Spring 3 MVC Interceptor tutorial with example

Spring MVC provides a powerful mechanism to intercept an http request. Similar to [Servlet Filter concept](http://viralpatel.net/blogs/tutorial-java-servlet-filter-example-using-eclipse-apache-tomcat/), Spring MVC provides a way to define special classes called Interceptors that gets called before and after a request is served.

## **Quick Overview**

Each interceptor you define must implementorg.springframework.web.servlet.HandlerInterceptor interface. There are three methods that need to be implemented.

preHandle(..) is called before the actual handler is executed;

The preHandle(..) method returns a boolean value. You can use this method to break or continue the processing of the execution chain. When this method returns true, the handler execution chain will continue; when it returns false, the DispatcherServlet assumes the interceptor itself has taken care of requests (and, for example, rendered an appropriate view) and does not continue executing the other interceptors and the actual handler in the execution chain.

postHandle(..) is called after the handler is executed;

afterCompletion(..) is called after the complete request has finished.

These three methods should provide enough flexibility to do all kinds of preprocessing and postprocessing.

You can define an interceptor class as follow:

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.springframework.web.servlet.HandlerInterceptor;

import org.springframework.web.servlet.ModelAndView;

**public class HelloWorldInterceptor implements HandlerInterceptor  {**

    @Override

    public boolean preHandle(HttpServletRequest request,

            HttpServletResponse response, Object handler) throws Exception {

        System.out.println("Pre-handle");

        return true;

    }

    //override postHandle() and afterCompletion()

}

Once the interceptor is defined, you can ask Spring MVC to configure it via <mvc:interceptors>tag within spring-servlet.xml file.

<mvc:interceptors>

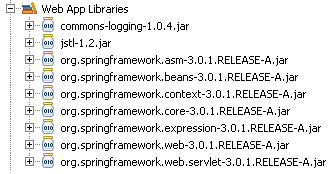
  <bean class="net.viralpatel.spring3.interceptor.HelloWorldInterceptor" />

</mvc:interceptors>

Let us start with the complete Demo application. We will create a demo Interceptor that logs each request.

Tools and technologies:

1. Java 5 (or above)
2. Spring MVC 3.0 (or above)
3. Eclipse 3.2 (or above)

We will need following JAR files in order to execute this project.  


If you are using [Apache Maven](http://viralpatel.net/blogs/introduction-apache-maven-build-framework-build-automation-tool/) as dependency management, add following dependencies to pom.xml.

|  |
| --- |
| <dependencies>    <dependency>        <groupId>org.springframework</groupId>        <artifactId>spring-webmvc</artifactId>        <version>3.0.1.RELEASE</version>    </dependency>    <dependency>        <groupId>jstl</groupId>        <artifactId>jstl</artifactId>        <version>1.2</version>    </dependency>  </dependencies> |

Let us create a simple Spring MVC Interceptor class that logs each request.

## **Step 1: The Controller – Create new Spring MVC Controller**

We create a simple Spring MVC controller that displays a plain JSP view.

HelloWorldController.java

|  |
| --- |
| package net.viralpatel.spring3.controller;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;    @Controller  public class HelloWorldController {      @RequestMapping(value = "/hello", method = RequestMethod.GET)      public String sayHello() {          return "hello";      }  } |

The controller has one method sayHello() which is mapped to URL **/hello** using@RequestMapping. This simply paints the hello.jsp view.

## **Step 2: The View – Create new JSPs**

Create two JSPs, hello.jsp that displays hello message and index.jsp that simply redirects first request to /hello.html.

/WEB-INF/jsp/hello.jsp

|  |
| --- |
| <html>  <head>      <title>Spring MVC Interceptor example</title>  </head>  <body>      <h1>Hello!!</h1>  </body>  </html> |

WebContent/index.jsp

|  |
| --- |
| <jsp:forward page="hello.html"></jsp:forward> |

The index.jsp simply redirects to hello.html which calls the HelloWorldController.

**Step 3: The Interceptor – Spring MVC HandlerInterceptor**

Let’s create the Spring based Handler Interceptor which will intercept the request and print a message.

HelloWorldInterceptor.java

|  |
| --- |
| package net.viralpatel.spring3.interceptor;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import org.springframework.web.servlet.HandlerInterceptor;  import org.springframework.web.servlet.ModelAndView;    **public class HelloWorldInterceptor implements HandlerInterceptor  {**      @Override      public boolean preHandle(HttpServletRequest request,              HttpServletResponse response, Object handler) throws Exception {            System.out.println("Pre-handle");          return true;      }        @Override      public void postHandle(HttpServletRequest request,              HttpServletResponse response, Object handler,              ModelAndView modelAndView) throws Exception {          System.out.println("Post-handle");      }        @Override      public void afterCompletion(HttpServletRequest request,              HttpServletResponse response, Object handler, Exception ex)              throws Exception {          System.out.println("After completion handle");      }  } |

Thus in each of the method preHandle, postHandle and afterCompletion we print a message on console.

**Step 4: Spring Configuration**

Now lets glue up the source code and configure Spring. Note how we declared the interceptor in spring-servlet.xml using <mvc:interceptors> tag.

spring-servlet.xml

|  |
| --- |
| <beans xmlns="http://www.springframework.org/schema/beans"      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"      xmlns:context="http://www.springframework.org/schema/context"      xmlns:mvc="http://www.springframework.org/schema/mvc"      xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd">        <context:annotation-config />      <context:component-scan base-package="net.viralpatel.spring3.controller"/>  **<mvc:interceptors>**  **<bean class="net.viralpatel.spring3.interceptor.HelloWorldInterceptor" />**  **</mvc:interceptors>**      <bean id="jspViewResolver"      class="org.springframework.web.servlet.view.InternalResourceViewResolver">          <property name="viewClass"              value="org.springframework.web.servlet.view.JstlView" />          <property name="prefix" value="/WEB-INF/jsp/" />          <property name="suffix" value=".jsp" />    </bean>  </beans> |

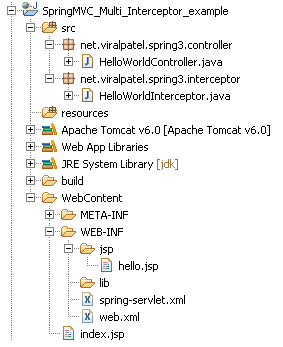
Also configure the spring’s DispatcherServlet in web.xml file.

web.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"      xmlns="http://java.sun.com/xml/ns/javaee"      xmlns:web="http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee          http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"      id="WebApp\_ID" version="2.5">        <display-name>Spring3MVC-Interceptor example</display-name>      <servlet>          <servlet-name>spring</servlet-name>          <servlet-class>              org.springframework.web.servlet.DispatcherServlet          </servlet-class>          <load-on-startup>1</load-on-startup>      </servlet>      <servlet-mapping>          <servlet-name>spring</servlet-name>          <url-pattern>\*.html</url-pattern>      </servlet-mapping>  </web-app> |

## **Final Project Structure**

Once all the source files and configuration files are in place, your project should look like below:



**Demo**

Compile and execute the project in Eclipse. Open below URL in browser.

**URL:** <http://localhost:8080/SpringMVC_Interceptor_example/>

Check the server console logs. You must see following:

|  |
| --- |
| Pre-handle  Post-handle  After completion handle |