Spring security

Podstawy podstaw

Zależności

```
<properties>
    <springsecurity.version>4.1.4.RELEASE</springsecurity.version>
</properties>
<dependencies>
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-web</artifactId>
        <version>${springsecurity.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-config</artifactId>
        <version>${springsecurity.version}</version>
    </dependency>
<dependencies>
```

Klasa konfiguracyjna

```
@Configuration
@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {
    @Autowired
    public void configureGlobalSecurity(AuthenticationManagerBuilder auth) throws Exception {
        auth.inMemoryAuthentication().withUser("bill").password("abc123").roles("USER");
    }
    protected void configure(HttpSecurity http) throws Exception {
      http.authorizeRequests()
        .antMatchers("/", "/home").permitAll()
        .antMatchers("/admin/**").access("hasRole('ADMIN')")
        .antMatchers("/db/**").access("hasRole('ADMIN') and hasRole('DBA')")
        .and().formLogin()
        .and().exceptionHandling().accessDeniedPage("/Access_Denied");
    }
}
```

Dodatkowy initializer

```
public class SecurityWebApplicationInitializer
     extends AbstractSecurityWebApplicationInitializer {


}
```

Role

```java
@Autowired
public void configureGlobalSecurity(AuthenticationManagerBuilder auth) throws Exception {
     auth.inMemoryAuthentication().withUser("bill").password("abc123").roles("USER");
     auth.inMemoryAuthentication().withUser("admin").password("root123").roles("ADMIN");
     auth.inMemoryAuthentication().withUser("dba").password("root123").roles("ADMIN","DBA");
}
```

Mapowanie ról dla grup adresów

```
@Override
protected void configure(HttpSecurity http) throws Exception {
      http.authorizeRequests()
        .antMatchers("/home").permitAll()
        .antMatchers("/admin/**").access("hasRole('ADMIN')")
        .antMatchers("/db/**").access("hasRole('ADMIN') and hasRole('DBA')")
        .and().formLogin()
        .and().exceptionHandling().accessDeniedPage("/Access_Denied");
    }
```

Metody dla zmapowanych adresów

```
    @RequestMapping(value = { "/", "/home" })
    public String homePage() {
        return "welcome";
    }
    @RequestMapping(value = "/admin")
    public String adminPage() {
        return "admin";
    }
    @RequestMapping(value = "/db")
    public String dbaPage() {
        return "dba";
    }
```

Kontekst Spring Security

```java
public String getPrincipal() {
    String userName = null;
    Object principal = SecurityContextHolder.getContext().getAuthentication().getPrincipal();
    if (principal instanceof UserDetails) {
        userName = ((UserDetails) principal).getUsername();
    } else {
        userName = principal.toString();
    }
    return userName;
}
```

Niszczenie sesji w kontekście SS

```
@RequestMapping(value = "/logout")
public String logoutPage(HttpServletRequest request, HttpServletResponse response) {
    Authentication auth = SecurityContextHolder.getContext().getAuthentication();
    if (auth != null) {
        new SecurityContextLogoutHandler().logout(request, response, auth);
    }
    return "welcome";
}
```

Adnotacje do autoryzacji

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity (prePostEnabled = true)
public class KlasaKonfiguracyjna{}


@RequestMapping (value = "/adminpage")
@PreAuthorize ("hasRole('ADMIN')")
public String pageForAdmin() {
    return "adminPage";
}
```