<http://blog.csdn.net/qq924862077/article/details/53524507>

版权声明：http://blog.csdn.net/qq924862077/

对SpringMVC有所了解的人肯定接触过HandlerInterceptor拦截器，HandlerInterceptor接口给我们提供了3个方法：

（1）preHandle: 在执行controller处理之前执行，返回值为boolean ,返回值为true时接着执行postHandle和afterCompletion，如果我们返回false则中断执行

（2）postHandle:在执行controller的处理后，在ModelAndView处理前执行

（3）afterCompletion ：在DispatchServlet执行完ModelAndView之后执行

源码如下：

**[java]** [view plain](http://blog.csdn.net/qq924862077/article/details/53524507) [copy](http://blog.csdn.net/qq924862077/article/details/53524507)

[print?](http://blog.csdn.net/qq924862077/article/details/53524507)

1. **public** **interface** HandlerInterceptor {

4. /\*\*
5. \* preHandle方法是进行处理器拦截用的，顾名思义，该方法将在Controller处理之前进行调用，SpringMVC中的Interceptor拦截器是链式的，可以同时存在
6. \* 多个Interceptor，然后SpringMVC会根据声明的前后顺序一个接一个的执行，而且所有的Interceptor中的preHandle方法都会在
7. \* Controller方法调用之前调用。SpringMVC的这种Interceptor链式结构也是可以进行中断的，这种中断方式是令preHandle的返
8. \* 回值为false，当preHandle的返回值为false的时候整个请求就结束了。
9. \*/
10. **boolean** preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
11. **throws** Exception;

14. /\*\*
15. \* 这个方法只会在当前这个Interceptor的preHandle方法返回值为true的时候才会执行。postHandle是进行处理器拦截用的，它的执行时间是在处理器进行处理之
16. \* 后，也就是在Controller的方法调用之后执行，但是它会在DispatcherServlet进行视图的渲染之前执行，也就是说在这个方法中你可以对ModelAndView进行操
17. \* 作。这个方法的链式结构跟正常访问的方向是相反的，也就是说先声明的Interceptor拦截器该方法反而会后调用，这跟Struts2里面的拦截器的执行过程有点像，
18. \* 只是Struts2里面的intercept方法中要手动的调用ActionInvocation的invoke方法，Struts2中调用ActionInvocation的invoke方法就是调用下一个Interceptor
19. \* 或者是调用action，然后要在Interceptor之前调用的内容都写在调用invoke之前，要在Interceptor之后调用的内容都写在调用invoke方法之后。
20. \*/
21. **void** postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView)
22. **throws** Exception;

25. /\*\*
26. \* 该方法也是需要当前对应的Interceptor的preHandle方法的返回值为true时才会执行。该方法将在整个请求完成之后，也就是DispatcherServlet渲染了视图执行，
27. \* 这个方法的主要作用是用于清理资源的，当然这个方法也只能在当前这个Interceptor的preHandle方法的返回值为true时才会执行。
28. \*/
29. **void** afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex)
30. **throws** Exception;
32. }

简单实现一个拦截器，几乎并没有做任何实现，可以看之前写的博客[springMVC源码分析--国际化实现Session和Cookie（二）](http://blog.csdn.net/qq924862077/article/details/52918455)中LanguageInterceptor的详细实现。

**[java]** [view plain](http://blog.csdn.net/qq924862077/article/details/53524507) [copy](http://blog.csdn.net/qq924862077/article/details/53524507)

[print?](http://blog.csdn.net/qq924862077/article/details/53524507)

1. **public** **class** MyInterceptor **implements** HandlerInterceptor {
2. /\*\*
3. \* @Title: preHandle
4. \* @Description: 在执行controller之前运行
5. \* @param request
6. \* @param response
7. \* @param handler
8. \* @return
9. \* @throws Exception
10. \*/
11. **public** **boolean** preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) **throws** Exception {
12. **return** **true**;
13. }
14. /\*\*
15. \* @Title: postHandle
16. \* @Description: 在执行完controller之后，ModelAndView渲染之前开始运行
17. \* @param request
18. \* @param response
19. \* @param handler
20. \* @param modelAndView
21. \* @throws Exception
22. \*/
23. **public** **void** postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView)
24. **throws** Exception {
25. }
26. /\*\*
27. \* @Title: afterCompletion
28. \* @Description: 该方法将在整个请求完成之后，也就是DispatcherServlet渲染了视图后执行
29. \* @param request
30. \* @param response
31. \* @param handler
32. \* @param ex
33. \* @throws Exception
34. \*/
35. **public** **void** afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex)
36. **throws** Exception {
37. }
39. }

SpringMVC还提供了HandlerInterceptorAdapter 其是抽象类，也是HandlerInterceptor的子类，在实现了HandlerInterceptor的三个函数后还增加了一个函数。  
（1）preHandle: 在执行controller处理之前执行，返回值为boolean ,返回值为true时接着执行postHandle和afterCompletion，如果我们返回false则中断执行  
（2）postHandle:在执行controller的处理后，在ModelAndView处理前执行  
（3）afterCompletion ：在DispatchServlet执行完ModelAndView之后执行  
（4）afterConcurrentHandlingStarted：这个方法会在Controller方法异步执行时开始执行，而Interceptor的postHandle方法则是需要等到Controller的异步执行完才能执行，只要继承这个类并实现其方法就可以了。

**[java]** [view plain](http://blog.csdn.net/qq924862077/article/details/53524507) [copy](http://blog.csdn.net/qq924862077/article/details/53524507)

[print?](http://blog.csdn.net/qq924862077/article/details/53524507)

1. **public** **abstract** **class** HandlerInterceptorAdapter **implements** AsyncHandlerInterceptor {
2. /\*\*
3. \* This implementation always returns {@code true}.
4. \*/
5. @Override
6. **public** **boolean** preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
7. **throws** Exception {
8. **return** **true**;
9. }
10. /\*\*
11. \* This implementation is empty.
12. \*/
13. @Override
14. **public** **void** postHandle(
15. HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView)
16. **throws** Exception {
17. }
18. /\*\*
19. \* This implementation is empty.
20. \*/
21. @Override
22. **public** **void** afterCompletion(
23. HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex)
24. **throws** Exception {
25. }
26. /\*\*
27. \* This implementation is empty.
28. \*/
29. @Override
30. **public** **void** afterConcurrentHandlingStarted(
31. HttpServletRequest request, HttpServletResponse response, Object handler)
32. **throws** Exception {
33. }
35. }

AsyncHandlerInterceptor也是一个接口，提供了afterConcurrentHandlingStarted方法定义

**[java]** [view plain](http://blog.csdn.net/qq924862077/article/details/53524507) [copy](http://blog.csdn.net/qq924862077/article/details/53524507)

[print?](http://blog.csdn.net/qq924862077/article/details/53524507)

1. **public** **interface** AsyncHandlerInterceptor **extends** HandlerInterceptor {
3. /\*\*
4. \* Called instead of {@code postHandle} and {@code afterCompletion}, when
5. \* the a handler is being executed concurrently.
6. \* <p>Implementations may use the provided request and response but should
7. \* avoid modifying them in ways that would conflict with the concurrent
8. \* execution of the handler. A typical use of this method would be to
9. \* clean up thread-local variables.
10. \*
11. \* @param request the current request
12. \* @param response the current response
13. \* @param handler the handler (or {@link HandlerMethod}) that started async
14. \* execution, for type and/or instance examination
15. \* @throws Exception in case of errors
16. \*/
17. **void** afterConcurrentHandlingStarted(
18. HttpServletRequest request, HttpServletResponse response, Object handler)
19. **throws** Exception;
21. }