目录

[Listener 监听器 3](#_Toc29225)

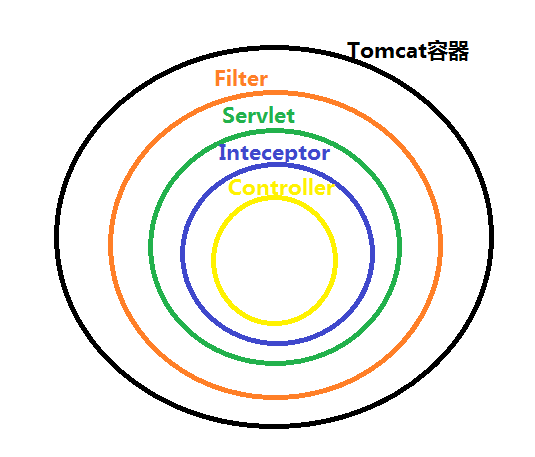
[ 一类:监听三个域对象的创建和销毁的监听器 3](#_Toc28070)

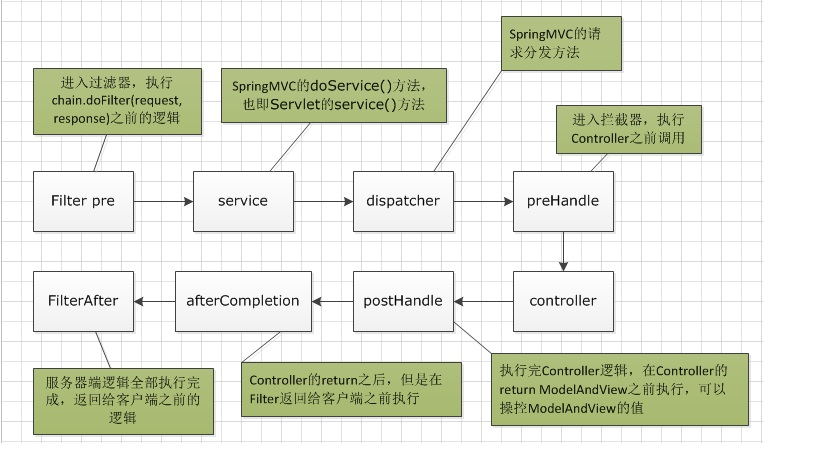
[ 二类:监听三个域对象的属性变更的监听器.(属性添加,属性移除,属性替换) 4](#_Toc24495)

[ 三类:监听HttpSession对象中的JavaBean的状态的改变.(绑定,解除绑定,钝化和活化)2个 4](#_Toc28232)

# 序

1. Listener 监听器
2. Filter过滤器
3. Servlet
4. Inteceptor 拦截器
5. Controller 控制器





# Listener 监听器

javaweb的监听器:监听ServletContext,HttpSession,ServletRequest三个域对象状态

​ 事件源和监听器绑定的过程:通过配置web.xml完成

Servlet中提供了8个监听器

## 一类:监听三个域对象的创建和销毁的监听器

|  |  |
| --- | --- |
| 对象类型 | 对应的监听器 |
| ServletContext | ServletContextListener |
| HttpSession | HttpSessionListener |
| HttpServletRequest | ServletRequestListener |

## 二类:监听三个域对象的属性变更的监听器.(属性添加,属性移除,属性替换)

|  |  |
| --- | --- |
| 对象类型 | 对应的监听器 |
| ServletContext | ServletContextAttributeListener |
| HttpServletRequest | ServletRequestAttributeListener |
| HttpSession | HttpSessionAttributeListener |

## 三类:监听HttpSession对象中的JavaBean的状态的改变.(绑定,解除绑定,钝化和活化)2个

|  |  |
| --- | --- |
| 对象类型 | 对应的监听器 |
| HttpSession | HttpSessionBindingListener(绑定,解除绑定) |
| HttpSession | HttpSessionActivationListener(钝化和活化) |

# Filter 过滤器 和 Interceptor 拦截器

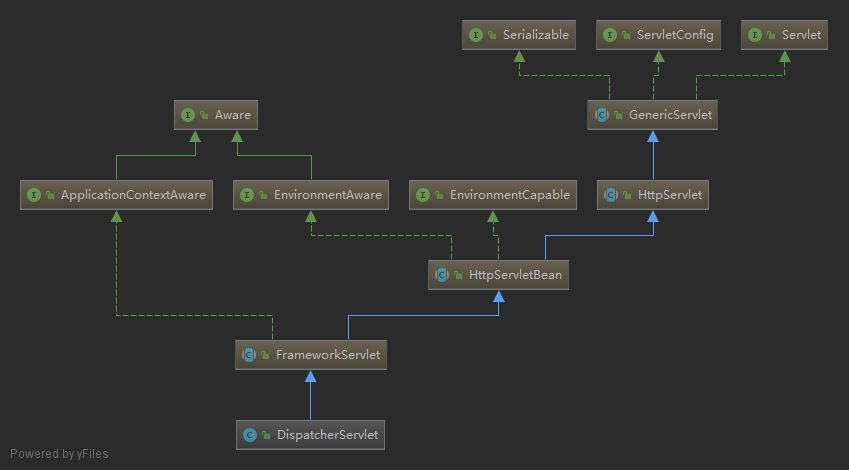
Spring的Interceptor(拦截器)与Servlet的Filter有相似之处，比如二者都是AOP编程思想的体现，都能实现权限检查、日志记录等。不同的是：

|  |  |  |
| --- | --- | --- |
| Filter | Interceptor | 总结 |
| Filter 接口定义在 javax.servlet 包中 | 接口 HandlerInterceptor 定义在org.springframework.web.servlet 包中 |  |
| Filter 定义在 web.xml 中 |  |  |
| Filter在只在 Servlet 前后起作用。Filters 通常将 请求和响应（request/response） 当做黑盒子，Filter 通常不考虑servlet 的实现。 | 拦截器能够深入到方法前后、异常抛出前后等，因此拦截器的使用具有更大的弹性。允许用户介入（hook into）请求的生命周期，在请求过程中获取信息，Interceptor 通常和请求更加耦合。 | 在Spring构架的程序中，要优先使用拦截器。几乎所有 Filter 能够做的事情， interceptor 都能够轻松的实现 |
| Filter 是 Servlet 规范规定的。 | 而拦截器既可以用于Web程序，也可以用于Application、Swing程序中。 | 使用范围不同 |
| Filter 是在 Servlet 规范中定义的，是 Servlet 容器支持的。 | 而拦截器是在 Spring容器内的，是Spring框架支持的。 | 规范不同 |
| Filter 不能够使用 Spring 容器资源 | 拦截器是一个Spring的组件，归Spring管理，配置在Spring文件中，因此能使用Spring里的任何资源、对象，例如 Service对象、数据源、事务管理等，通过IoC注入到拦截器即可 | Spring 中使用 interceptor 更容易 |
| Filter 是被 Server(like Tomcat) 调用 | Interceptor 是被 Spring 调用 | 因此 Filter 总是优先于 Interceptor 执行 |

# SpringMVC 与 Servlet

Springmvc 是通过一个Servlet来处理请求的

org.springframework.web.servlet.DispatcherServlet



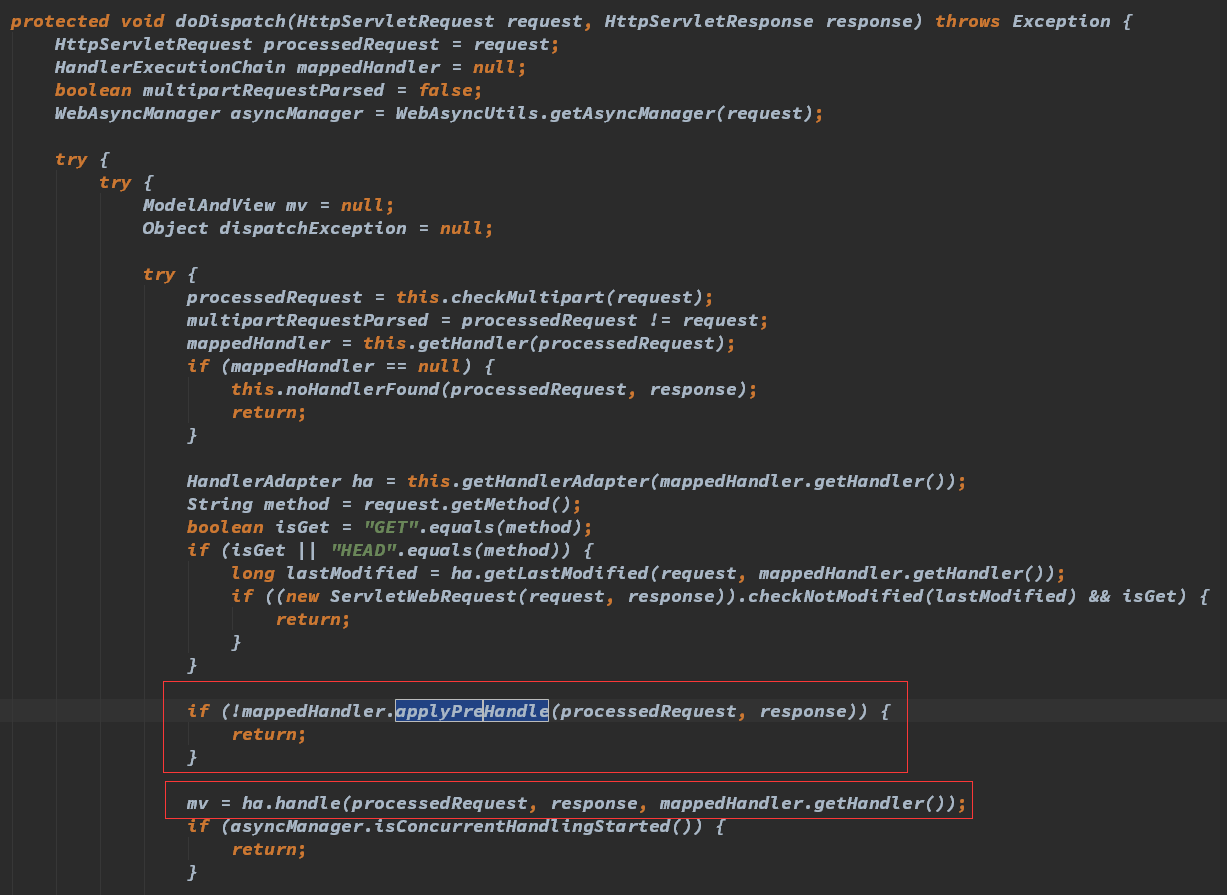
（idea Ctrl + Alt + u 可以显示类的继承关系）

//通过HandlerMapping，将请求映射到处理器

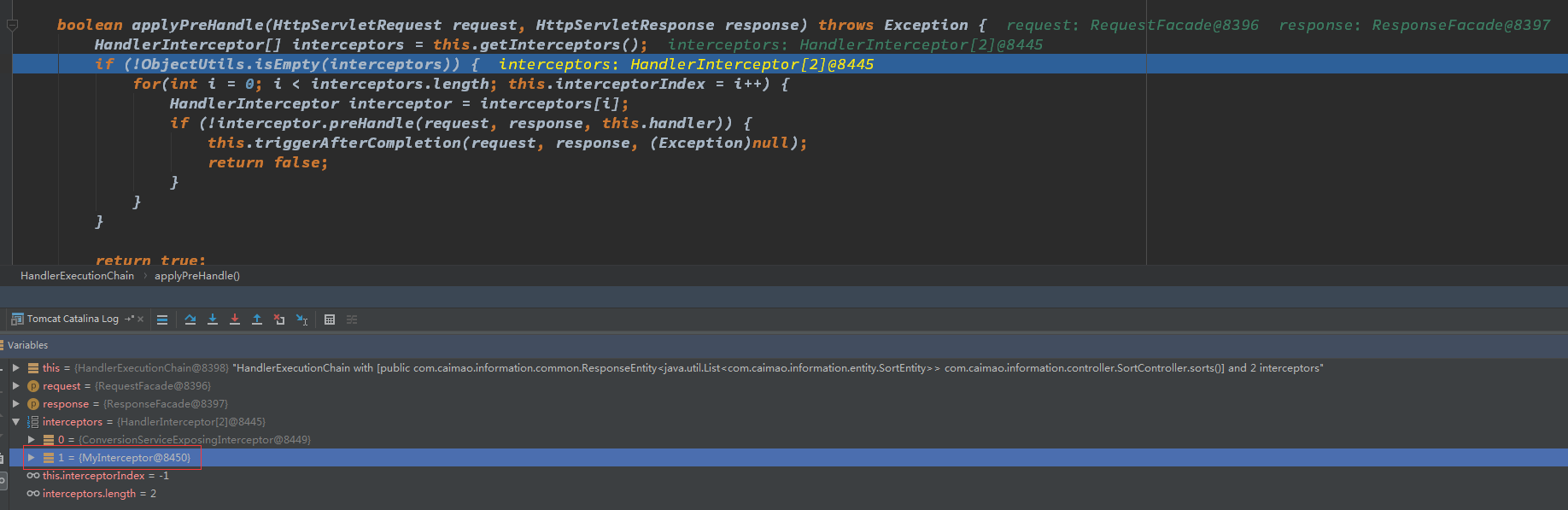
this.initHandlerMappings(context);

将找到的HandlerMapping全部存到DispatcherServlet的this.handlerMappings中

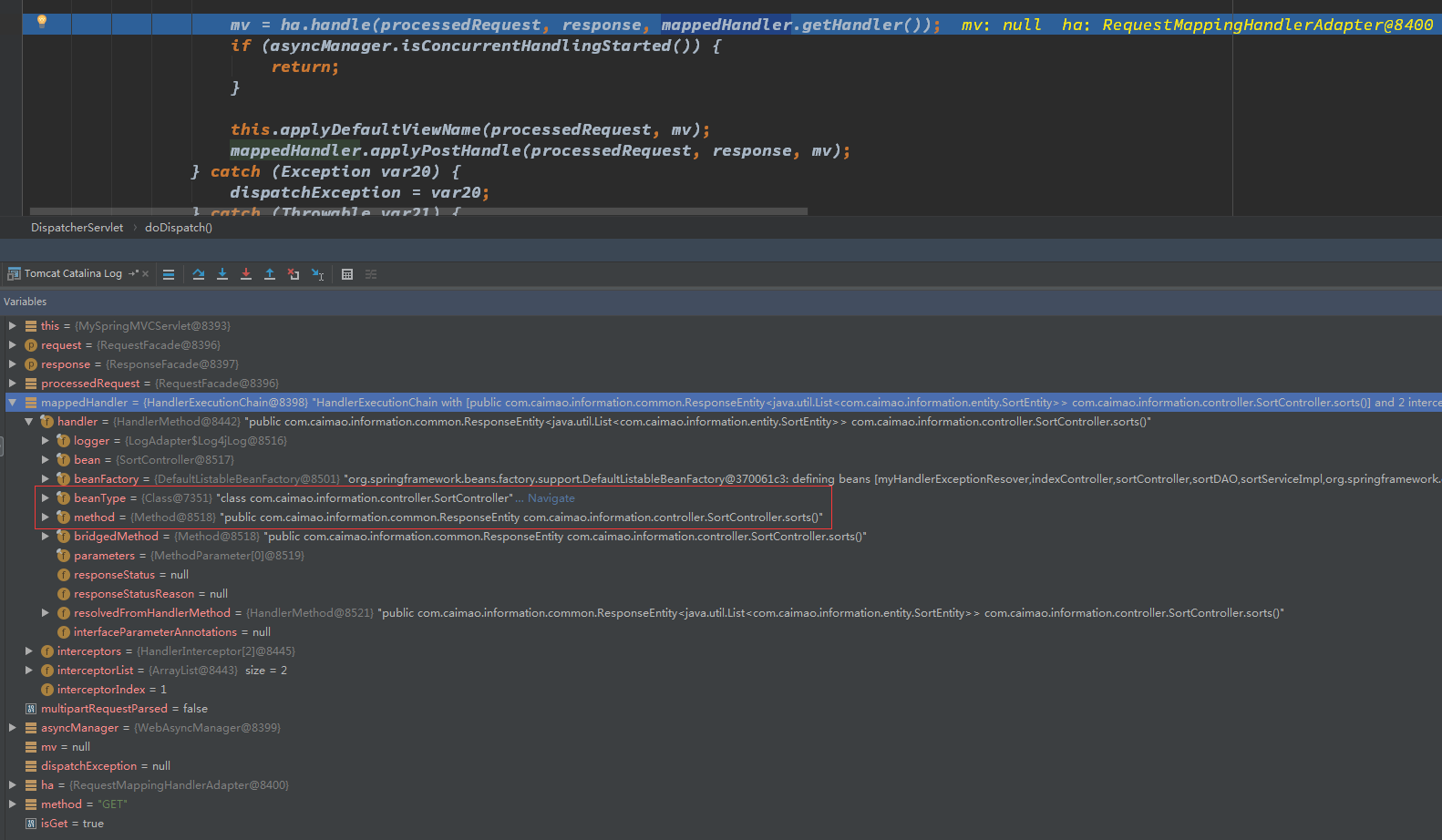
在执行控制器前



先执行Interceptor 拦截器



执行具体的controller 控制器



//如果执行过程中遇到异常将交给HandlerExceptionResolver来解析

this.initHandlerExceptionResolvers(context);

