Spring Web MVC Security

Presented by Ramin Zare

Spring Security

Spring Security راهکاری قوی , پویا برای برقراری امنیت در نرم افزار های سازمانی ارائه میدهد .

- روشی تعریف کردنی (Declarative) برای برنامه های Spring معرفی میکند
- Authentication و Authorization را مدیریت میکند
- میتوان در آن از مفاهیم و تکنیکهای DI - AOP استفاده کرد

Spring Security

- این سیستم با اکثر تکنولوژی ها سازگاری دارد :
  - AspectJ
  - JA-SIG CAS
  - OpenID
  - LDAP
  - Atlassian Crowd
  - jCaptcha
  - JAAS

قابلیت ها

- Authentication
- Web Url Authorization
- Method invocation Authorization
- ACL
- WS-Security
- Flow Authorization
- Remember Me
- Captcha

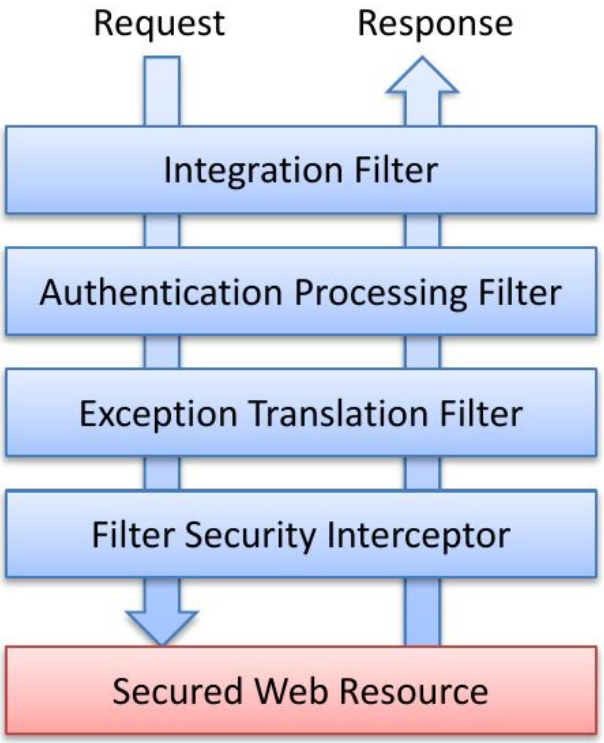
مفاهيم

- Filters
- Authentication
- Authorization
  - Web Authorization
  - Method Authorization

Method Authorization

```
@Secured("ROLE_ADMIN")
@Secured("ROLE_REGISTRAR")
public void enrollStudentInCourse(Course course, Student student)
throws CourseException {
    ......
}
```

Filters


Request
Response
Integration Filter
Authentication Processing Filter
Exception Translation Filter
Filter Security Interceptor
Secured Web Resource

Configuration

```
@Configuration
@EnableGlobalMethodSecurity(prePostEnabled = true , securedEnabled = true)
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

     private static final String DEFAULT_PASSWORD = new BCryptPasswordEncoder()
                                                        .encode("12345678");

   @Autowired
   public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
       auth.inMemoryAuthentication()
               .withUser("user")
               .password(DEFAULT_PASSWORD).roles("USER")
               .and()
               .withUser("admin")
               .password(DEFAULT_PASSWORD).roles("ADMIN");
   }

   @Bean
   public PasswordEncoder passwordEncoder() {
       return new BCryptPasswordEncoder();
   }
```

Configuration

```
@Override
protected void configure(HttpSecurity http) throws Exception {
   http.csrf().disable()
           .authorizeRequests()
           .antMatchers("/static/**/*").permitAll()
           .antMatchers("/home").permitAll()
           .antMatchers("/**").authenticated()
           //.hasAnyRole("USER").anyRequest()
           .and().formLogin()
           .successHandler((request, response, authentication)
                   -> {request.getSession(false).setAttribute("currentUser",
                   authentication.getPrincipal());
               response.sendRedirect("/product/top_product");})
           .loginPage("/login").permitAll()
           .and().logout().permitAll()
           .and().exceptionHandling().accessDeniedPage("/403");
}
```

JSP

```
<form action="${pageContext.servletContext.contextPath}/login" method="post">
   <div class="form-group">
       <br><br>
       <h2>Login to purchase products</h2>
       <br>
       <input type="text" name="username" placeholder="username" class="form-control" style="width: 200px;">
       <br>
       <input type="password" name="password" placeholder="password" class="form-control" style="width: 200px;">
       <br>
       <button type="submit" class="btn btn-primary" dir="ltr">Login To DIGIShop</button>
       <c:if test="${param.error ne null}">
           <br>
           <span style="color:red">User Name Or Password is not correct</span>
       </c:if>
       <c:if test="${param.logout ne null}">
           <br>
           <span style="color:green">You were logged out successfully</span>
       </c:if>
   </div>
</form>
```

Controller

```
@Controller
@RequestMapping("/order")
@Secured({"ROLE_USER" , "ROLE_ADMIN"})
public class OrderController {


@Controller
@RequestMapping("/admin/warehouse")
@PreAuthorize("hasAnyRole('ROLE_ADMIN')")
public class WarehouseAdmin {
```

JPA Authentication

```
@Service
public class UserDetailsServiceImpl implements UserDetailsService {
   @Autowired
   private UserRepository userRepository;

   @Override
   @Transactional(readOnly = true)
   public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
       User user = userRepository.findByUsername(username.toLowerCase());

       Set<GrantedAuthority> grantedAuthorities = new HashSet<>();
       for (Role role : user.getRoles()){
           grantedAuthorities.add(new SimpleGrantedAuthority(role.getName()));
       }

       return new
org.springframework.security.core.userdetails.User(user.getUsername(),
user.getPassword(), grantedAuthorities);
   }
}
```

Configuration

```
@Autowired
public void configureGlobal(AuthenticationManagerBuilder auth)
              throws Exception {
  auth.userDetailsService(userDetailsService)
                .passwordEncoder(passwordEncoder());
}
```

Thanks!

Presented by :

Ramin Zare