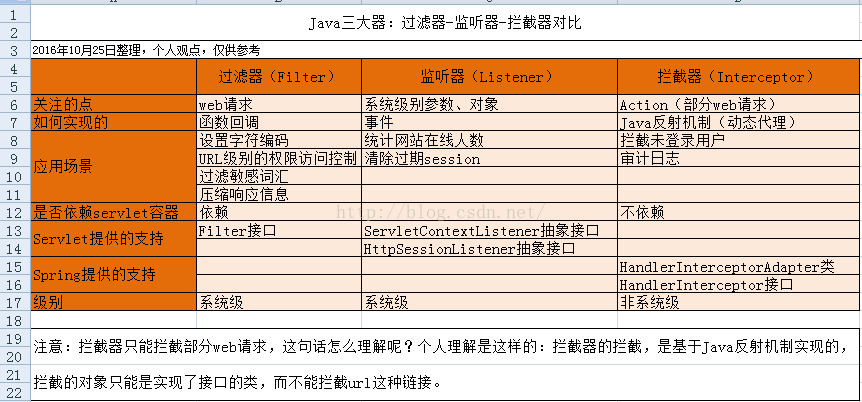
拦截器



# ****springboot拦截器配置****

## Interceptor 作用

1. 日志记录：记录请求信息的日志，以便进行信息监控、信息统计、计算 PV（Page View）等；
2. 权限检查：如登录检测，进入处理器检测是否登录；
3. 性能监控：通过拦截器在进入处理器之前记录开始时间，在处理完后记录结束时间，从而得到该请求的处理时间。（反向代理，如 Apache 也可以自动记录）
4. 通用行为：读取 Cookie 得到用户信息并将用户对象放入请求，从而方便后续流程使用，还有如提取 Locale、Theme 信息等，只要是多个处理器都需要的即可使用拦截器实现

## 自定义 Interceptor

需要自定义 **Interceptor** 的话必须实现 org.springframework.web.servlet.HandlerInterceptor接口或继承 org.springframework.web.servlet.handler.HandlerInterceptorAdapter类，并且需要重写下面下面 3 个方法：

1. preHandler(HttpServletRequest request, HttpServletResponse response, Object handler) 方法在请求处理之前被调用。该方法在 Interceptor 类中最先执行，用来进行一些前置初始化操作或是对当前请求做预处理，也可以进行一些判断来决定请求是否要继续进行下去。该方法的返回至是 Boolean 类型，当它返回 false 时，表示请求结束，后续的 Interceptor 和 Controller 都不会再执行；当它返回为 true 时会继续调用下一个 Interceptor 的 preHandle 方法，如果已经是最后一个 Interceptor 的时候就会调用当前请求的 Controller 方法。
2. postHandler(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView) 方法在当前请求处理完成之后，也就是 Controller 方法调用之后执行，但是它会在 DispatcherServlet 进行视图返回渲染之前被调用，所以我们可以在这个方法中对 Controller 处理之后的 ModelAndView 对象进行操作。
3. afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handle, Exception ex) 方法需要在当前对应的 Interceptor 类的 preHandle 方法返回值为 true 时才会执行。顾名思义，该方法将在整个请求结束之后，也就是在 DispatcherServlet 渲染了对应的视图之后执行。此方法主要用来进行资源清理。

<https://juejin.cn/post/6844904020675559432>

# AOP （Aspect）

## AOP概述

  AOP是Aspect-Oriented Programming，即为面向（切面）方面编程。Aspect是一种新的模块化机制，用来描述分散在对象、类或函数中的横切关注点。从关注点中分离出横切关注点是面向切面的程序设计核心概念。

分离关注点使得解决特定领域问题的代码从业务逻辑中独立出来，业务逻辑代码不需要再包含针对特定领域问题代码的调用，比如一些公用模块的日志、安全等代码。

代码通过切面抽离，更加整齐和清晰，将重复的代码抽取出来单独的进行维护，在需要使用的时候，统一调用这些公共模块的代码，这样一个类就是一个基本的模块，方便统一维护和扩展更新。

 AOP就是为业务实现提供了切面注入的一种机制，将定义好的切面通过切入点（pointcut）在业务逻辑中进行绑定。

比如SpringBoot微服务中的所有controller层需要对http请求进行一些常规日志的打印，如果每次在controller进行打印，代码就会冗余，如果说将这些公共代码进行封装，也需要每一个controller类进行调用，所以AOP出现的恰到好处，这时候引入AOP对http相关的日志逻辑进行统一管理编写代码，不需要controller层进行调用，只需要创建一个切面，并通过切入点绑定controller即可。

## AOP相关术语

 切面（Aspect）：是指横切多个对象的关注点的一个模块化，事务管理就是J2EE应用中横切关注点的很好示例。在Spring AOP中，切面通过常规类（基本模式方法）或者通过使用了注解@Aspect的常规类来实现。

 连接点（Joint point）:是指在程序执行期间的一个点，比如某个方法的执行或者是某个异常的处理。在Spring AOP中，一个连接点往往代表的是一个方法执行。

 通知（Advice）：是指切面在某个特殊连接点上执行的动作。通知有不同类型，包括"around","before"和"after"通知。许多AOP框架包括Spring，将通知建模成一个拦截器，并且围绕连接点维持一个拦截器链。

 切入点（Pointcut）：是指匹配连接点的一个断言。通知是和一个切入点表达式关联的，并且在任何被切入点匹配的连接点上运行（举例，使用特定的名字执行某个方法）。AOP的核心就是切入点表达式匹配连接点的思想。Spring默认使用AspectJ切入点表达式语言

 引入（Introduction）：代表了对一个类型额外的方法或者属性的声明。Spring AOP允许引入新接口到任何被通知对象（以及一个对应实现）。比如，可以使用一个引入去使一个bean实现IsModified接口，从而简化缓存机制。（在AspectJ社区中，一个引入也称为一个inter-type declaration类型间声明）

 目标对象（Target object）：是指被一个或多个切面通知的那个对象。也指被通知对象（"advised object"），由于Spring AOP是通过运行时代理事项的，这个目标对象往往是一个代理对象。

 AOP 代理（AOP proxy）：是指通过AOP框架创建的对象，用来实现切面合约的（执行通知方法等等）。在Spring框架中，一个AOP代理是一个JDK动态代理或者是一个CGLIB代理。

 织入（Weaving）：将切面和其他应用类型或者对象连接起来，创骗一个被通知对象。这些可以在编译时（如使用AspectJ编译器）、加载时或者运行时完成。Spring AOP，比如其他纯Java AOP框架一般是在运行时完成织入。

## AOP Advice相关术语

* @Aspect ： 将一个 java 类定义为切面类。
* @Pointcut ：定义一个切入点，可以是一个规则表达式，比如下例中某个 package 下的所有函数，也可以是一个注解等。
* @Before ：在切入点开始处切入内容。
* @After ：在切入点结尾处切入内容。
* @AfterReturning ：在切入点 return 内容之后切入内容（可以用来对处理返回值做一些加工处理）。
* @Around ：在切入点前后切入内容，并自己控制何时执行切入点自身的内容。
* @AfterThrowing ：用来处理当切入内容部分抛出异常之后的处理逻辑。

## pom.xml引入

<**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-aop</**artifactId**>  
 <**version**>2.3.6.RELEASE</**version**>  
</**dependency**>

## 参考链接

https://juejin.cn/post/6844904121883951117

# 参考拦截器、过滤器 、监听器

<https://juejin.cn/post/6844903981886488590>

# git

下载地址：<https://gitforwindows.org/>

安装步骤：<https://blog.csdn.net/qq_43715354/article/details/108638061>

教程：https://blog.csdn.net/qq\_43715354/article/details/108638061

把代码上传的github上待完成