SPRING REST Security

Papildomos bibliotekos

Įsidėkime papildomas bibliotekas:

```
<!-- spring-security-web and spring-security-config -->
 <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>
```

Spring security filtas

Konfigūracijos kataloge sukurkime klasę SecurityWebApplicationInitializer

```
import org.springframework.security.web.context.AbstractSecurityWebApplicationInitializer;

public class SecurityWebApplicationInitializer  extends AbstractSecurityWebApplicationInitializer {

}
```

Prisijungimų konfigūracija

Sukurkime prisijungimų konfigūravimo failą:

```
@Configuration
@EnableWebSecurity
public class DemoSecurityConfig extends WebSecurityConfigurerAdapter {
		@Override
		protected void configure(HttpSecurity http) throws Exception {

				// secures all REST endpoints under "/api/customers"
				http.authorizeRequests()
				.antMatchers("/api/customers/**").authenticated()
				.and()
				.httpBasic()
				.and()
				.csrf().disable()
				.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);
		}
}
```

Bibliotekų importavimas

Prieš tai buvusiam kodui mums reikės šių bibliotekų:

```
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
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.config.http.SessionCreationPolicy;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.User.UserBuilder;
```

Vartotojų prisijungimai

Sukurkime vartotojų prisijungimų konfigūraciją (metodą):

```
@Override
        protected void configure(AuthenticationManagerBuilder auth) throws Exception {

                UserBuilder users = User.withDefaultPasswordEncoder();

                auth.inMemoryAuthentication()
                        .withUser(users.username("john").password("test123").roles("EMPLOYEE"))
                        .withUser(users.username("mary").password("test123").roles("EMPLOYEE", "MANAGER"))
                        .withUser(users.username("susan").password("test123").roles("EMPLOYEE", "ADMIN"));
        }
```

Užduotis

Išbandykime autentifikavimą ar viskas sėkmingai veikia.

Jungiantis su POSTMAN: Authorization->Type->Basic Auth

Skirtingų teisių pridėjimas

Norėdami pridėti skirtingas teises galime tai padaryti priskiriant URL ir metodus:

```
http.authorizeRequests()
                                .antMatchers(HttpMethod.GET, "/api/customers").hasRole("EMPLOYEE")
                                .antMatchers(HttpMethod.GET, "/api/customers/*").hasRole("EMPLOYEE")
                                .antMatchers(HttpMethod.POST, "/api/customers").hasAnyRole("MANAGER", "ADMIN")
                                .antMatchers(HttpMethod.PUT, "/api/customers").hasAnyRole("MANAGER", "ADMINmary")
                                .antMatchers(HttpMethod.DELETE, "/api/customers/*").hasRole("ADMIN")
```

Užduotis

Išmėginkime skirtingas teises skirtingiems uždaviniams (metodams) atlikti.