CS544 EA

Applications

Spring Security: Authorizing Web Requests

Turn off Filter for URL

- It is possible to turn off the security filter

```
@Configuration
@EnableWebSecurity
public class WebSecurityConfig {
    @Bean
    public WebSecurityCustomizer ignoringCustomizer() {
     return (web) -> web
                .debug(true)
                .ignoring().requestMatchers("/js/**", "/css/**");
    }
}
```

The 3rd config @Bean

Both /js/ and /css/ and everything below those directories has no security applied

We can also enable debugging here.

Spring Security can be very tricky to work with

```
<beans ...>

    <sec:http pattern="/js/**" security="none" />
    <sec:http pattern="/css/**" security="none" />
    <sec:debug />
```

XML equivalent

Debugging

- 2 things needed to debug Spring Security:
  - Enable in settings (previous slide)
  - Enable debug output on logger (log4j2.xml shown)

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="debug">
    <Appenders>
        <Console name="Console" target="SYSTEM_OUT">
            <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"/>
        </Console>
    </Appenders>
    <Loggers>
        <Root level="warn">
            <AppenderRef ref="Console"/>
        </Root>
        <Logger name="org.springframework.security" level="debug">
            <AppenderRef ref="Console"/>
        </Logger>
    </Loggers>
</Configuration>
```

Permit All Access

- Requests everyone should be able to make

```java
@Configuration
@EnableWebSecurity
public class SecurityConfig {
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http.authorizeHttpRequests(auth -> auth
                .requestMatchers("/login", "/logout", "/index")
                .permitAll()
        );
        return http.build();
    }
```

Allows everyone to request these URLs

```xml
    <sec:http>
        <sec:intercept-url pattern="/login" access="permitAll" />
        <sec:intercept-url pattern="/logout" access="permitAll" />
        <sec:intercept-url pattern="/index" access="permitAll" />
    </sec:http>
```

Role Access

```java
@Configuration
@EnableWebSecurity
public class WebSecurityConfig {
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http.authorizeHttpRequests(auth -> auth
                .requestMatchers(HttpMethod.GET, "/addContact")
                .hasRole("ADMIN")
            );
        return http.build();
    }
```

.requestMatchers() has optional HttpMethod arg

Specify the required role

```xml
    <sec:http>
        <sec:intercept-url method="GET" pattern="/addContact" access="hasRole('ADMIN')" />
    </sec:http>
```

Optional method property

Chaining & Order

- Order specified is **important**
  - Spring Sec goes top to bottom until match (then stops)

```
@Configuration
@EnableWebSecurity
public class WebSecurityConfig {
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http)
            throws Exception {
        http.authorizeHttpRequests(auth -> auth
                .requestMatchers("/", "/index", "/login", "/logout").permitAll()
                .requestMatchers(HttpMethod.GET, "/addContact").hasRole("ADMIN")
                .requestMatchers(HttpMethod.POST).hasRole("ADMIN")
                .requestMatchers(HttpMethod.GET, "/contacts").hasRole("USER")
            )
            .formLogin(Customizer.withDefaults())
            .logout(Customizer.withDefaults());
        return http.build();
    }
```

Any POST needs ADMIN

XML Version

- Order in XML is important for the same reason

```
<sec:http>
    <sec:intercept-url pattern="/login.jsp" access="permitAll" />
    <sec:intercept-url pattern="/logout" access="permitAll" />
    <sec:intercept-url pattern="/index" access="permitAll" />
    <sec:intercept-url pattern="/addContact" method="GET" access="hasRole('ADMIN')" />
    <sec:intercept-url method="POST" access="hasRole('ADMIN')" />
    <sec:intercept-url pattern="/contacts" method="GET" access="hasRole('USER')" />
    <sec:form-login />
    <sec:logout />
</sec:http>
```

Expressions

- You can write expressions to specify multiple attributes that may be needed for authorization
  - Primarily XML, but also JavaConf

```
<sec:http>
    <sec:intercept-url pattern="/admin/**" access="hasRole('ADMIN') and hasIpAddress('192.168.1.0/24')" />
     ...
</sec:http>
```

```
        http.authorizeHttpRequests(auth -> auth
            .requestMatchers("/admin/**")
            .access(new WebExpressionAuthorizationManager(
                    "hasRole('Admin') and hasIpAddress('192.168.1.0/24')"))
```

Common Built-in Expressions

Expression	Description
hasRole([role])	Returns true if the principal has the role
hasAnyRole([role1,role2])	Returns true if the principal has any of the roles
principal	Gives direct access to the principal object
authentication	Gives direct access to the authentication object
permitAll	Always evaluates to true
denyAll	Always evaluates to false
isAnonymous()	Returns true if the principal is anonymous
isRememberMe()	Returns true if the principal is a remember-me user
isAuthenticated()	Returns true if the principal is not anonymous
isFullyAuthenticated()	Returns true if the principal is not anon or remember-me


MAHARISHI
HIGHER EDUCATION
FOR HIGHER CONSCIOUSNESS
INTERNATIONAL UNIVERSITY