(http://baeldung.com)

Writing a Custom Filter in
Spring Security

Last modified: November 30, 2017

> by baeldung (http://www.baeldung.com/author/baeldung/)

Security (http://www.baeldung.com/category/security-2/)
Spring (http://www.baeldung.com/category/spring/) +

I just announced the new *Spring Security 5* modules (primarily focused on OAuth2) in the course:

>> CHECK OUT LEARN SPRING SECURITY (/learn-spring-security-course#new-modules)

---

1. Overview

In this quick article, we'll focus on writing a custom filter for the Spring Security filter chain.

2. Creating the Filter

Spring Security provides a number of filters by default, and most of the time, these are enough.

But of course sometimes it's necessary to implement new functionality with create a new filter to use in the chain.

We'll start by implementing the *org.springframework.web.filter.GenericFilterBean (http://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/filter/GenericFilterBean.html)*.

The *GenericFilterBean* is a simple *javax.servlet.Filter* (http://docs.oracle.com/javaee/7/api/javax/servlet/Filter.html) implementation implementation that is Spring aware.

On to the implementation – we only need to implement a single method:

```
public class CustomFilter extends GenericFilterBean {

    @Override
    public void doFilter(
      ServletRequest request,
      ServletResponse response,
      FilterChain chain) throws IOException, ServletException {
        chain.doFilter(request, response);
    }
}
```

3. Using the Filter in the Security Config

We're free to choose either XML configuration or Java configuration to wire the filter into the Spring Security configuration.

3.1. Java Configuration

You can register the filter programmatically overriding the *configure* (http://docs.spring.io/spring-security/site/docs/current/apidocs/org/springframework/security/config/annotation/web/configuration/WebSecurityConfig org.springframework.security.config.annotation.web.builders.HttpSecurity-) method from *WebSecurityConfigurerAdapter* (http://docs.spring.io/spring-security/site/docs/current/apidocs/org/springframework/security/config/annotation/web/configuration/WebSecurityConfig For example, it works with the *addFilterAfter* method on a *HttpSecurity* (https://docs.spring.io/spring-security/site/docs/5.0.0.M5/api/org/springframework/security/config/annotation/web/builders/HttpSecurity.html) instance:

```
1    @Configuration
2    public class CustomWebSecurityConfigurerAdapter
3      extends WebSecurityConfigurerAdapter {
4
5        @Override
6        protected void configure(HttpSecurity http) throws Exception {
7            http.addFilterAfter(
8                new CustomFilter(), BasicAuthenticationFilter.class);
9        }
10   }
```

There are a couple of possible methods:

- **addFilterBefore(filter, class) (https://docs.spring.io/spring-security/site/docs/5.0.0.M5/api/org/springframework/security/config/annotation/web/builders/HttpSecurity.htm** – adds a *filter* before the position of the specified filter *class*
- **addFilterAfter(filter, class) (https://docs.spring.io/spring-security/site/docs/5.0.0.M5/api/org/springframework/security/config/annotation/web/builders/HttpSecurity.htm** – adds a *filter* after the position of the specified filter *class*
- **addFilterAt(filter, class) (https://docs.spring.io/spring-security/site/docs/5.0.0.M5/api/org/springframework/security/config/annotation/web/builders/HttpSecurity.htm** – adds a *filter* at the location of the specified filter *class*
- **addFilter(filter) (https://docs.spring.io/spring-security/site/docs/5.0.0.M5/api/org/springframework/security/config/annotation/web/builders/HttpSecurity.htm** – adds a *filter* that must be an instance of or extend one of the filters provided by Spring Security

3.2. XML Configuration

You can add the filter to the chain using the *custom-filter* tag and one of these names (http://docs.spring.io/autorepo/docs/spring-security/current/reference/htmlsingle/#filter-stack) to specify the position of your filter. For instance, it can be pointed out by the *after* attribute:

```
1    <http>
2        <custom-filter after="BASIC_AUTH_FILTER" ref="myFilter" />
3    </http>
4
5    <beans:bean id="myFilter" class="org.baeldung.security.filter.CustomFilter"/>
```

Here are all attributes to specify exactly a place your filter in the stack:

- *after* – describes the filter immediately after which a custom filter will be placed in the chain
- *before* – defines the filter before which our filter should be placed in the chain
- *position* – allows replacing a standard filter in the explicit position by a custom filter

4. Conclusion

In this quick article, we created a custom filter and wired that into the Spring Security filter chain.

As always, all code examples are available in the sample Github project (https://github.com/eugenp/tutorials/tree/master/spring-security-rest-basic-auth).

I just announced the new Spring Security 5 modules (primarily focused on OAuth2) in the course:

>> CHECK OUT LEARN SPRING SECURITY (/learn-spring-security-course#new-modules)
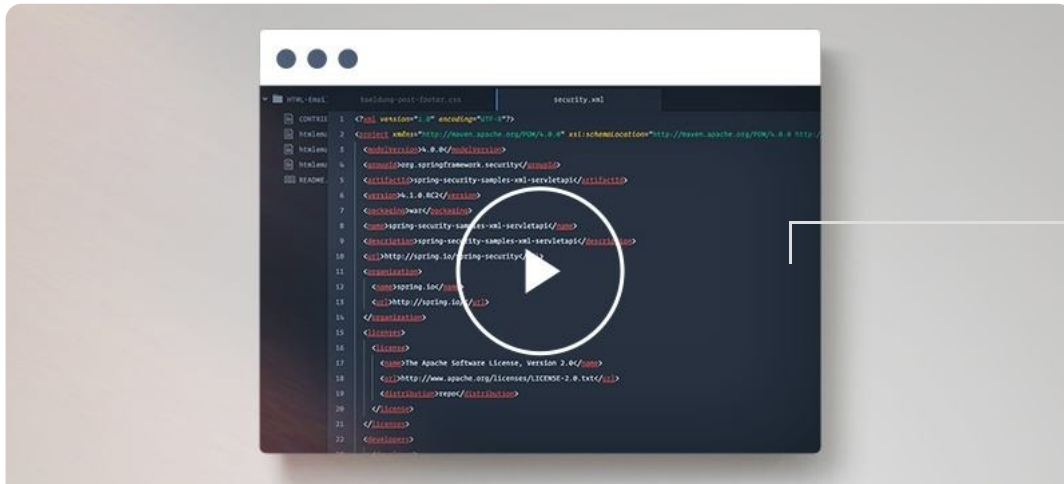
Learn the basics of securing a REST API with Spring

Get access to the video lesson!

Email Address

Access >>

Sort by: newest|oldest|most voted

Amit Patel

Guest

i want to use spring security in my spring mvc project , i am not using password for authentication , i am asking three question to user and he pass the three answers for these question if answers is correct login success otherwise failed how to implement spring security for this

👍 0 👎   ⊘ 1 year 20 days ago   ⌃

Eugen Paraschiv
(http://www.baeldung.com/)

Guest

Hey Amit,

Well, that's certainly an interesting case – it's very unusual to ask security questions as a primary means to authenticate the user. If that's really what you're doing (do let me know if I understood it wrong) – then you'll need to go into Spring Security and write your own logic, likely as a custom authentication provider.

But – before you do that, I would reconsider the approach and think through both the security implications (which are significant) but also the user experience for your system.

Hope that helps. Cheers,

Eugen.
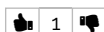
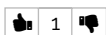👍 1 👎   ⊘ 1 year 18 days ago   ⌃

Amit Patel

Guest

I wrote the custom authentication for that as u suggested.

```
while(true){
System.out.println( " thank you very much . " )
}
```

👍 1 👎

🕐 11 months 10 days ago ⌃

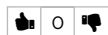
Eugen Paraschiv (http://www.baeldung.com/)

Guest

I'm glad everything's sorted out Amit. Cheers,

Eugen.

👍 0 👎

🕐 11 months 10 days ago

CATEGORIES

SPRING (HTTP://WWW.BAELDUNG.COM/CATEGORY/SPRING/)
REST (HTTP://WWW.BAELDUNG.COM/CATEGORY/REST/)
JAVA (HTTP://WWW.BAELDUNG.COM/CATEGORY/JAVA/)
SECURITY (HTTP://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/)
PERSISTENCE (HTTP://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/)
JACKSON (HTTP://WWW.BAELDUNG.COM/CATEGORY/JACKSON/)
HTTPCLIENT (HTTP://WWW.BAELDUNG.COM/CATEGORY/HTTP/)

ABOUT

ABOUT BAELDUNG (HTTP://WWW.BAELDUNG.COM/ABOUT/)
THE COURSES (HTTP://COURSES.BAELDUNG.COM)
META BAELDUNG (HTTP://META.BAELDUNG.COM/)
THE FULL ARCHIVE (HTTP://WWW.BAELDUNG.COM/FULL_ARCHIVE)
WRITE FOR BAELDUNG (HTTP://WWW.BAELDUNG.COM/CONTRIBUTION-GUIDELINES)
PRIVACY POLICY (HTTP://WWW.BAELDUNG.COM/PRIVACY-POLICY)
TERMS OF SERVICE (HTTP://WWW.BAELDUNG.COM/TERMS-OF-SERVICE)
CONTACT (HTTP://WWW.BAELDUNG.COM/CONTACT)
COMPANY INFO (HTTP://WWW.BAELDUNG.COM/BAELDUNG-COMPANY-INFO)
ADVERTISE ON THE JAVA WEEKLY (HTTP://WWW.BAELDUNG.COM/JAVA-WEEKLY-SPONSORSHIP)
CONSULTING WORK (HTTP://WWW.BAELDUNG.COM/CONSULTING)