

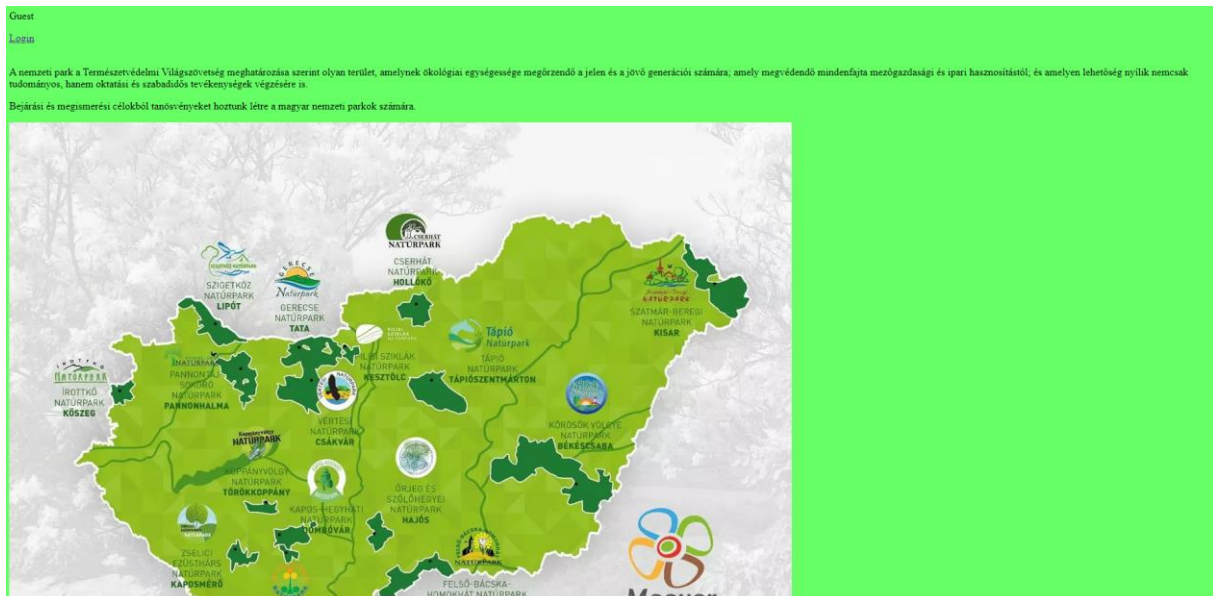
<https://github.com/TumzunAGozi/java-beadando-feladat>

1. Az első oldalon mutassa be a céget egy látványos weboldalon

Első oldal/Kezdőoldal kód:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org">
<head>
  <title>Guest Home</title>
</head>
<body style="background-color:rgba(0,255,0,0.6)">
<p>Guest</p>
<div th:insert="menu"></div>
<br />
<p>A nemzeti park a Természetvédelmi Világszövetség meghatározása szerint olyan terület, amelynek
ökológiai egységessége megőrzendő a jelen és a jövő generációi számára; amely megvédendő
mindenfajta mezőgazdasági és ipari hasznosítástól; és amelyen lehetőség nyílik nemcsak tudományos,
hanem oktatási és szabadidős tevékenységek végzésére is.<br /></p>
<p>Bejárási és megismerési célokból tanösvényeket hoztunk létre a magyar nemzeti parkok
számára.</p>

</body>
</html>
```




Login gombbal lehet eljutni a „Belépés” oldalhoz.

2. Legyen Regisztráció, Bejelentkezési lehetőség

- A „Belépés” menüpont akkor látható, ha nincs bejelentkezve a felhasználó.
- A „Kilépés” menüpont akkor látható, ha be van jelentkezve a felhasználó.

A rendszer fejlécen jelenítse meg a bejelentkezett felhasználót, ha be van lépve.

A Belépés menüpont:



Please sign in

Username

Password

Sign in

Kód:

```
<div xmlns:th="http://www.thymeleaf.org"
  xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity3">
  <div>
    <span sec:authorize="isAnonymous()">
      <a th:href="@{/login}">Login</a>
    </span>
    <span sec:authorize="isAuthenticated()">
      <a th:href="@{/home}">Home</a>
      <a th:href="@{/logout}">Logout</a>
    </span>
    <span sec:authorize="hasRole('ROLE_ADMIN')">
      <a th:href="@{/admin/home}">Admin</a>
    </span>
  </div>
  <div sec:authorize="isAuthenticated()">
    <h3>Welcome <span sec:authentication="principal.username">User</span></h3>
  </div>
</div>
```

Kilépés menüpont mind a „user” és „admin” oldalon látszik és kilép a kezdőoldalra.

Admin:

[Home](#) [Logout Admin](#)

Welcome admin@gmail.com

Only logged in users can see this page.

A teljes képernyős nézetből való kilépéshez nyomja le a következő 

Kód:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org">
<head>
  <title>Admin Home</title>
</head>
<body>
  <div th:insert="menu"></div>
  <h1>Only Admin can see this page</h1>
</body>
</html>
```

User:

[Home](#) [Logout](#)

Welcome user@gmail.com

Only logged in users can see this page.

Kód:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org">
<head>
  <title>User Home</title>
</head>
<body>
  <div th:insert="menu"></div>
  <h1>Only logged in users can see this page.</h1>
</body>
</html>
```

3. Legalább 3 felhasználói szerepet különböztessen meg:

Admin, User, Látogató

Admin

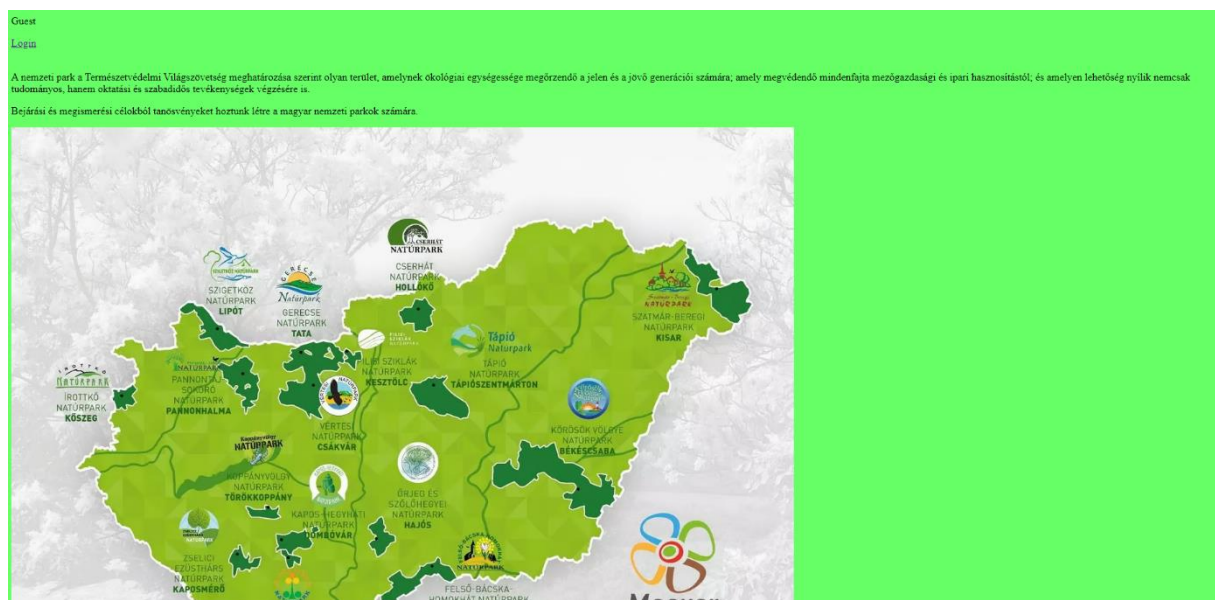
email: admin@gmail.com jelszó: jelszo1

User

email: user@gmail.com jelszó: jelszo2

Látogató=Guest

A „Guest” látszik a kezdőlap fejlécen és az oldal 1. sorában is.



9. Használják a GitHub (github.com) verziókövető rendszert.

Branchok: main -> main-edit1 -> main-edit2. Main-edit2 merge-elve lett a main-nel emiatt törölve is lett.

A main a végleges változat.

Egyéb kódok

BeadandoApplication

```
package com.example.beadando;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class BeadandoApplication {
    public BeadandoApplication() {
    }

    public static void main(String[] args) {
        SpringApplication.run(BeadandoApplication.class, args);
    }
}
```

SecurityRoleApplication

```
package com.example.securityrole;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SecurityRoleApplication {

    public static void main(String[] args) {

        SpringApplication.run(SecurityRoleApplication.class, args);
    }
}
```

CustomUserDetailsService

```
package com.example.securityrole;
import java.util.Collection;
```

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.AuthorityUtils;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service
@Transactional
public class CustomUserDetailsService implements UserDetailsService {
    @Autowired
    private UserRepository userRepo;          // Dependency injection
    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        User user = userRepo.findByEmail(username)
            .orElseThrow(() -> new UsernameNotFoundException("Email " + username + " not found"));
        return new org.springframework.security.core.userdetails.User(user.getEmail(),
            user.getPassword(),
            getAuthorities(user));
    }
    private static Collection<? extends GrantedAuthority> getAuthorities(User user) {
        Collection<GrantedAuthority> authorities = AuthorityUtils.createAuthorityList(user.getRole());
        return authorities;
    }
}

```

HomeController

```

package com.example.securityrole;

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

@Controller
public class HomeController {
    public HomeController() {
    }

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

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

```

```

    }

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

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

```

User

```

package com.example.securityrole;

import jakarta.persistence.*;

@Entity
@Table(name="users")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String name;
    private String email;
    private String password;
    private String role;

    public Integer getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

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

    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 String getRole() {
        return role;
    }

    public void setRole(String role) {
        this.role = role;
    }
}

```

UserRepository

```

package com.example.securityrole;

import org.springframework.data.repository.CrudRepository;

import java.util.Optional;

public interface UserRepository extends CrudRepository<User, Integer> {
    Optional<User> findByEmail(String email);
}

```

WebSecurityConfig

```

package com.example.securityrole;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfigur
ation;
import
org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;

```



```
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;
```

```
@Configuration
```

```
@EnableWebSecurity
```

```
@EnableGlobalMethodSecurity(securedEnabled = true, proxyTargetClass = true)
```

```
public class WebSecurityConfig {
```

```
    @Autowired
```

```
    private UserDetailsService userDetailsService;
```

```
    @Bean
```

```
    public static PasswordEncoder passwordEncoder(){
```

```
        return new BCryptPasswordEncoder();
```

```
    }
```

```
    @Bean
```

```
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
```

```
        http.csrf(csrf -> csrf.disable())
```

```
            .authorizeHttpRequests(
```

```
                auth -> auth
```

```
                    .requestMatchers("/resources/**", "/", "/home").authenticated()
```

```
                    .requestMatchers("/admin/**").hasRole("ADMIN")
```

```
            )
```

```
            .formLogin(
```

```
                form -> form
```

```
                    .defaultSuccessUrl("/home").permitAll()
```

```
            ).logout(
```

```
                logout -> logout
```

```
                    .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
```

```
                    .logoutSuccessUrl("/")
```

```
                    .permitAll()
```

```
            );
```

```
        return http.build();
```

```
    }
```

```
    @Bean
```

```
    public AuthenticationManager authenticationManager(AuthenticationConfiguration configuration)
```

```
    throws Exception {
```

```
        return configuration.getAuthenticationManager();
```

```
    }
```

```
}
```

ApplicationProperties

```
spring.application.name=demo
spring.datasource.url=jdbc:mysql://localhost:3306/adatok
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.format_sql=true
```