Spring Interceptor Example, How to configure an interceptor in Spring MVC

By Yashwant Chavan, Views 47413, *Last updated on 13-Feb-2019*

One of my colleague facing some configuration issues with spring interceptor, Long time back I did that but not able to recall, how to configure spring interceptor. So started digging on this topic on Google and find out the solution. And outcome is in front of you in the form of this article. Lets have a closer look on how to configure Spring Interceptor.

tags spring spring-interceptor spring-aop

What is Spring Interceptor

In Web application when request comes to the controller , the HandlerMapping handler matches the incoming request. DispatcherServlet will hand it over to the handler mapping, to let it inspect the request and come up with an appropriate HandlerExecutionChain. Then the DispatcherServlet will execute the handler and interceptors in the chain. When it comes to interceptors you can perform the your logic like logging, method tracking etc.

Deployment Descriptor (web.xml) configuration

Define Web application deployment descriptor "web.xml" file, which is located under WEB-INF folder of application war file.

```
5.       <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
6.       <load-on-startup>1</load-on-startup>
7.    </servlet>
8.    <servlet-mapping>
9.       <servlet-name>springtutorial</servlet-name>
10.       <url-pattern>*.htm</url-pattern>
11.    </servlet-mapping>
12.    <welcome-file-list>
13.       <welcome-file>index.jsp</welcome-file>
14.    </welcome-file-list>
15. </web-app>
```

Spring Interceptor configuration (springtutorial-servlet.xml)

Here we configure two interceptors LoggerInterceptor and PerformanceInterceptor. Spring AOP provides the facility to control execution of interceptors with out modifying your existing code. E.g. In PerformanceInterceptor you can find out the method execution time base on your configuration , here we configure only for controller. For MethodInterceptor you need to define the proxyCreator.

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
3.    <bean id="helloController" class="com.net.technicalkeeda.controller.HelloController" />
4.    <bean id="loggerInterceptor" class="com.net.technicalkeeda.interceptor.LoggerInterceptor" />
5.    <bean id="performanceInterceptor" class="com.net.technicalkeeda.interceptor.PerformanceInterceptor" />
6.    <bean id="urlMapping" class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
7.       <property name="mappings">
8.          <props>
9.             <prop key="hello.htm">helloController</prop>
10.          </props>
11.       </property>
```

```
16.       </property>
17.    </bean>
18.    <bean name="proxyCreator" class="org.springframework.aop.framework.autoproxy.BeanNameAutoProxyCreator">
19.       <property name="beanNames">
20.          <list>
21.             <value>*Controller</value>
22.          </list>
23.       </property>
24.       <property name="interceptorNames">
25.          <list>
26.             <value>performanceInterceptor</value>
27.          </list>
28.       </property>
29.    </bean>
30. </beans>
```

Interceptor classes

LoggerInterceptor.java extends HandlerInterceptorAdapter , in that we need to override the preHandle(), postHandle() and afterCompletion() methods.

1] preHandle() method calls before the handler execution, returns a Boolean value "true" and "false".

2]postHandle() method calls after the handler execution, Here you can manipulate the ModelAndView object before render it to view page.

3]afterCompletion() method calls after the complete request has finished.

```
1. package com.net.technicalkeeda.interceptor;
```

```
6. import org.springframework.web.servlet.handler.HandlerInterceptorAdapter;
7.
8. public class LoggerInterceptor extends HandlerInterceptorAdapter {
9.
10.     @Override
11.     public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
12.                     throws Exception {
13.             System.out.println("Calling preHandle");
14.             return super.preHandle(request, response, handler);
15.     }
16.
17.     @Override
18.     public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler,
19.                     ModelAndView modelAndView) throws Exception {
20.             System.out.println("Calling postHandle");
21.             super.postHandle(request, response, handler, modelAndView);
22.     }
23.
24.     @Override
25.     public void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exce
26.                     throws Exception {
27.             System.out.println("Calling afterCompletion");
28.             super.afterCompletion(request, response, handler, ex);
29.     }
30. }
31.
```

PerformanceInterceptor.java class

Performance Interceptor implements the MethodInterceptor. Here you need to implement the invoke() method. It gives MethodInvocation object to retrieve the method details. In below example we find out the method execution time

```
4. import org.aopalliance.intercept.MethodInvocation;
5.
6. public class PerformanceInterceptor implements MethodInterceptor {
7.      public Object invoke(MethodInvocation method) throws Throwable {
8.              long startTime = System.currentTimeMillis();
9.              try {
10.                     Object result = method.proceed();
11.                     return result;
12.             } finally {
13.                     long endTime = System.currentTimeMillis();
14.                     long executionTime = endTime - startTime;
15.                     System.out.println(
16.                                     "Method Name: " + method.getMethod().getName() + " Execution Time:- " + ex
17.             }
18.     }
19. }
20.
```

Finish and Output

When you access the HelloController using URL "http://localhost:8080/SpringTutorial/hello.htm" it display below output , of both Interceptors on server console.

```
Calling preHandle
Calling HelloController.handleRequest()
Method Name: handleRequest Execution Time:- 1 ms
Calling postHandle
Calling afterCompletion
```

Comment *

Comment

Name *

Name

Email *

Email

Website

Website

Submit Cancel

Yashwant

Hi there! I am founder of technicalkeeda.com and programming enthusiast. My skills includes Java,J2EE, Spring Framework, Nodejs, PHP and lot more. If you have any idea that you would want me to develop? Lets connect: yashwantchavan[at][gmail.com]