[**Java过滤器，SpringMVC拦截器之间的一顺序点关系**](http://www.cnblogs.com/dreamroute/p/4198087.html)

**参考：**[**https://www.cnblogs.com/dreamroute/p/4198087.html**](https://www.cnblogs.com/dreamroute/p/4198087.html)

　　由于最近做的项目中有一部分是接口远程调用，用到了接入权限和业务权限的鉴定，需要采用SpringMVC的拦截器，以前用Struts2的时候用过拦截器，而SpringMVC的拦截器功能之前没研究过，所以这次来稍微研究一下，得出的结论是SpringMVC的拦截器和Struts2的拦截器原理几乎是一模一样的，都是利用反射功能实现动态代理。

　　因为过滤器和拦截器有很多相似甚至相同的地方，因为很多时候二者都能达到相同的能力。所以也重新看了下过滤器。

　　过滤器和拦截器的区别，百度了一下：

　　①拦截器是基于java的反射机制的，而过滤器是基于函数回调。  
　　②拦截器不依赖与servlet容器，过滤器依赖与servlet容器。  
　　③拦截器只能对action请求起作用，而过滤器则可以对几乎所有的请求起作用。  
　　④拦截器可以访问action上下文、值栈里的对象，而过滤器不能访问。  
　　⑤在action的生命周期中，拦截器可以多次被调用，而过滤器只能在容器初始化时被调用一次。

　　6.拦截器可以获取IOC容器中的各个bean，而过滤器就不行，这点很重要，在拦截器里注入一个service，可以调用业务逻辑。

　　写了点测试代码，顺便整理一下思路，搞清楚这几者之间的顺序：

　　1.过滤器是JavaEE标准，采用函数回调的方式进行。是在请求进入容器之后，还未进入Servlet之前进行预处理，并且在请求结束返回给前端这之间进行后期处理。

@Override

public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {

System.out.println("before...");

chain.doFilter(request, response);

System.out.println("after...");

}

　　chain.doFilter(request, response);这个方法的调用作为分水岭。事实上调用Servlet的doService()方法是在chain.doFilter(request, response);这个方法中进行的。

　　2.拦截器是被包裹在过滤器之中的。

@Override

public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {

System.out.println("preHandle");

return true;

}

@Override

public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView) throws Exception {

System.out.println("postHandle");

}

@Override

public void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex) throws Exception {

System.out.println("afterCompletion");

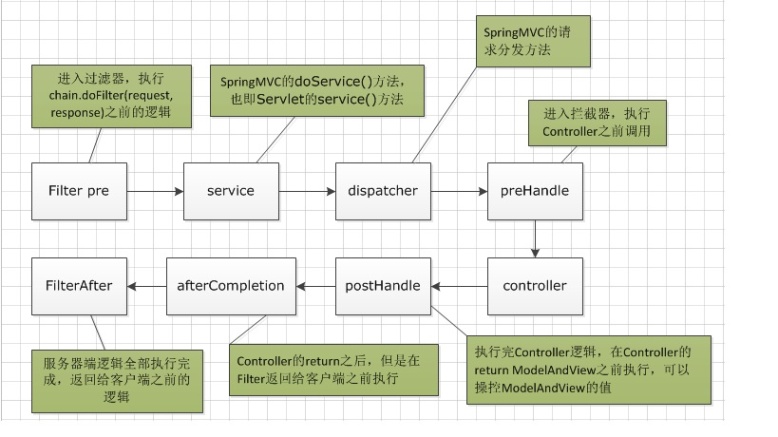
}

　　a.preHandle()这个方法是在过滤器的chain.doFilter(request, response)方法的前一步执行，也就是在 [System.out.println("before...")][chain.doFilter(request, response)]之间执行。

　　b.preHandle()方法之后，在return ModelAndView之前进行，可以操控Controller的ModelAndView内容。

　　c.afterCompletion()方法是在过滤器返回给前端前一步执行，也就是在[chain.doFilter(request, response)][System.out.println("after...")]之间执行。

　　3.SpringMVC的机制是由同一个Servlet来分发请求给不同的Controller，其实这一步是在Servlet的service()方法中执行的。所以过滤器、拦截器、service()方法，dispatc()方法的执行顺序应该是这样的，大致画了个图：其实非常好测试，自己写一个过滤器，一个拦截器，然后在这些方法中都加个断点，一路F8下去就得出了结论。



　　总结：拦截器功在对请求权限鉴定方面确实很有用处，在我所参与的这个项目之中，第三方的远程调用每个请求都需要参与鉴定，所以这样做非常方便，而且他是很独立的逻辑，这样做让业务逻辑代码很干净。和框架的其他功能一样，原理很简单，使用起来也很简单，大致看了下SpringMVC这一部分的源码，其实还是比较容易理解的。

　　我们项目中仅仅用到了preHandle这个方法，而未用其他的，框架提供了一个已经实现了拦截器接口的适配器类HandlerInterceptorAdapter，继承这个类然后重写一下需要用到的方法就行了，可以少几行代码，这种方式Java中很多地方都有体现。

　　参考了一下这个帖子：http://haohaoxuexi.iteye.com/blog/1750680