@Controller  
@RequestMapping("user")  
public class UserController2 {  
 @Autowired  
 private UserService userService;  
 @RequestMapping(value = "/me", method = RequestMethod.*GET*)  
 public String getMyUserPage(Model model, @AuthenticationPrincipal UserDetails currentUser) {  
 User = userService.getByNickname(currentUser.getUsername());  
 model.addAttribute("userInfo", user);  
 return "userPage";  
 }  
 @RequestMapping(value = "/{id}", method = RequestMethod.*GET*)  
 public String getUserPage(Model model,  
 @PathVariable("id") Long id) {  
 User = userService.getById(id);  
 model.addAttribute("userInfo", user);  
 return "userPage";  
 }  
 @RequestMapping(value = "/add", method = RequestMethod.*POST*)  
 public String addUser(@ModelAttribute("userForm") @Valid User userForm,   
 BindingResult, Model model) {  
 if (bindingResult.hasErrors()) {  
 return "redirect:/registration";  
 }  
 if (!userForm.getPassword().equals(userForm.getPasswordConfirm())){  
 model.addAttribute("errorMessage", "Пароли не совпадают");  
 return "registrationPage";  
 }  
 if (!userService.add(userForm)){  
 model.addAttribute("errorMessage", "Пользователь с таким именем уже существует");  
 return "registrationPage";  
 }  
 return "redirect:/login";  
 }  
 @RequestMapping(value = "/edit", method = RequestMethod.*PUT*)  
 public String editUser(@RequestBody User userForm) {  
 if (!userForm.getPassword().equals(userForm.getPasswordConfirm())){  
 return "redirect:/user/me/edit?error=1";  
 }  
 if (!userService.edit(userForm)){  
 return "redirect:/user/me/edit?error=2";  
 }  
 return "redirect:/user/me";  
 }  
 @DeleteMapping(value = "/{id}")  
 public String deleteUser(Model model, @AuthenticationPrincipal UserDetails currentUser,  
 @PathVariable("id") Long id) {  
 User = userService.getById(id);  
 userService.delete(user);  
 model.addAttribute("userInfo", user);  
 return "userPage";  
 }  
}

@Service  
@Transactional  
public class CommentServiceImpl implements CommentService {  
  
 @Autowired  
 private CommentRepository commentRepository;  
  
 @Override  
 public Iterable<Comment> getAllByAudiobook(Audiobook audiobook) {  
 return commentRepository.findAllByAudiobook(audiobook);  
 }  
  
 @Override  
 public void add(Comment comment) {  
 commentRepository.save(comment);  
 }  
  
 @Override  
 public void edit(Comment comment) { commentRepository.save(comment); }  
  
 @Override  
 public void delete(Comment comment) { commentRepository.delete(comment); }  
  
}

private final AudiobookService audiobookService;  
private final CommentService commentService;  
private final AudiobookFileService audiobookFileService;  
private final UserService userService;  
private final CreatorService creatorService;  
  
@Autowired  
public AudiobookController(AudiobookService audiobookService, CommentService commentService, AudiobookFileService audiobookFileService, UserService userService, CreatorService creatorService) {  
 this.audiobookService = audiobookService;  
 this.commentService = commentService;  
 this.audiobookFileService = audiobookFileService;  
 this.userService = userService;  
 this.creatorService = creatorService;  
}

@Repository  
public interface CrudRepository<T, ID> extends Repository<T, ID> {  
 <S extends T> S save(S var1);  
  
 <S extends T> Iterable<S> saveAll(Iterable<S> var1);  
  
 Optional<T> findById(ID var1);  
  
 boolean existsById(ID var1);  
  
 Iterable<T> findAll();  
  
 Iterable<T> findAllById(Iterable<ID> var1);  
  
 long count();  
  
 void deleteById(ID var1);  
  
 void delete(T var1);  
  
 void deleteAll(Iterable<? extends T> var1);  
  
 void deleteAll();  
}

@Entity  
@Table(name = "user")  
public class User {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 @Column(name = "id", nullable = false)  
 private Long id;  
  
 @NotNull  
 @Column(name = "nickname", nullable = false)  
 private String nickname;  
  
 @Column(name = "email", nullable = true)  
 private String email;  
  
 @NotNull  
 @Column(name = "password", length = 70, nullable = false)  
 private String password;  
  
 @ManyToOne(fetch = FetchType.*EAGER*)  
 @JoinColumn(name = "role\_id", nullable = false)  
 private Role role;  
  
 public User() { }  
  
 public Long getId() { return id; }  
  
 public String getNickname() { return nickname; }  
  
 public void setNickname(String nickname) { this.nickname = nickname; }  
  
 public String getEmail() { return email;}  
  
 public void setEmail(String email) { this.email = email; }  
  
 public String getPassword() { return password; }  
  
 public void setPassword(String password) { this.password = password; }  
  
 public Role getRole() {return role;}  
  
 public void setRole(Role role) { this.role = role;}  
  
}

<!DOCTYPE html>  
<html lang="ru"  
 xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org"  
 xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity4">  
<head>  
 <title>Spring Security Example </title>  
 <div th:replace="fragments/header :: header-css"/>  
</head>  
<body>  
 <div th:replace="fragments/header :: header"/>  
 <div id="content" class="container mt-4">  
 <div class="row justify-content-center" >  
 <div class="card col-lg-5 col-md-7 col-sm-10" th:object="${userInfo}">  
 <div class="card-body" >  
 <div class="row justify-content-center" >  
 <div class="col-9 mb-1" th:if="${userInfo.nickname.equals(#authorization.authentication.name)}">  
 <h2 class="card-title" style="text-align: center">Мой профиль</h2>  
 </div>  
 <div class="col-9 mb-1" th:unless="${userInfo.nickname.equals(#authorization.authentication.name)}">  
 <h2 class="card-title" style="text-align: center">Профиль пользователя <p th:text="${userInfo.nickname}"></p></h2>  
 </div>  
 </div>  
 <input type="hidden" required name="id" th:field="\*{id}"><br>  
 <div class="row justify-content-center">  
 <div class="mb-2 col-9">  
 <span class="" name="nickname" id="nickname">  
 <p th:text="'Никнейм: '+${userInfo.nickname}"></p></span>  
 </div>  
 </div>  
 <div class="row justify-content-center">  
 <div class="mb-2 col-9" id="email" name="email">  
 <div class="col" th:unless="${#strings.isEmpty(userInfo.email)}" >  
 <p th:text="'Email: '+ ${userInfo.email}"></p>  
 </div>  
 <p class="col" th:if="${#strings.isEmpty(userInfo.email)}">Email: не указан</p>  
 </div>  
 </div>  
 <div class="row justify-content-center">  
 <div class="mb-2 col-9" th:switch="${userInfo.role.id}">  
 <p th:case="2">Статус: администратор сайта</p>  
 <p th:case="3">Статус: Администратор сайта</p>  
 <p th:case="1">Статус: пользователь сайта</p>  
 </div>  
 </div>  
 <div class="row justify-content-center mt-2" th:if="${userInfo.nickname.equals(#authorization.authentication.name)}" >  
 <a id="edit-profile-btn" class="col-lg-7 col-md-8 col-sm-9 btn btn-primary btn-block"  
 th:href="@{/user/me/edit}">  
 Редактировать профиль  
 </a>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
 <div th:replace="fragments/footer :: footer"/>  
</body>  
</html>