solid #fff;  
 border-left: 1px solid #e8e8e8;  
 border-top: 1px solid #fff;  
 border-bottom: 1px solid #e8e8e8;  
 padding: 5px 5px;  
 line-height: 5px;  
 position: relative;  
 transition: all 300ms;  
 }  
  
 td:first-child {  
 box-shadow: inset 1px 0 0 #fff;  
 }  
  
 td:last-child {  
 border-right: 1px solid #e8e8e8;  
 box-shadow: inset -1px 0 0 #fff;  
 }  
  
 tr {  
 }  
  
 tr:nth-child(odd) td {  
 background: #f1f1f1;  
 }  
  
 tr:last-of-type td {  
 box-shadow: inset 0 -1px 0 #fff;  
 }  
  
 tr:last-of-type td:first-child {  
 box-shadow: inset 1px -1px 0 #fff;  
 }  
  
 tr:last-of-type td:last-child {  
 box-shadow: inset -1px -1px 0 #fff;  
 }  
  
 tbody:hover td {  
 color: transparent;  
 text-shadow: 0 0 3px #aaa;  
 }  
  
 tbody:hover tr:hover td {  
 color: #444;  
 text-shadow: 0 1px 0 #fff;  
 }  
  
 </style>  
  
</head>  
  
<script type="text/javascript">  
 $(***document***).ready(function (){  
 refresh();  
 })  
  
 function refresh(){  
 draw\_table();  
 }  
  
 function draw\_table(){  
 $.ajax({  
 url: "costs/get/all",  
 dataType: "json",  
 contentType: "application/json",  
 type: "GET",  
 async: false,  
 cache: false,  
 success: function (data) {  
 let $costs = $('#costs\_table')  
  
 let table = "<table border='1'>";  
 table +=  
 " <tr>\n" +  
 " <th>id</th>\n" +  
 " <th>Name</th>\n" +  
 " <th>Price</th>\n" +  
 " <th>Category</th>\n" +  
 " <th>Customer</th>\n" +  
 " <th>Count</th>\n" +  
 " <th></th>\n" +  
 " <th></th>\n" +  
 " </tr>";  
  
 for(let i=0; i<data.length; i++) {  
 table +=  
 "<tr>\n" +  
 " <td>" + data[i].id + "</td>\n" +  
 " <td>" + data[i].name + "</td>\n" +  
 " <td>" + data[i].price + "</td>\n" +  
 " <td>" + data[i].category + "</td>\n" +  
 " <td>" + data[i].customer + "</td>\n" +  
 " <td>" + data[i].count + "</td>\n" +  
 " <td>\n" +  
 " <button class='btn red\_btn' onclick='deleteCost(" + data[i].id + ")'>Delete</button>\n" +  
 " </td>\n" +  
 " <td>\n" +  
 // " <br>" +  
 " <button class='btn orange\_btn' onclick='editCost(" + data[i].id + ")'>Edit</button>\n" +  
 " </td>\n" +  
 "</tr>";  
 }  
  
 table += "</table>";  
  
 $costs.html(table);  
 },  
 error: function (error) {  
 console.log(error);  
 }  
 });  
 }  
  
 function addCost() {  
 $.ajax({  
 url: "/costs/add/",  
 dataType: "json",  
 contentType: "application/json",  
 type: "POST",  
 async: false,  
 cache: false,  
 data: JSON.stringify({  
 name: $("#fiendName").val(),  
 price: $("#fieldPrice").val(),  
 category: $("#fieldCategory").val(),  
 customer: $("#fieldCustomer").val()  
 }),  
 success: function () {  
 refresh();  
 },  
 error: function (error) {  
 console.log(error);  
 }  
 });  
 }  
  
 function deleteCost(idCost){  
 $.ajax({  
 url:'costs/delete/'+ idCost,  
 type: "DELETE",  
 async: false,  
 cache: false,  
 dataType: "json",  
 contentType: "application/json; charset=utf-8",  
 success: function (){  
 refresh();  
 },  
 error: function (error){  
 console.log(error);  
 }  
 })  
 }  
  
 function editCost(idCost){  
 $.ajax({  
 url: "/costs/put/" + idCost,  
 dataType: "json",  
 contentType: "application/json",  
 type: "PUT",  
 async: false,  
 cache: false,  
 data: JSON.stringify({  
 id: idCost,  
 name: $("#fiendName").val(),  
 price: $("#fieldPrice").val(),  
 category: $("#fieldCategory").val(),  
 customer: $("#fieldCustomer").val()  
 }),  
 success: function (){  
 refresh();  
 },  
 error: function (error){  
 console.log(error);  
 }  
 })  
 }  
  
</script>  
  
<body>  
 <center>  
 <h1>Costs Management</h1>  
 <div>  
 <form>  
 <table class="input\_form" border="1" cellpadding="5" align="center">  
 <tr><th>Name</th> <td>  
 <input type="text" name="name" id="fiendName"> </td></tr>  
 <tr><th>Price</th> <td>   
 <input type="text" name="price" id="fieldPrice"> </td></tr>  
 <tr><th>Category</th> <td>  
 <input type="text" name="category" id="fieldCategory"> </td></tr>  
 <tr><th>Customer</th> <td>  
 <input type="text" name="customer" id="fieldCustomer"> </td> </tr>  
 </table>  
 </form>  
 </div>  
 <button class="right btn green\_btn" type="button" onclick='addCost()'>Add</button>  
 <br>  
 <br>  
 <div id='costs\_table'></div>  
 </center>  
  
</body>  
</html>

### WebSecurityConfig

@Configuration  
@EnableWebSecurity  
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {  
  
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
 http.csrf().disable()  
  
 .authorizeRequests()  
 .antMatchers("/", "/home-page").permitAll()  
 .antMatchers("/read-access-pane").hasRole("USER")  
 .antMatchers("/full-access-pane").hasRole("ADMIN")  
 .and()  
 .formLogin()  
 .loginPage("/login")  
 .permitAll()  
 .and()  
 .logout()  
 .permitAll()  
 .and()  
 .exceptionHandling().accessDeniedPage("/forbidden-error");  
 }  
  
  
 @Bean  
 @Override  
 protected UserDetailsService userDetailsService() {  
 UserDetails user = User.*withDefaultPasswordEncoder*()  
 .username("user")  
 .password("1")  
 .roles("USER")  
 .build();  
  
 UserDetails admin = User.*withDefaultPasswordEncoder*()  
 .username("admin")  
 .password("1")  
 .roles("ADMIN")  
 .build();  
  
 return new InMemoryUserDetailsManager(user, admin);  
 }  
  
}

## Выводы

* Spring security сильно упрощает контроль доступа к ресурсам.

## Приложение:

// @filename \src\main\java\suai\trsis2021\lab4\Lab4Application.java

package suai.trsis2021.lab4;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class Lab4Application {

public static void main(String[] args) {

SpringApplication.run(Lab4Application.class, args);

}

}

// @filename \src\main\java\suai\trsis2021\lab4\controller\CostController.java

package suai.trsis2021.lab4.controller;

import com.google.gson.Gson;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

import suai.trsis2021.lab4.entity.CostEntity;

import suai.trsis2021.lab4.exceptions.CostNotFoundException;

import suai.trsis2021.lab4.service.CostService;

import java.util.Arrays;

@RestController

@RequestMapping("/costs")

public class CostController {

@Autowired

private CostService costService;

private static final Gson gson = new Gson();

@PostMapping("add")

public ResponseEntity<String> addCost(@RequestBody CostEntity cost){

try {

if(costService.addCost(cost)) {

return ResponseEntity.status(HttpStatus.CREATED).body(gson.toJson("Cost added successfully"));

}

throw new CostNotFoundException(gson.toJson("Cost don't found"));

} catch (CostNotFoundException e){

return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(gson.toJson(e.getMessage()));

} catch (Exception e){

e.printStackTrace();

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body("CostController error -> " + e.getMessage() +

"\n\n" + Arrays.toString(e.getStackTrace()));

}

}

@DeleteMapping("delete/{id}")

public ResponseEntity<String> deleteCost(@PathVariable Long id){

try {

if (costService.deleteCost(id)){

return ResponseEntity.status(HttpStatus.OK).body(gson.toJson("Cost delete successfully"));

}

throw new CostNotFoundException("Cost don't found");

} catch (CostNotFoundException e){

return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(gson.toJson(e.getMessage()));

}catch (Exception e){

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body(gson.toJson("CostController error"));

}

}

@PutMapping("put/{id}")

public ResponseEntity<String> putCost(@RequestBody CostEntity cost){

try {

if(costService.putCost(cost)) {

return ResponseEntity.status(HttpStatus.ACCEPTED).body(gson.toJson("Cost updated successfully"));

}

throw new CostNotFoundException(gson.toJson("Cost don't found"));

} catch (CostNotFoundException e){

return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(gson.toJson(e.getMessage()));

}catch (Exception e){

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body(gson.toJson("CostController error"));

}

}

@GetMapping("get/all")

public ResponseEntity<Object> getCostList(){

try {

return ResponseEntity.status(HttpStatus.OK).body(costService.getCosts());

}catch (Exception e){

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body(gson.toJson("CostController error"));

}

}

@GetMapping("get/{id}")

public ResponseEntity<String> getCost(@PathVariable Long id){

try {

return ResponseEntity.status(HttpStatus.FOUND).body(gson.toJson(costService.getCost(id)));

}catch (CostNotFoundException e){

return ResponseEntity.status(HttpStatus.NOT\_FOUND).body(gson.toJson(e.getMessage()));

}catch (Exception e){

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body("CostController error");

}

}

}

// @filename \src\main\java\suai\trsis2021\lab4\controller\WebController.java

package suai.trsis2021.lab4.controller;

import lombok.extern.slf4j.Slf4j;

import org.springframework.http.ResponseEntity;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.ResponseBody;

import org.springframework.web.bind.annotation.RestController;

import org.springframework.web.servlet.view.RedirectView;

import java.io.File;

import java.nio.charset.StandardCharsets;

import java.nio.file.Files;

import java.nio.file.Path;

@Controller

public class WebController {

@GetMapping("")

public RedirectView redirect(){

return new RedirectView("/home-page");

}

@RequestMapping("/login")

public String login(){

return "LoginPage";

}

@GetMapping("/home-page")

public String homePage() {

return "HomePage";

}

@GetMapping("/full-access-pane")

public String getAdminPage(){

return "FullAccessPane";

}

@GetMapping("/read-access-pane")

public String getReadPage() {

return "ReadAccessPane";

}

@GetMapping("/forbidden-error")

public String getForbiddenPage() {

return "ForbiddenPage";

}

}

// @filename \src\main\java\suai\trsis2021\lab4\entity\CostEntity.java

package suai.trsis2021.lab4.entity;

import lombok.\*;

import org.hibernate.Hibernate;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import java.util.Objects;

@Entity(name = "cost\_entity")

public class CostEntity {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Getter @Setter private Long id;

@Getter @Setter private String name;

@Getter @Setter private Integer price;

@Getter @Setter private String category;

@Getter @Setter private String customer;

@Getter @Setter private Integer count;

public CostEntity() {

count = 1;

}

public void countUp(){

count += 1;

}

public void countDown(){

count -= 1;

}

}

// @filename \src\main\java\suai\trsis2021\lab4\exceptions\CostNotFoundException.java

package suai.trsis2021.lab4.exceptions;

public class CostNotFoundException extends Exception{

public CostNotFoundException(String message) {

super(message);

}

}

// @filename \src\main\java\suai\trsis2021\lab4\model\Cost.java

package suai.trsis2021.lab4.model;

import lombok.\*;

import suai.trsis2021.lab4.entity.CostEntity;

public class Cost {

public static Cost toModel(CostEntity entity){

Cost cost = new Cost();

cost.id = entity.getId();

cost.name = entity.getName();

cost.price = entity.getPrice();

cost.category = entity.getCategory();

cost.customer = entity.getCustomer();

cost.count = entity.getCount();

return cost;

}

@Getter @Setter private Long id;

@Getter @Setter private String name;

@Getter @Setter private Integer price;

@Getter @Setter private String category;

@Getter @Setter private String customer;

@Getter @Setter private Integer count;

public Cost() {

count = 1;

}

public void countUp(){

count += 1;

}

public void countDown(){

count -= 1;

}

}

// @filename \src\main\java\suai\trsis2021\lab4\repository\CostRepository.java

package suai.trsis2021.lab4.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Modifying;

import org.springframework.data.jpa.repository.Query;

import org.springframework.stereotype.Repository;

import suai.trsis2021.lab4.entity.CostEntity;

import java.util.List;

import java.util.Optional;

@Repository

public interface CostRepository extends JpaRepository<CostEntity, Long> {

@Modifying

@Query("update cost\_entity c " +

"set c.name = ?1, " +

" c.price = ?2," +

" c.category = ?3," +

" c.customer = ?4," +

" c.count = ?5 " +

"where c.id = ?6")

void updateCostById(String name,

Integer price,

String category,

String customer,

Integer count,

Long id);

List<CostEntity> findByName(String name);

Optional<CostEntity> findById(Long id);

}

// @filename \src\main\java\suai\trsis2021\lab4\service\CostService.java

package suai.trsis2021.lab4.service;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import suai.trsis2021.lab4.entity.CostEntity;

import suai.trsis2021.lab4.exceptions.CostNotFoundException;

import suai.trsis2021.lab4.model.Cost;

import suai.trsis2021.lab4.repository.CostRepository;

import java.util.LinkedList;

import java.util.List;

import java.util.Optional;

@Service

@Transactional

public class CostService {

@Autowired

private CostRepository costRepository;

public boolean addCost(CostEntity cost){

if(cost.getName() == null ||

cost.getPrice() == null ||

cost.getCustomer() == null ||

cost.getCategory() == null) {

return false;

}

try {

CostEntity c = findCostEntity(cost);

if(c == null){

costRepository.save(cost);

}

else{

c.countUp();

updateCostById(c);

}

return true;

}

catch (Exception e){

e.printStackTrace();

return false;

}

}

public boolean deleteCost(Long id){

try {

Optional<CostEntity> c = costRepository.findById(id);

if(c.isEmpty()){

return false;

}

CostEntity cost = c.get();

if(cost.getCount() > 1){

cost.countDown();

updateCostById(cost);

}

else{

costRepository.deleteById(id);

}

return true;

}catch (Exception e){

return false;

}

}

public boolean putCost(CostEntity cost){

if(cost.getName() == null ||

cost.getPrice() == null ||

cost.getCustomer() == null ||

cost.getCategory() == null) {

return false;

}

Optional<CostEntity> c = costRepository.findById(cost.getId());

if(c.isEmpty()){

return false;

}

CostEntity costEntity = c.get();

costEntity.setName(cost.getName());

costEntity.setPrice(cost.getPrice());

costEntity.setCategory(cost.getCategory());

costEntity.setCustomer(cost.getCustomer());

updateCostById(costEntity);

return true;

}

public List<Cost> getCosts(){

List<Cost> costs = new LinkedList<>();

costRepository.findAll().forEach(e -> costs.add(Cost.toModel(e)));

return costs;

}

public Cost getCost(Long id) throws CostNotFoundException {

var cost = costRepository.findById(id);

if(cost.isEmpty()){

throw new CostNotFoundException("Cost don't found");

}

return Cost.toModel(cost.get());

}

private boolean equalsCosts(CostEntity e1, CostEntity e2){

return e1.getName().equals(e2.getName()) &&

e1.getCustomer().equals(e2.getCustomer()) &&

e1.getPrice().equals(e2.getPrice()) &&

e1.getCategory().equals(e2.getCategory());

}

public CostEntity findCostEntity(CostEntity cost) {

for (var c : costRepository.findByName(cost.getName())) {

if (equalsCosts(c, cost)) {

return c;

}

}

return null;

}

private void updateCostById(CostEntity cost){

costRepository.updateCostById(

cost.getName(),

cost.getPrice(),

cost.getCategory(),

cost.getCustomer(),

cost.getCount(),

cost.getId()

);

}

}

// @filename \src\main\java\suai\trsis2021\lab4\util\WebSecurityConfig.java

package suai.trsis2021.lab4.util;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

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.configuration.EnableWebSecurity;

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

import org.springframework.security.core.Authentication;

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

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.web.bind.annotation.GetMapping;

import java.util.Arrays;

@Configuration

@EnableWebSecurity

public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeRequests()

.antMatchers("/", "/home-page").permitAll()

.antMatchers("/read-access-pane").hasRole("USER")

.antMatchers("/full-access-pane").hasRole("ADMIN")

.and()

.formLogin()

.loginPage("/login")

.permitAll()

.and()

.logout()

.permitAll()

.and()

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

}

@Bean

@Override

protected UserDetailsService userDetailsService() {

UserDetails user = User.withDefaultPasswordEncoder()

.username("user")

.password("1")

.roles("USER")

.build();

UserDetails admin = User.withDefaultPasswordEncoder()

.username("admin")

.password("1")

.roles("ADMIN")

.build();

return new InMemoryUserDetailsManager(user, admin);

}

}