RESTful API的拦截：记录所有的服务的处理时间

1. 过滤器：可以获取到http请求的信息，不能获取真正处理请求方法的信息

```
/**
 * 记录所有的服务的处理时间
 * @author 江小明
 *
 */
//@Component 注释掉了就成了一个普通的类，相当于是一个第三方的过滤器，加到过滤器链上
public class TimeFilter implements Filter {

 @Override
  public void init(FilterConfig filterConfig) throws ServletException {
  System.out.println("time filter init");
  }

  @Override
  public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
  throws IOException, ServletException {
  System.out.println("time filter start");
  long start = new Date().getTime();
  //调用后续的过滤器
  chain.doFilter(request, response);
  System.out.println("time filter 耗时:"+ (new Date().getTime() - start));
  System.out.println("time filter finish");
  }

  @Override
  public void destroy() {
  System.out.println("time filter destroy");
  }

}
```

```
@Configuration
public class WebConfig {

 //将timeFilter加到过滤器链上 和在timeFilter上加Component注解是一样的
 @Bean
 public FilterRegistrationBean timeFilter() {
```

```
7   FilterRegistrationBean registrationBean = new FilterRegistrationBean();
8
9   TimeFilter timeFilter = new TimeFilter();
10   registrationBean.setFilter(timeFilter);
11   List<String> url = new ArrayList<String>();//配置这个过滤器在哪些url下起作
用
12   url.add("/*");//所有路径
13   registrationBean.setUrlPatterns(url);
14   return registrationBean;
15   }
16  }
```

2. 拦截器：可以获取到http请求的信息，也可以获取真正处理请求方法的信息，但是不能获取到那个方法的值

```
1  /**
2   * 时间拦截器
3   * @author 江小明
4   *
5   */
6  @Component //这个注解并不能使这个拦截器起作用，必须还要配置
7  public class TimeInterceptor implements HandlerInterceptor {
8
9   /**
10   * controller方法之前
11   */
12   @Override
13   public boolean preHandle(HttpServletRequest request, HttpServletRespons
e response, Object handler)
14   throws Exception {
15   System.out.println("preHandle");
16
17   System.out.println(((HandlerMethod)handler).getBean().getClass().getNam
e());
18   System.out.println(((HandlerMethod)handler).getMethod().getName());
19
20   request.setAttribute("startTime", new Date().getTime());
21   return true; //是否调用controller里的方法
22   }
23   /**
24   * 执行controller方法时被调用，有异常则不调用
```

```
25   */
26   @Override
27   public void postHandle(HttpServletRequest request, HttpServletResponse
response, Object handler,
28   ModelAndView modelAndView) throws Exception {
29   System.out.println("postHandle");
30   long start = (long) request.getAttribute("startTime");
31   System.out.println("time interceptor 耗时:"+ (new Date().getTime() - sta
rt));
32   }
33   /**
34   * 执行controller方法之后调用，有异常也被调用
35   */
36   @Override
37   public void afterCompletion(HttpServletRequest request, HttpServletResp
onse response, Object handler, Exception ex)
38   throws Exception {
39   System.out.println("afterCompletion");
40   long start = (long) request.getAttribute("startTime");
41   System.out.println("time interceptor 耗时:"+ (new Date().getTime() - sta
rt));
42   System.out.println("ex is "+ex);
43   }
44
45  }
```

```
1  @Configuration
2  public class WebConfig extends WebMvcConfigurerAdapter{
3   @Autowired
4   private TimeInterceptor interceptor;
5
6   @Override
7   public void addInterceptors(InterceptorRegistry registry) {
8   registry.addInterceptor(interceptor);
9   }
10  }
```

拦截器局限性：handler获取不到参数的值