

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

должность, уч. степень, звание

подпись, дата

Кочин Д.А.

инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №6

«Разработка AJAX приложения с использованием
Hibernate»

по курсу: Технологии разработки серверных информационных систем

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

4932

06.12.2021

С.И. Коваленко

подпись, дата

инициалы, фамилия

Санкт-Петербург 2021

Цель работы

Познакомится с основами взаимодействия приложения с базой данных посредством Hibernate.

Задание:

1. Добавьте базу данных, модель данных, JPA репозитории и сервисы.
2. Реализуйте операции чтения/записи из одной из предыдущих лабораторных работ (3 или 5) с помощью JPA

Вариант: 9. Складской учет.

Описание разрабатываемого продукта:

application.yaml

```
spring:
  jpa:
    hibernate:
      ddl-auto: update
      show-sql: true
    datasource:
      url: jdbc:mysql://localhost:3306/trsis_tessier #TODO: добавить свои данные для
авторизации в БД
      username: root #TODO: добавить свои данные для авторизации в БД
      password: root #TODO: добавить свои данные для авторизации в БД
```

ItemEntity

```
package suai.trsis2021.labs.entity;

import lombok.*;

import javax.persistence.*;
import java.time.LocalDate;

@Table(schema = "trsis_tessier")
@Entity(name = "item")
public class ItemEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Getter @Setter private Long id;
    @Getter @Setter private String section;
    @Getter @Setter private String name;
    @Getter @Setter private Integer count;

    @Column(name = "date_add")
    @Getter @Setter private LocalDate dateAdd;
    @Column(name = "storage_time")
    @Getter @Setter private Integer storageTime;
    @Column(name = "storage_price")
    @Getter @Setter private Integer storagePrice;

    public ItemEntity() {
        count = 1;
    }

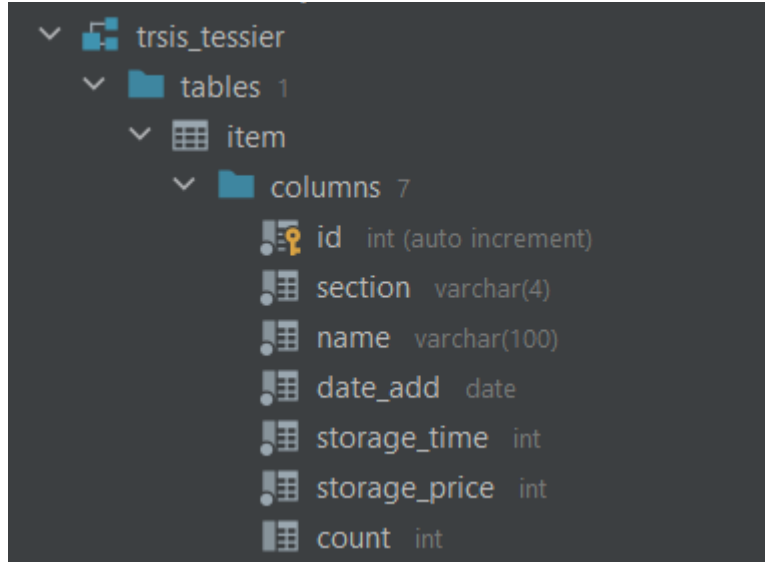
    public void countUp() {
        count += 1;
    }
}
```

```

    public void countDown() {
        count -= 1;
    }
}

```

table



| id | section | name | date_add | storage_time | storage_price | count |
|----------------------|------------|--------------|----------|--------------|---------------|-------|
| int (auto increment) | varchar(4) | varchar(100) | date | int | int | int |

ItemRepository

```

package suai.trsis2021.labs.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.labs.entity.ItemEntity;

import java.time.LocalDate;
import java.util.List;
import java.util.Optional;

@Repository
public interface ItemRepository extends JpaRepository<ItemEntity, Long> {

    @Modifying
    @Query("update item i " +
        "set i.section = ?1, " +
        "    i.name = ?2, " +
        "    i.dateAdd = ?3, " +
        "    i.storageTime = ?4, " +
        "    i.storagePrice = ?5, " +
        "    i.count = ?6 " +
        "where i.id = ?7")
    void updateCostById(
        String section,
        String name,
        LocalDate dateAdd,
        Integer storageTime,
        Integer storagePrice,
        Integer count,
        Long id
    );
}

```

```

        List<ItemEntity> findByName(String name);
        Optional<ItemEntity> findById(Long id);
    }

```

ItemService

```

package suai.trsis2021.labs.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import suai.trsis2021.labs.entity.ItemEntity;
import suai.trsis2021.labs.exceptions.ItemNotFoundException;
import suai.trsis2021.labs.model.Item;
import suai.trsis2021.labs.repository.ItemRepository;

import java.util.LinkedList;
import java.util.List;
import java.util.Optional;

@Service
@Transactional
public class ItemService {

    @Autowired
    private ItemRepository itemRepository;

    public boolean addItem(ItemEntity item){
        if(isEmptyItem(item)) {
            return false;
        }

        try {
            ItemEntity c = findItemEntity(item);
            if(c == null){
                itemRepository.save(item);
            }
            else{
                c.countUp();
                updateItemById(c);
            }
            return true;
        }
        catch (Exception e){
            e.printStackTrace();
            return false;
        }
    }

    public boolean deleteItem(Long id){
        try {
            Optional<ItemEntity> c = itemRepository.findById(id);
            if(c.isEmpty()){
                return false;
            }

            ItemEntity item = c.get();
            if(item.getCount() > 1){
                item.countDown();
                updateItemById(item);
            }
        }
    }

```

```

        else{
            itemRepository.deleteById(id);
        }
        return true;
    }catch (Exception e){
        return false;
    }
}

public boolean putItem(ItemEntity item){
    if(isEmptyItem(item)) {
        return false;
    }

    Optional<ItemEntity> c = itemRepository.findById(item.getId());
    if(c.isEmpty()){
        return false;
    }

    ItemEntity itemEntity = c.get();
    itemEntity.setSection(item.getSection());
    itemEntity.setName(item.getName());
    itemEntity.setDateAdd(item.getDateAdd());
    itemEntity.setStorageTime(item.getStorageTime());
    itemEntity.setStoragePrice(item.getStoragePrice());
    updateItemById(itemEntity);
    return true;
}

public List<Item> getItems(){
    List<Item> items = new LinkedList<>();
    itemRepository.findAll().forEach(e -> items.add(Item.toModel(e)));
    return items;
}

public Item getItem(Long id) throws ItemNotFoundException {
    var item = itemRepository.findById(id);
    if(item.isEmpty()){
        throw new ItemNotFoundException("Item don't found");
    }
    return Item.toModel(item.get());
}

public ItemEntity findItemEntity(ItemEntity item) {
    for (var c : itemRepository.findByName(item.getName())) {
        if (equalItems(c, item)) {
            return c;
        }
    }
    return null;
}

private void updateItemById(ItemEntity item){
    itemRepository.updateCostById(
        item.getSection(),
        item.getName(),
        item.getDateAdd(),
        item.getStorageTime(),
        item.getStoragePrice(),
        item.getCount(),
        item.getId()
    );
}

```

```

    );
}

private boolean equalItems(ItemEntity e1, ItemEntity e2){
    return e1.getSection().equals(e2.getSection())
        && e1.getName().equals(e2.getName())
        && e1.getDateAdd().equals(e2.getDateAdd())
        && e1.getStorageTime().equals(e2.getStorageTime())
        && e1.getStoragePrice().equals(e2.getStoragePrice());
}

private boolean isEmptyItem(ItemEntity item){
    return item.getSection() == null
        || item.getName() == null
        || item.getDateAdd() == null
        || item.getStorageTime() == null
        || item.getStoragePrice() == null;
}
}

```

Выводы

- В результате выполнения данной лабораторной работы проект был подключен к базе данных mysql.

Приложение:

```
// @filename \src\main\java\suai\trsis2021\labs\Lab4Application.java
package suai.trsis2021.labs;
```

```
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\labs\controller\RestApiController.java
package suai.trsis2021.labs.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.labs.entity.ItemEntity;
import suai.trsis2021.labs.exceptions.ItemNotFoundException;
import suai.trsis2021.labs.service.ItemService;
```

```
import java.util.Arrays;
```

```
@RestController
@RequestMapping("/items")
public class RestApiController {
```

```
    @Autowired
    private ItemService itemService;
    private static final Gson gson = new Gson();

    @PostMapping("add")
    public ResponseEntity<String> addCost(@RequestBody ItemEntity cost){
        try {
            if(itemService.addItem(cost)) {
                return ResponseEntity.status(HttpStatus.CREATED).body(gson.toJson("Item added successfully"));
            }
            throw new ItemNotFoundException(gson.toJson("Item don't found"));
        } catch (ItemNotFoundException 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("RestApiController error ->
" + e.getMessage() +
        "\n\n" + Arrays.toString(e.getStackTrace()));
    }
}

@DeleteMapping("delete/{id}")
public ResponseEntity<String> deleteCost(@PathVariable Long id){
    try {
        if (itemService.deleteItem(id)){
            return ResponseEntity.status(HttpStatus.OK).body(gson.toJson("Item delete
successfully"));
        }
        throw new ItemNotFoundException("Item don't found");
    } catch (ItemNotFoundException 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("RestApiContro
ller error"));
    }
}

@PutMapping("put/{id}")
public ResponseEntity<String> putCost(@RequestBody ItemEntity cost){
    try {
        if(itemService.putItem(cost)) {
            return ResponseEntity.status(HttpStatus.ACCEPTED).body(gson.toJson("Item updated
successfully"));
        }
        throw new ItemNotFoundException(gson.toJson("Item don't found"));
    } catch (ItemNotFoundException 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("RestApiContro
ller error"));
    }
}

@GetMapping("get/all")
public ResponseEntity<Object> getCostList(){
    try {
        return ResponseEntity.status(HttpStatus.OK).body(itemService.getItems());
    } catch (Exception e){
        return
        ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(gson.toJson("RestApiContro
ller error"));
    }
}

```



```

    }
}

@GetMapping("get/{id}")
public ResponseEntity<String> getCost(@PathVariable Long id){
    try {
        return
ResponseEntity.status(HttpStatus.FOUND).body(gson.toJson(itemService.getItem(id)));
    }catch (ItemNotFoundException e){
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body(gson.toJson(e.getMessage()));
    }catch (Exception e){
        return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("RestApiController error");
    }
}
}

```

```

// @filename \src\main\java\suai\trsis2021\labs\controller\WebController.java
package suai.trsis2021.labs.controller;

```

```

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.view.RedirectView;

```

```

@Controller
public class WebController {

```

```

    @GetMapping("")
    public RedirectView redirect(){
        return new RedirectView("/home");
    }

```

```

    @RequestMapping("/login")
    public String login(){
        return "LoginPage";
    }

```

```

    @GetMapping("/home")
    public String homePage() {
        return "HomePage";
    }

```

```

    @GetMapping("/crud-pane")
    public String getAdminPage(){
        return "CrudPane";
    }

```

```

    @GetMapping("/read-pane")

```

```

    public String getReadPage() {
        return "ReadPane";
    }

    @GetMapping("/forbidden-page")
    public String getForbiddenPage() {
        return "ForbiddenPage";
    }
}

// @filename \src\main\java\suai\trsis2021\labs\entity\ItemEntity.java
package suai.trsis2021.labs.entity;

import lombok.*;

import javax.persistence.*;
import java.time.LocalDate;

@Table(schema = "trsis_tessier")
@Entity(name = "item")
public class ItemEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Getter @Setter private Long id;
    @Getter @Setter private String section;
    @Getter @Setter private String name;
    @Getter @Setter private Integer count;

    @Column(name = "date_add")
    @Getter @Setter private LocalDate dateAdd;
    @Column(name = "storage_time")
    @Getter @Setter private Integer storageTime;
    @Column(name = "storage_price")
    @Getter @Setter private Integer storagePrice;

    public ItemEntity() {
        count = 1;
    }

    public void countUp(){
        count += 1;
    }

    public void countDown(){
        count -= 1;
    }
}

```

```
// @filename \src\main\java\suai\trsis2021\labs\exceptions\ItemNotFoundException.java
package suai.trsis2021.labs.exceptions;
```

```
public class ItemNotFoundException extends Exception{
    public ItemNotFoundException(String message) {
        super(message);
    }
}
```

```
// @filename \src\main\java\suai\trsis2021\labs\model\Item.java
package suai.trsis2021.labs.model;
```

```
import lombok.*;
import suai.trsis2021.labs.entity.ItemEntity;
```

```
import java.time.LocalDate;
```

```
public class Item {

    public static Item toModel(ItemEntity entity){
        Item item = new Item();

        item.setId(entity.getId());
        item.setSection(entity.getSection());
        item.setName(entity.getName());
        item.setDateAdd(entity.getDateAdd());
        item.setStorageTime(entity.getStorageTime());
        item.setStoragePrice(entity.getStoragePrice());
        item.setCount(entity.getCount());

        return item;
    }

    @Getter @Setter private Long id;
    @Getter @Setter private String section;
    @Getter @Setter private String name;
    @Getter @Setter private LocalDate dateAdd;
    @Getter @Setter private Integer storageTime;
    @Getter @Setter private Integer storagePrice;
    @Getter @Setter private Integer count;

    public Item() {
        count = 1;
    }

    public void countUp(){
        count += 1;
    }
}
```

```

    public void countDown(){
        count -= 1;
    }
}

```

```

// @filename \src\main\java\suai\trsis2021\labs\repository\ItemRepository.java
package suai.trsis2021.labs.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.labs.entity.ItemEntity;

```

```

import java.time.LocalDate;
import java.util.List;
import java.util.Optional;

```

```

@Repository
public interface ItemRepository extends JpaRepository<ItemEntity, Long> {

```

```

    @Modifying
    @Query("update item i " +
        "set i.section = ?1, " +
        "    i.name = ?2," +
        "    i.dateAdd = ?3," +
        "    i.storageTime = ?4," +
        "    i.storagePrice = ?5," +
        "    i.count = ?6 " +
        "where i.id = ?7")
    void updateCostById(
        String section,
        String name,
        LocalDate dateAdd,
        Integer storageTime,
        Integer storagePrice,
        Integer count,
        Long id
    );

    List<ItemEntity> findByName(String name);
    Optional<ItemEntity> findById(Long id);
}

```

```

// @filename \src\main\java\suai\trsis2021\labs\service\ItemService.java
package suai.trsis2021.labs.service;

```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import suai.trsis2021.labs.entity.ItemEntity;
import suai.trsis2021.labs.exceptions.ItemNotFoundException;
import suai.trsis2021.labs.model.Item;
import suai.trsis2021.labs.repository.ItemRepository;
```

```
import java.util.LinkedList;
import java.util.List;
import java.util.Optional;
```

```
@Service
@Transactional
public class ItemService {
```

```
    @Autowired
    private ItemRepository itemRepository;
```

```
    public boolean addItem(ItemEntity item){
        if(isEmptyItem(item)) {
            return false;
        }
```

```
        try {
            ItemEntity c = findItemEntity(item);
            if(c == null){
                itemRepository.save(item);
            }
            else{
                c.countUp();
                updateItemById(c);
            }
            return true;
        }
        catch (Exception e){
            e.printStackTrace();
            return false;
        }
    }
```

```
    public boolean deleteItem(Long id){
        try {
            Optional<ItemEntity> c = itemRepository.findById(id);
            if(c.isEmpty()){
                return false;
            }

```

```
            ItemEntity item = c.get();
```

```

        if(item.getCount() > 1){
            item.countDown();
            updateItemById(item);
        }
        else{
            itemRepository.deleteById(id);
        }
        return true;

    }catch (Exception e){
        return false;
    }
}

public boolean putItem(ItemEntity item){
    if(isEmptyItem(item)) {
        return false;
    }

    Optional<ItemEntity> c = itemRepository.findById(item.getId());
    if(c.isEmpty()){
        return false;
    }

    ItemEntity itemEntity = c.get();
    itemEntity.setSection(item.getSection());
    itemEntity.setName(item.getName());
    itemEntity.setDateAdd(item.getDateAdd());
    itemEntity.setStorageTime(item.getStorageTime());
    itemEntity.setStoragePrice(item.getStoragePrice());
    updateItemById(itemEntity);
    return true;
}

public List<Item> getItems(){
    List<Item> items = new LinkedList<>();
    itemRepository.findAll().forEach(e -> items.add(Item.toModel(e)));
    return items;
}

public Item getItem(Long id) throws ItemNotFoundException {
    var item = itemRepository.findById(id);
    if(item.isEmpty()){
        throw new ItemNotFoundException("Item don't found");
    }
    return Item.toModel(item.get());
}

public ItemEntity findItemEntity(ItemEntity item) {
    for (var c : itemRepository.findByName(item.getName())) {

```

```

        if (equalItems(c, item)) {
            return c;
        }
    }
    return null;
}

private void updateItemById(ItemEntity item){
    itemRepository.updateCostById(
        item.getSection(),
        item.getName(),
        item.getDateAdd(),
        item.getStorageTime(),
        item.getStoragePrice(),
        item.getCount(),
        item.getId()
    );
}

private boolean equalItems(ItemEntity e1, ItemEntity e2){
    return e1.getSection().equals(e2.getSection())
        && e1.getName().equals(e2.getName())
        && e1.getDateAdd().equals(e2.getDateAdd())
        && e1.getStorageTime().equals(e2.getStorageTime())
        && e1.getStoragePrice().equals(e2.getStoragePrice());
}

private boolean isEmptyItem(ItemEntity item){
    return item.getSection() == null
        || item.getName() == null
        || item.getDateAdd() == null
        || item.getStorageTime() == null
        || item.getStoragePrice() == null;
}
}

```

```

// @filename \src\main\java\suai\trsis2021\labs\util\WebSecurityConfig.java
package suai.trsis2021.labs.util;

```

```

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.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapt
er;
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;
```

```
@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-pane").hasAnyRole("USER", "ADMIN")
```

```
        .antMatchers("/crud-pane").hasRole("ADMIN")
```

```
        .and()
```

```
        .formLogin()
```

```
        .loginPage("/login")
```

```
        .permitAll()
```

```
        .and()
```

```
        .logout()
```

```
        .permitAll()
```

```
        .and()
```

```
        .exceptionHandling().accessDeniedPage("/forbidden-page");
```

```
    }
```

```
@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);
```

```
}
```

```
}
```

```
// @filename \src\main\resources\templates\CrudPane.html
```

```
<!DOCTYPE html>
```



```

<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>Складской учет</title>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
</head>

<script type="text/javascript">
  $(document).ready(function (){
    refresh();
  })

  function refresh(){
    draw_table();
  }

  function draw_table(){
    $.ajax({
      url: "items/get/all",
      dataType: "json",
      contentType: "application/json",
      type: "GET",
      async: false,
      cache: false,
      success: function (data) {
        let $items = $('#items_table')

        let table = "<table border='1'>";
        table +=
          " <tr>\n" +
          "   <th>id</th>\n" +
          "   <th>Секция</th>\n" +
          "   <th>Название товара</th>\n" +
          "   <th>Дата добавления</th>\n" +
          "   <th>Срок хранения</th>\n" +
          "   <th>Стоимость хранения</th>\n" +
          "   <th>Количество</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].section + "</td>\n" +
            "  <td>" + data[i].name + "</td>\n" +
            "  <td>" + data[i].dateAdd + "</td>\n" +
            "  <td>" + data[i].storageTime + "</td>\n" +
            "  <td>" + data[i].storagePrice + "</td>\n" +

```

```

        "    <td>" + data[i].count + "</td>\n" +
        "    <td>\n" +
        "        <button onclick='deleteCost(" + data[i].id + ")'>Delete</button>\n" +
        "    </td>\n" +
        "    <td>\n" +
        "        <br>" +
        "        <button onclick='editCost(" + data[i].id + ")'>Edit</button>\n" +
        "    </td>\n" +
        "</tr>";
    }

    table += "</table>";

    $items.html(table);
},
error: function (error) {
    console.log(error);
}
});
}

```

```

function addCost() {
    $.ajax({
        url: "/items/add/",
        dataType: "json",
        contentType: "application/json",
        type: "POST",
        async: false,
        cache: false,
        data: JSON.stringify({
            section: $("#fieldSection").val(),
            name: $("#fiendName").val(),
            dateAdd: $("#fieldDateAdd").val(),
            storageTime: $("#fieldStorageTime").val(),
            storagePrice: $("#fieldStoragePrice").val()
        }),
        success: function () {
            refresh();
        },
        error: function (error) {
            console.log(error);
        }
    });
}

```

```

function deleteCost(idCost){
    $.ajax({
        url: 'items/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: "/items/put/" + idCost,
        dataType: "json",
        contentType: "application/json",
        type: "PUT",
        async: false,
        cache: false,
        data: JSON.stringify({
            id: idCost,
            section: $("#fieldSection").val(),
            name: $("#fieldName").val(),
            dateAdd: $("#fieldDateAdd").val(),
            storageTime: $("#fieldStorageTime").val(),
            storagePrice: $("#fieldStoragePrice").val()
        }),
        success: function (){
            refresh();
        },
        error: function (error){
            console.log(error);
        }
    })
}

```

</script>

<body>

<div th:include="Logout"></div>

<h1>Складской учет</h1>

<div>

<form>

<table border="1" cellpadding="5">

<tr><th>Секция</th> <td>

<input type="text" name="section" id="fieldSection"> </td></tr>

<tr><th>Название товара</th> <td>

```

        <input type="text" name="name" id="fiendName"> </td></tr>
        <tr><th>Дата добавления</th> <td>
            <input type="date" name="date_add" id="fieldDateAdd"> </td></tr>
        <tr><th>Срок хранения</th> <td>
            <input type="number" name="storage_time" id="fieldStorageTime"> </td> </tr>
        <tr><th>Стоимость хранения</th> <td>
            <input type="number" name="storage_price" id="fieldStoragePrice"> </td> </tr>
    </table>
</form>
</div>
<button type="button" onclick='addCost()'>Добавить</button>
<br>
<br>
<div id="items_table"></div>

</body>
</html>

```

```

// @filename \src\main\resources\templates\ForbiddenPage.html
<!DOCTYPE html>
<html lang="ru">

```

```

<head>
    <meta charset="UTF-8">
    <title>Складской учет</title>
</head>

```

```

<body>
    <div style="text-align: center;">
        <h2>Доступ запрещен.</h2>
    </div>
</body>

```

```

// @filename \src\main\resources\templates\HomePage.html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:th="https://www.thymeleaf.org"
    xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity3"
    lang="ru">

```

```

<head>
    <title>Home</title>
</head>

```

```

<div th:include="Logout"></div>

```

```

<div>
    <h3>Pages</h3>

```

```

<ul>
  <li>
    <a href="login">Авторизация </a></li>
    <li sec:authorize="hasRole('ADMIN')">
      <a href="crud-pane">Панель администратора</a></li>
<!--    <li sec:authorize="hasRole('USER')">-->
      <li sec:authorize="hasAnyRole('USER', 'ADMIN')">
        <a href="read-pane">Панель пользователя </a></li>
    </ul>
</div>

</html>

```

```

// @filename \src\main\resources\templates\LoginPage.html
<!DOCTYPE html>
<html lang="ru"
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:th="https://www.thymeleaf.org"
  xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity3">
<head>
  <meta charset="UTF-8">
  <title>Логин</title>
</head>

<style>
div.error{
  color: #000000;
  background-color: rgba(219, 112, 147, 0.67);
  text-align: center;
  border-radius: 15px;
}
div.logout{
  color: #000000;
  background-color: rgba(152, 251, 152, 0.67);
  text-align: center;
  border-radius: 15px;
}
</style>

<body>

<div th:include="Logout"></div>
<a href="/" >Home</a>

<div style="
position: absolute;

```

```

top: 50%;
left: 50%;
margin-right: -50%;
transform: translate(-50%, -50%);
">
    <div class="error" th:if="${param.error}">
        Неверное имя пользователя или пароль.
    </div>

    <div class="logout" th:if="${param.logout}">
        Вы вышли из учетной записи.
    </div>

    <form th:action="@{/login}" method="post">
        <table border="0" cellpadding="5">
            <tr><th align="right"> Имя пользователя :</th> <td>
                <input type="text" name="username"> </td></tr>
            <tr><th align="right"> Пароль : </th> <td>
                <input type="password" name="password"> </td></tr>
            <tr><th> </th> <td align="right">
                <input type="submit" value="Войти"/> </td></tr>
        </table>
    </form>
</div>

</body>

</html>

// @filename \src\main\resources\templates\Logout.html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="https://www.thymeleaf.org"
    xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity3">

<body>

<form th:action="@{/logout}" method="post">
    <div th:if="${#request.userPrincipal != null}" th:inline="text">
        Вы вошли как: [[${#request.remoteUser}]]
        <input type="submit" value="Выйти"/>
    </div>
</form>
</body>
</html>

// @filename \src\main\resources\templates\ReadPane.html
<!DOCTYPE html>

```

```

<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>Складской учет</title>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
</head>

<script type="text/javascript">
  $(document).ready(function (){
    refresh();
  })

  function refresh(){
    draw_table();
  }

  function draw_table(){
    $.ajax({
      url: "items/get/all",
      dataType: "json",
      contentType: "application/json",
      type: "GET",
      async: false,
      cache: false,
      success: function (data) {
        let $items = $('#items_table')

        let table = "<table border='1'>";
        table +=
          " <tr>\n" +
          "   <th>id</th>\n" +
          "   <th>Секция</th>\n" +
          "   <th>Название товара</th>\n" +
          "   <th>Дата добавления</th>\n" +
          "   <th>Срок хранения</th>\n" +
          "   <th>Стоимость хранения</th>\n" +
          "   <th>Количество</th>\n" +
          " </tr>";

        for(let i=0; i<data.length; i++) {
          table +=
            "<tr>\n" +
            "  <td>" + data[i].id + "</td>\n" +
            "  <td>" + data[i].section + "</td>\n" +
            "  <td>" + data[i].name + "</td>\n" +
            "  <td>" + data[i].dateAdd + "</td>\n" +
            "  <td>" + data[i].storageTime + "</td>\n" +
            "  <td>" + data[i].storagePrice + "</td>\n" +
            "  <td>" + data[i].count + "</td>\n" +
            "</tr>";
        }
      }
    });
  }
}

```

```
    }

    table += "</table>";

    $items.html(table);
  },
  error: function (error) {
    console.log(error);
  }
});
}
</script>

<body>

  <div th:include="Logout"></div>

  <div>
    <h1>Складской учет</h1>
    <div id='items_table'></div>
  </div>

</body>

</html>
```