**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«Национальный исследовательский университет ИТМО»**

Факультет информационных технологий и программирования

Лабораторная работа № 4

*Авторизация*

**Выполнил студент группы M32051**

Писарева Юлия Игоревна

****

**Подпись:**

**Проверил:**

Бутенко Олег Романович

Санкт-Петербург

2022

**Текст задания**

Владельцы недовольны, что информацию о котиках может получить кто угодно. В этой лабораторной мы добавим авторизацию к сервису.  
  
Добавляется роль администратора. Он имеет доступ ко всем методам и может создавать новых пользователей. Пользователь связан с владельцем в соотношении 1:1.  
  
Методы по получению информации и котиках, и владельцах должны быть защищены Spring Security. Доступ к соответствующим endpoint’ам имеют только владельцы котиков и администраторы. Доступ к методам для фильтрации имеют все авторизованные пользователи, но на выходе получают только данные о своих котиках.

Внимание: эндпоинты, созданные на предыдущем этапе, не должны быть удалены.

**Решение с комментариями**

package com.kotiki.infrastructure.daos;

import com.kotiki.infrastructure.entities.Role;

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

import java.util.List;

public interface RoleDao extends JpaRepository<Role, Long> {

List<Role> findByName(String name);

}

package com.kotiki.infrastructure.daos;

import com.kotiki.infrastructure.entities.User;

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

public interface UserDao extends JpaRepository<User, Long> {

User findByUsername(String username);

}

package com.kotiki.infrastructure.entities;

import org.springframework.security.core.GrantedAuthority;

import javax.persistence.\*;

import java.util.Set;

@Entity

@Table(name = "roles")

public class Role implements GrantedAuthority {

@Id

private Long id;

private String name;

@Transient

@ManyToMany

@JoinTable(

name = "user\_roles",

joinColumns = { @JoinColumn(name = "user\_id") },

inverseJoinColumns = { @JoinColumn(name = "role\_id") }

)

private Set<User> users;

public Role() { }

public Role(Long id) { this.id = id; }

public Role(Long id, String name) {

this.id = id;

this.name = name;

}

@Override

public String getAuthority() { return name; }

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public void setName(String name) { this.name = name; }

public Set<User> getUsers() { return users; }

public void setUsers(Set<User> users) { this.users = users; }

}

package com.kotiki.infrastructure.entities;

import com.kotiki.core.entities.Owner;

import org.springframework.security.core.GrantedAuthority;

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

import javax.persistence.\*;

import javax.validation.constraints.Size;

import java.util.Collection;

import java.util.Collections;

import java.util.Set;

@Entity

@Table(name = "users")

public class User implements UserDetails {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Size(min=6, message = "Не меньше 6 знаков")

private String username;

@Size(min=6, message = "Не меньше 6 знаков")

private String password;

@ManyToMany(fetch = FetchType.EAGER)

private Set<Role> roles;

@OneToOne

@JoinColumn(name = "owner\_id")

private Owner owner;

public User() { }

public User(String username, String password, Set<Role> roles, Owner owner) {

this.username = username;

this.password = password;

this.roles = roles;

this.owner = owner;

}

public Long getId() { return id; }

@Override

public String getUsername() { return username; }

@Override

public boolean isAccountNonExpired() { return true; }

@Override

public boolean isAccountNonLocked() { return true; }

@Override

public boolean isCredentialsNonExpired() { return true; }

@Override

public boolean isEnabled() { return true; }

@Override

public Collection<? extends GrantedAuthority> getAuthorities() { return getRoles(); }

@Override

public String getPassword() { return password; }

public void setUsername(String username) { this.username = username; }

public void setPassword(String password) { this.password = password; }

public Set<Role> getRoles() { return roles; }

public void setRoles(Set<Role> roles) { this.roles = roles; }

public void addRoles(Role... roles) { Collections.addAll(this.roles, roles); }

public void setOwner(Owner owner) { this.owner = owner; }

public Owner getOwner() { return owner; }

}

package com.kotiki.infrastructure.services;

import com.kotiki.infrastructure.daos.RoleDao;

import com.kotiki.infrastructure.entities.Role;

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

import org.springframework.stereotype.Service;

@Service

public class RoleService {

@Autowired

private RoleDao roleDao;

public Role findRoleByName(String name) { return roleDao.findByName(name).stream().findFirst().orElse(null); }

}

package com.kotiki.infrastructure.services;

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

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.crypto.bcrypt.BCryptPasswordEncoder;

@Configuration

@EnableWebSecurity

public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

@Autowired

private UserService userService;

@Autowired

private BCryptPasswordEncoder bCryptPasswordEncoder;

@Autowired

private void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {

auth.userDetailsService(userService).passwordEncoder(bCryptPasswordEncoder);

}

@Override

protected void configure(HttpSecurity httpSecurity) throws Exception {

httpSecurity

.csrf().disable()

.authorizeRequests()

.antMatchers("/owner/\*\*").hasAnyAuthority("OWNER", "ADMIN")

.antMatchers("/cats/\*\*").hasAuthority("ADMIN")

.antMatchers("/admin/\*\*").hasAuthority("ADMIN")

.and()

.formLogin()

.loginProcessingUrl("/login")

.defaultSuccessUrl("/")

.permitAll()

.and()

.logout()

.logoutUrl("/logout")

.deleteCookies("JSESSIONID")

.permitAll();

}

}

package com.kotiki.infrastructure.services;

import com.kotiki.dataAccess.tools.KotikiExeption;

import com.kotiki.infrastructure.daos.UserDao;

import com.kotiki.infrastructure.entities.User;

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

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

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

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

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class UserService implements UserDetailsService {

@Autowired

private UserDao userDao;

@Autowired

private BCryptPasswordEncoder bCryptPasswordEncoder;

@Override

public UserDetails loadUserByUsername(String username) throws KotikiExeption {

User user = userDao.findByUsername(username);

if (user == null) { throw new KotikiExeption("user not found"); }

return user;

}

public List<User> allUsers() { return userDao.findAll(); }

public User saveUser(User user) {

user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));

return userDao.save(user);

}

public void deleteUser(Long userId) {

if (userDao.findById(userId).isPresent())

userDao.deleteById(userId);

}

public User getUser(Long id) { return userDao.findById(id).orElse(null); }

}

package com.kotiki.presentation.controllers;

import com.kotiki.infrastructure.services.InfrastructureCatService;

import com.kotiki.infrastructure.services.InfrastructureOwnerService;

import com.kotiki.infrastructure.services.RoleService;

import com.kotiki.infrastructure.services.UserService;

import com.kotiki.presentation.dtos.CatDto;

import com.kotiki.presentation.dtos.OwnerDto;

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

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

import java.util.List;

@RestController

@RequestMapping("admin")

public class AdministrationController {

@Autowired

private InfrastructureOwnerService ownerService;

@Autowired

private UserService userService;

@Autowired

private RoleService roleService;

@Autowired

private InfrastructureCatService catService;

@PostMapping("set")

public void setAdmin(@RequestParam Long userId) {

var user = userService.getUser(userId);

var role = roleService.findRoleByName("ADMIN");

user.addRoles(role);

}

@GetMapping("owners/all")

public List<OwnerDto> getAllOwners() { return ownerService.getAll().stream().map(OwnerDto::new).toList(); }

@GetMapping("owners/{id}")

public OwnerDto getOwnerWithId(@PathVariable Long id) { return new OwnerDto(ownerService.getById(id)); }

@GetMapping("owners/{id}/cats")

public List<CatDto> getCats(@PathVariable Long id) {

return catService.getAll().stream()

.filter(c -> c.getOwner() != null && c.getOwner().getId().equals(id))

.map(CatDto::new)

.toList();

}

}

package com.kotiki.presentation.controllers;

import com.kotiki.core.entities.Cat;

import com.kotiki.presentation.dtos.CatDto;

import com.kotiki.core.models.Color;

import com.kotiki.presentation.dtos.OwnerDto;

import com.kotiki.presentation.models.CreateCatModel;

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

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

import com.kotiki.infrastructure.services.InfrastructureCatService;

import java.time.LocalDateTime;

import java.util.List;

@RestController

@RequestMapping("cats")

public class CatController {

@Autowired

private InfrastructureCatService catService;

@GetMapping("all")

public List<CatDto> getAllCats() { return catService.getAll().stream().map(CatDto::new).toList(); }

@GetMapping("{id}")

public CatDto getCatWithId(@PathVariable Long id) { return new CatDto(catService.getById(id)); }

@GetMapping("color")

public List<CatDto> getCatsWithColor(@RequestParam String color) {

var colorEnum = Color.valueOf(color);

return catService.getAll().stream().filter(c -> c.getColor() == colorEnum).map(CatDto::new).toList();

}

@GetMapping("{id}/friends")

public List<CatDto> getCatsFriends(@PathVariable Long id) {

var cat = catService.getById(id);

return cat.getFriends().stream().map(CatDto::new).toList();

}

@GetMapping("{id}/owner")

public OwnerDto getOwner(@PathVariable Long id) {

var cat = catService.getById(id);

return new OwnerDto(cat.getOwner());

}

@PostMapping("create")

public void createCat(@RequestBody CreateCatModel model) {

var cat = new Cat(model.getCatName(), LocalDateTime.now(), model.getBreed(), model.getColor());

catService.addToDatabase(cat);

}

}

package com.kotiki.presentation.controllers;

import com.kotiki.core.models.Color;

import com.kotiki.infrastructure.entities.User;

import com.kotiki.infrastructure.services.InfrastructureCatService;

import com.kotiki.presentation.dtos.CatDto;

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

import org.springframework.security.core.Authentication;

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

import java.util.List;

@RestController

@RequestMapping("owner")

public class OwnerController {

@Autowired

private InfrastructureCatService catService;

@GetMapping("cats")

public List<CatDto> getCats(Authentication authentication) {

var user = (User) authentication.getPrincipal();

var owner = user.getOwner();

return catService.getAll().stream().filter(c -> owner.equals(c.getOwner())).map(CatDto::new).toList();

}

@GetMapping("cats/{color}")

public List<CatDto> getColoredCats(@PathVariable String color, Authentication authentication) {

var user = (User) authentication.getPrincipal();

var owner = user.getOwner();

var colorEnum = Color.valueOf(color);

return catService.getAll().stream()

.filter(c -> owner.equals(c.getOwner()) && c.getColor().equals(colorEnum))

.map(CatDto::new)

.toList();

}

@GetMapping("{breed}\_cats")

public List<CatDto> getCatsByBreed(@PathVariable String breed, Authentication authentication) {

var user = (User) authentication.getPrincipal();

var owner = user.getOwner();

return catService.getAll().stream()

.filter(c -> owner.equals(c.getOwner()) && c.getBreed().equals(breed))

.map(CatDto::new)

.toList();

}

}

package com.kotiki.presentation.controllers;

import com.kotiki.core.entities.Owner;

import com.kotiki.dataAccess.tools.KotikiExeption;

import com.kotiki.infrastructure.entities.User;

import com.kotiki.infrastructure.services.InfrastructureOwnerService;

import com.kotiki.infrastructure.services.RoleService;

import com.kotiki.infrastructure.services.UserService;

import com.kotiki.presentation.models.RegistrationModel;

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

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

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

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

import java.time.LocalDateTime;

import java.time.temporal.ChronoUnit;

import java.util.HashSet;

import java.util.List;

@RestController

public class RegistrationController {

@Autowired

private InfrastructureOwnerService ownerService;

@Autowired

private UserService userService;

@Autowired

private RoleService roleService;

@PostMapping("/register")

public String registerUser(@RequestBody RegistrationModel model) {

if (!uniqueUsername(model.getUsername())) throw new KotikiExeption("user name is not unique");

var owner = new Owner(model.getUsername(), LocalDateTime.now().minus(1, ChronoUnit.DAYS));

owner = ownerService.addToDatabase(owner);

var role = roleService.findRoleByName("OWNER");

var set = new HashSet<>(List.of(role));

var user = new User(model.getUsername(), model.getPassword(), set, owner);

user = userService.saveUser(user);

return user.getId().toString();

}

private boolean uniqueUsername(String username) {

return userService.allUsers().stream().map(User::getUsername).noneMatch(n -> n.equals(username));

}

}

package com.kotiki.presentation.models;

import com.kotiki.core.models.Color;

public class CreateCatModel {

private String catName;

private String breed;

private Color color;

public String getCatName() { return catName; }

public String getBreed() { return breed; }

public Color getColor() { return color; }

}

package com.kotiki.presentation.models;

public class RegistrationModel {

private String username;

private String password;

public String getUsername() { return username; }

public String getPassword() { return password; }

}

package com.kotiki.presentation;

import com.kotiki.core.entities.Cat;

import com.kotiki.infrastructure.daos.UserDao;

import com.kotiki.infrastructure.entities.Role;

import com.kotiki.presentation.controllers.CatController;

import com.kotiki.dataAccess.daos.CatDao;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.autoconfigure.domain.EntityScan;

import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;

import org.springframework.boot.jdbc.DataSourceBuilder;

import org.springframework.context.annotation.\*;

import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

import com.kotiki.core.services.CatService;

import com.kotiki.infrastructure.services.InfrastructureCatService;

import com.kotiki.core.services.OwnerService;

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

import javax.sql.DataSource;

@Configuration

@ComponentScan(basePackageClasses = { InfrastructureCatService.class, CatController.class })

@EntityScan(basePackageClasses = { Cat.class, Role.class })

@EnableJpaRepositories(basePackageClasses = { CatDao.class, UserDao.class })

@SpringBootApplication(exclude = { DataSourceAutoConfiguration.class })

public class Main {

public static void main(String[] args) {

var app = new SpringApplication(Main.class);

app.run(args);

}

@Bean

public BCryptPasswordEncoder bCryptPasswordEncoder() { return new BCryptPasswordEncoder(); }

@Bean

public DataSource datasource() {

return DataSourceBuilder.create()

.driverClassName("org.postgresql.Driver")

.url("jdbc:postgresql://localhost:5432/kotiki-java")

.username("postgres")

.password("1234567890")

.build();

}

@Bean

public CatService catService() { return new CatService(); }

@Bean

public OwnerService ownerService() { return new OwnerService(); }

@Bean

public BCryptPasswordEncoder encoder() { return new BCryptPasswordEncoder(); }

}