**Using Spring MVC HandlerInterceptor with Spring Boot - Hello World example**

In this post we develop a simple **Spring Boot application with interceptors.**  
We use the interceptor to log the user activity. We will expose an API and then log the status of User interaction.

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.javainuse</groupId>

<artifactId>springboot-interceptor</artifactId>

<version>0.0.1</version>

<packaging>jar</packaging>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.0.RELEASE</version>

<relativePath/>

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

Next we define the LoggerInterceptor which implements the HandlerInterceptor.  
The LoggerInterceptor will override the following methods-

* preHandle() - This method is used to intercept the request before it is handed over to the handler method. If It turns a boolean value, true : continue the handler execution chain; false , stop the execution chain and return it.
* postHandle() - This method is used to intercept the request after the handler execution, Here user can manipulate the ModelAndView object before render it to view page.
* afterCompletion() - This is a HandlerInterceptor callback method called after the complete request has finished.

Define the class implementing the spring HandlerInterceptor interface as follows-

package com.javainuse.interceptor.config;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.slf4j.Logger;

import org.springframework.stereotype.Component;

import org.springframework.web.servlet.HandlerInterceptor;

import org.springframework.web.servlet.ModelAndView;

@Component

public class LoggerInterceptor implements HandlerInterceptor {

Logger log = org.slf4j.LoggerFactory.getLogger(this.getClass());

@Override

public void afterCompletion(HttpServletRequest request,

HttpServletResponse response, Object object, Exception arg3)

throws Exception {

log.info("Request is complete");

}

@Override

public void postHandle(HttpServletRequest request,

HttpServletResponse response, Object object, ModelAndView model)

throws Exception {

log.info("Handler execution is complete");

}

@Override

public boolean preHandle(HttpServletRequest request,

HttpServletResponse response, Object object) throws Exception {

log.info("Before Handler execution");

return true;

}

}

Next we register the custom loggerInterceptor using the Spring WebMvcConfigurerAdapter class.

package com.javainuse.interceptor.config;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.config.annotation.InterceptorRegistry;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

@Configuration

public class InterceptorConfig extends WebMvcConfigurerAdapter {

@Autowired

LoggerInterceptor logInterceptor;

@Override

public void addInterceptors(InterceptorRegistry registry) {

registry.addInterceptor(logInterceptor);

}

}

Finally we expose the LoggerController as follows-

package com.javainuse.interceptor.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class LoggerController {

Logger log = LoggerFactory.getLogger(this.getClass());

@RequestMapping("/logger")

public String executeLogger() {

log.info("inside the executeLogger method");

return "Hello World Logger Interceptor";

}

}

Finally create the SpringBoot class annotated with @SpringBootApplication.

package com.javainuse.interceptor.config;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringBootHelloWorldApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBootHelloWorldApplication.class, args);

}

}

After running the application- go to **http://localhost:8080/logger**  
  
  
In the console we can see the request gets intercepted as follows-  
  
