## 过滤器

依赖于servlet容器。在实现上基于函数回调，可以对几乎所有请求进行过滤，但是缺点是一个过滤器实例只能在容器初始化时调用一次。使用过滤器的目的是用来做一些过滤操作，获取我们想要获取的数据，比如：在过滤器中修改字符编码；在过滤器中修改HttpServletRequest的一些参数，包括：过滤低俗文字、危险字符等。

Java中的Filter并不是一个标准的Servlet ，它不能处理用户请求，也不能对客户端生成响应。 主要用于对HttpServletRequest 进行预处理，也可以对HttpServletResponse 进行后处理，是个典型的处理链。完整的流程是：Filter对用户请求进行预处理，接着将请求交给Servlet进行处理并生成响应，最后Filter再对服务器响应进行后处理。在HttpServletRequest 到达Servlet 之前，拦截客户的HttpServletRequest。根据需要检查HttpServletRequest ，也可以修改HttpServletRequest 头和数据。在HttpServletResponse 到达客户端之前，拦截HttpServletResponse。根据需要检查HttpServletResponse ，可以修改HttpServletResponse 头和数据。

Filter随web应用的启动而启动，只初始化一次，随web应用的停止而销毁。  
1.启动服务器时加载过滤器的实例，并调用init()方法来初始化实例；  
2.每一次请求时都只调用方法doFilter()进行处理；  
3.停止服务器时调用destroy()方法，销毁实例。

## 拦截器

依赖于web框架，在SpringMVC中就是依赖于SpringMVC框架。在实现上基于Java的反射机制，属于面向切面编程（AOP）的一种运用。由于拦截器是基于web框架的调用，****拦截器可以调用IOC容器中的各种依赖，而过滤器不能，****因此可以使用Spring的依赖注入（DI）进行一些业务操作，同时一个拦截器实例在一个controller生命周期之内可以多次调用。但是缺点是只能对controller请求进行拦截，对其他的一些比如直接访问静态资源的请求则没办法进行拦截处理。

spring mvc中的Interceptor可以理解为是Spring MVC框架对AOP的一种实现方式。一般简单的功能又是通用的,每个请求都要去处理的,比如判断token是否失效可以使用spring mvc的HanlderInterceptor, 复杂的,比如缓存,需要高度自定义的就用spring aop。一般来说service层更多用spring aop，controller层有必要用到request和response的时候，可以用拦截器。

spring mvc中的Interceptor拦截请求是通过HandlerInterceptor来实现的。所以HandlerInteceptor拦截器只有在Spring Web MVC环境下才能使用。在SpringMVC中定义一个拦截器主要有两种方式，第一种方式是要实现Spring的HandlerInterceptor接口，或者是其它实现了HandlerInterceptor接口的类，比如HandlerInterceptorAdapter。第二种方式是实现WebRequestInterceptor接口，或者其它实现了WebRequestInterceptor的类。

HandlerInterceptor接口中定义了三个方法preHandle, postHandle, 和afterCompletion：

* preHandle：预处理回调方法，实现处理器的预处理（如登录检查），返回值：true表示继续流程（如调用下一个拦截器或处理器），false表示流程中断（如登录检查失败），不会继续调用其他的拦截器或处理器，此时我们需要通过response来产生响应。
* postHandle：后处理回调方法，实现处理器的后处理（但在渲染视图之前），此时我们可以通过modelAndView（模型和视图对象）对模型数据进行处理或对视图进行处理，modelAndView也可能为null。
* afterCompletion：整个请求处理完毕回调方法，即在视图渲染完毕时回调。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。这个方法的主要作用是用于进行资源清理工作的，如性能监控中我们可以在此记录结束时间并输出消耗时间。

## Filter和HandlerInterceptor的区别

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Controller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Controller上下文、值栈里的对象，而过滤器不能访问。  
5.在Controller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。

## Filter和HandlerInterceptor的执行顺序

