Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| институт |
|  |
| Кафедра «Информатика» |
| кафедра |

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №5**

|  |
| --- |
| Конфигурация Spring Security |
| Тема |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Преподаватель | |  |  |  |  |  | А.С. Черниговский |
|  | |  |  |  | подпись, дата |  | инициалы, фамилия |
| Студент | КИ19-16/1б 031939175 | | |  |  |  | А.Д. Непомнящий |
|  | номер группы, зачетной книжки | | |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2021

1. Цель работы

Цель состоит в ознакомлении с настройкой безопасности в Spring.

1. Задачи

Задача работы – изменить практическую работу №4 таким образом, чтобы в ней был реализован следующий функционал.

1. Простейшая страница регистрации. Пользователь вводит свои логин и пароль, и данная информация вносится в базу данных, пользователю присваивается роль пользователя (User) приложения.
2. Простейшая форма аутентификации. Форма должна быть создана студентом, а не автоматически сгенерированной Spring.
3. Пользователь-администратор, с, отличной от роли User, ролью администратора (Admin).
4. Разграничение уровня доступа к страницам приложения. Пользователь (User) имеет доступ только к страницам просмотра всех записей и запросов. Администратор (Admin) имеет возможность добавлять, редактировать и удалять записи.
5. Хранение информации о пользователях в базе данных.
6. Возможность пользователю выйти из приложения (logout).
7. Продемонстрировать умение настраивать безопасность на уровне представлений. Для этого реализуйте приветствие пользователя после его входа и отображение элемента на основе его роли.

Вариант 10. Одежда.

1. Ход работы

В ходе работы было изменено в соответствии с заданием приложение, сделанное в ходе четвертой работы. В листингах приведен код некоторых файлов.

Листинг 1 – код класса SecurityConfiguration

@Configuration

@EnableWebSecurity

@ComponentScan(basePackageClasses = com.github.durakin.isdlabs.lab5.service.impl.UserDetailsServiceImpl.class)

public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

@Qualifier("userDetailsServiceImpl")

@Autowired

private UserDetailsService userDetailsService;

@Bean("passwordEncoder")

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

public DaoAuthenticationProvider authenticationProvider() {

DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();

authProvider.setUserDetailsService(userDetailsService);

authProvider.setPasswordEncoder(passwordEncoder());

return authProvider;

}

/\*

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.authenticationProvider(authenticationProvider());

}

\*/

@Override

protected void configure(final HttpSecurity http) throws Exception {

http

.csrf().disable()

.authorizeRequests()

.antMatchers("/", "/registration").permitAll()

.antMatchers("/find").hasAuthority("USER")

.antMatchers("/add", "/edit", "/delete", "/admin").hasAuthority("ADMIN")

.anyRequest().authenticated()

.and()

.exceptionHandling().accessDeniedPage("/error")

.and()

.formLogin()

.permitAll()

.and()

Окончание листинга 1

.logout()

.permitAll();

}

@Autowired

public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {

auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());

}

}

Листинг 2 – Код класса RegistrationController

@Controller

@RequestMapping("/registration")

public class RegistrationController {

private final PasswordEncoder passwordEncoder;

private final UserService userService;

public RegistrationController(@Qualifier("userServiceImpl") UserService userService, @Qualifier("passwordEncoder") PasswordEncoder passwordEncoder) {

this.userService = userService;

this.passwordEncoder = passwordEncoder;

}

@GetMapping

public String showAddShoesForm(Model model) {

model.addAttribute("userForm", new UserForm());

return "registration";

}

@PostMapping

public String saveUserToDb(@Valid UserForm userForm, BindingResult bindingResult, Model model) {

try {

User newUser = new User(userForm.getUsername(), passwordEncoder.encode(userForm.getPassword()));

userService.SaveUser(newUser);

System.out.println("New user created");

} catch (Exception e) {

System.out.println(e.getMessage());

}

return "redirect:/";

}

}

Листинг 3 – Код сущности пользователя

@Table(name = "users", indexes = {

@Index(name = "users\_name\_uindex", columnList = "name", unique = true)

})

@Entity

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

Окончание листинга 3

@Column(name = "id", nullable = false)

private Integer id;

@Column(name = "name", nullable = false)

private String name;

@Column(name = "password", nullable = false)

private String password;

@ManyToMany(fetch = FetchType.EAGER)

@JoinTable(

name = "users\_roles",

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

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

)

Set<Role> roles = new HashSet<>();

// Get- и Set- методы

Листинг 4 – Код сущности роли пользователя

@Table(name = "roles", indexes = {

@Index(name = "roles\_role\_uindex", columnList = "role", unique = true)

})

@Entity

public class Role {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id", nullable = false)

private Integer id;

@Column(name = "role", nullable = false)

private String role;

// Get- и Set- методы

1. Вывод

Было произведено ознакомление с настройкой безопасности в Spring.