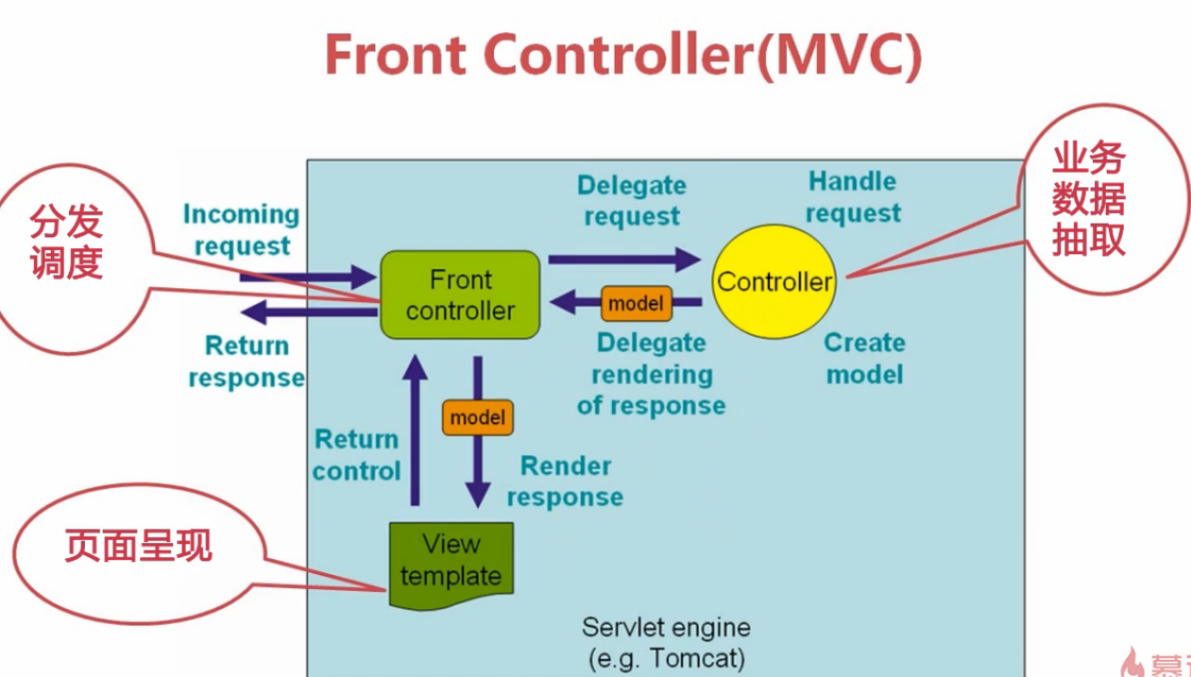
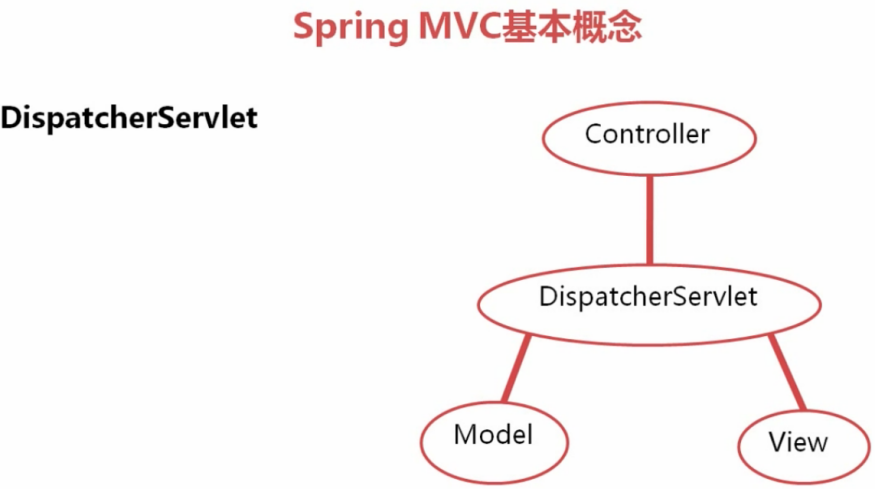
**一．Spring-MVC**



FC前端控制器：Spring开发团队共享给Java社区的一种MVC开发模式

**解析：首先用户端的请求通过HTTP协议到达FC前端控制器，FC了解这个请求应当被谁处理，所以这个请求代理给了controller控制器，controller了解业务逻辑的细节，因此调用了业务逻辑并生成了业务数据返回给FC，FC再将这些数据分发给业务视图View，业务视图来呈现最终的用户页面并返回给FC，FC再传递给浏览器端**

MVC的核心思想是讲业务数据的抽取同业务数据呈现相分离



Spring作为一种前端控制器的实现形式，DispatcherServlet就是他的FC前端控制器

前端【浏览器端】的请求正是通过DispatcherServlet进行了分发到达合适的Controller，来生产我们需要的业务数据Model，这个数据Model再通过DispatcherServlet传递给View进行最终的页面呈现，因此MVC进行业务逻辑和页面的分离就是通过DispatcherServlet实现的

Controller：调用业务逻辑生成Model的地方

HandlerAdapter：DispatcherServlet内部使用的一个类，其实就是Controller的一个表现形式，

SpringMVC中并没有一个interface叫做Controller，但是有一个annotation标签，这个标签是为了识别一个Controller而引入的，既然没有Controller这样一个interface类似于接口的东西，那么DispatcherServlet如何去识别Controller，答案就是Handler，在DispatcherServlet中最终调用的Controller是以Handler的形式出现的，那么什么是HandlerAdapter【适配器模式】：将各种不同类型的Handler适配成DispatcherServlet可以使用的Handler，这样DispatcherServlet就可以很轻松的调用控制器Controller

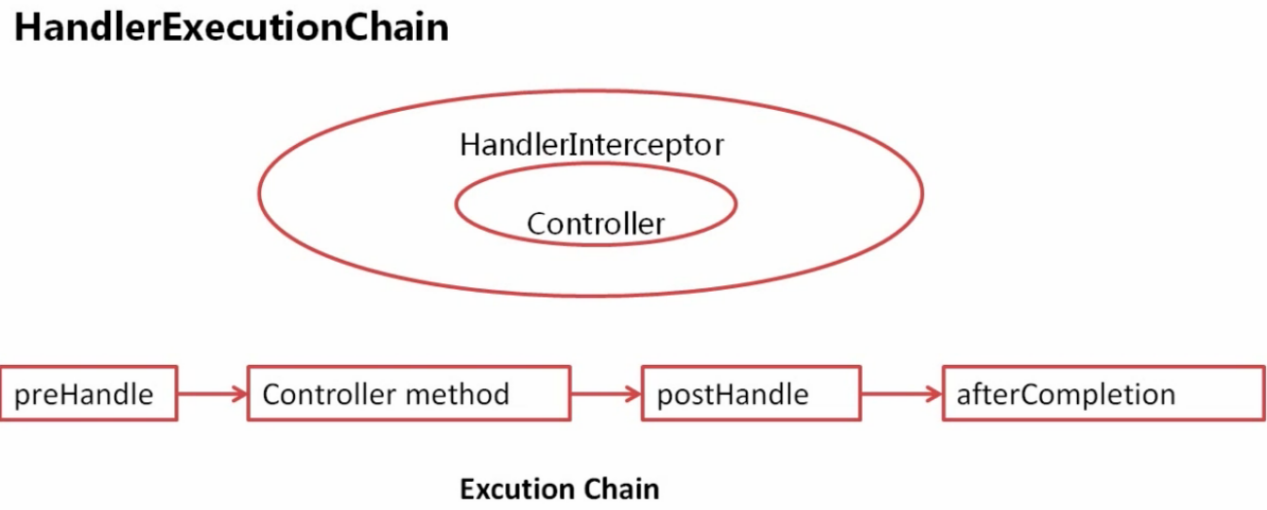
HandlerInterceptor:拦截器，需要被拦截对象两侧加控制模块，是一个接口

Handler：是DispatcherServlet调用Controller的一种中间过渡对象

HandlerMapping:DispatcherServlet前端控制器与Controller之间映射关系类，用来告诉DispatcherServlet某个请求到来之后由哪个Controller响应，HandlerInterceptor会对Controller加一些模块，这也是HandlerMapping需要考虑的，不管多复杂，结论：在HandlerMapping工作完毕之后，他可以给DispatcherServlet一个HandlerAdapter，这个HandlerAdapter内就包括了核心内容，某个具体的Controller的实例以及这个Controller被包裹的HandlerInterceptor

HandlerExcutionChain:HandlerMapping返回的东西包括HandlerAdapter和HandlerInterceptor

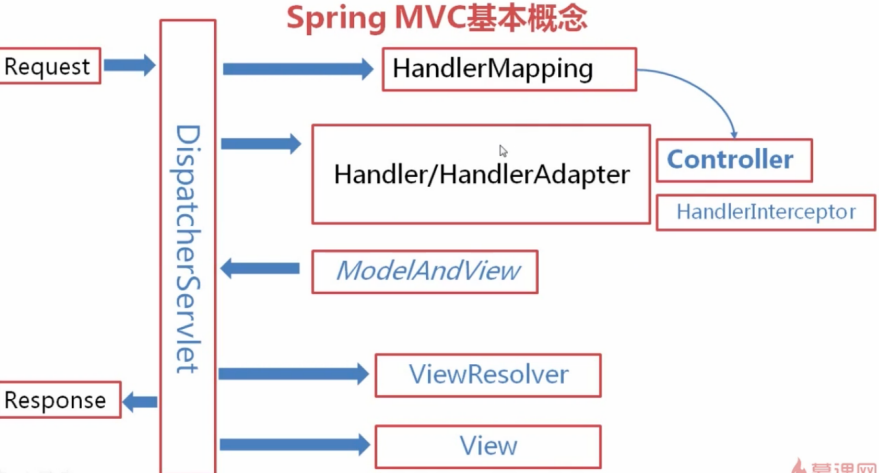
他们之间构成一个执行链条：通过反射实现



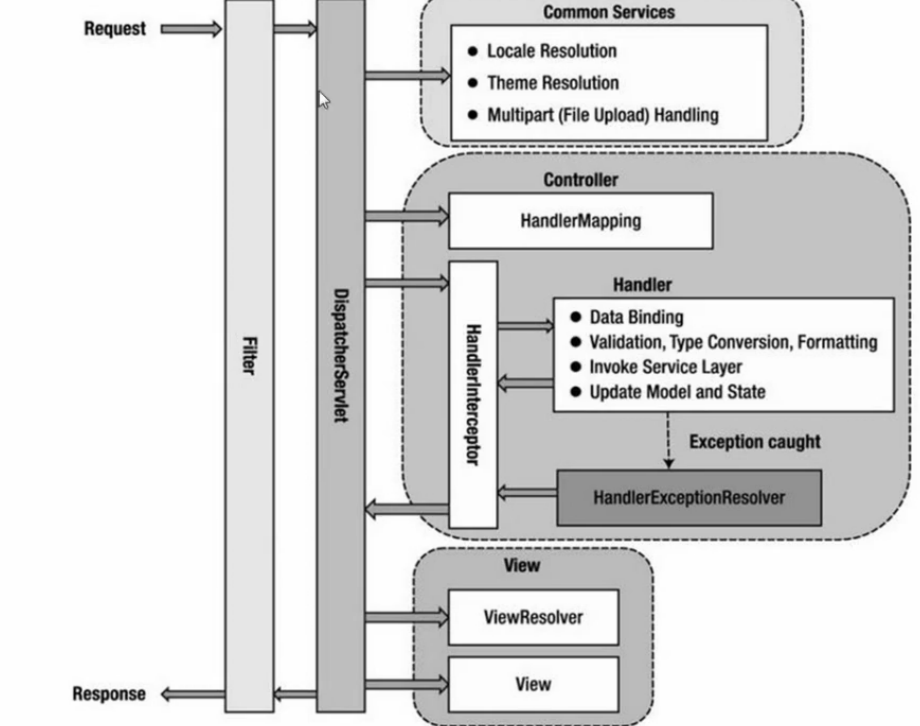
ModelAndView：SpringMVC对model的一种表现形式，也可以使用Map实现model的功能，不管controller中使用map还是model，DispatcherServlet都会将其统统转化为ModelAndView

ViewResolver：视图解析器，会通知DispatcherServlet使用哪个视图进行呈现，作用是根据配置找出需要的视图

View：页面



HandlerMapping找到了Controller和HandlerInterceptor生成了HandlerExcutionChain，作为一个Handler/HandlerAdapter返回给DispatcherServlet，接下来DispatcherServlet调用Handler/HandlerAdapter



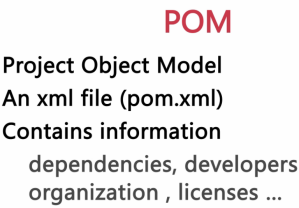
**Maven:一个构件管理工具【仓库】，解决项目环境管理**

利用一个xml文件管理整个项目的依赖管理



DM：依赖管理，提供直接或传递依赖管理，通过坐标实现

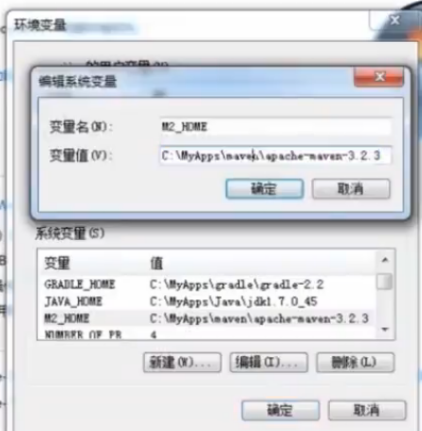
Coordinates坐标：groupId，artifactId，version，packaging四个坐标定位唯一一个产品

就是一个xml文件

**安装：**



配置环境：



修改cof目录下的setting文件，将该文件copy到C:\Users\ClearC\.m2下【maven升级后就不需要再手动修改配置文件了，因为将配置文件单独的放到了本地目录下】，打开此处的文件进行修改：

【此处不修改该项，记住默认软件库位置即可】

<!-- localRepository

| The path to the local repository maven will use to store artifacts.

|

| Default: ${user.home}/.m2/repository 【默认软件库】

<localRepository>/path/to/local/repo</localRepository>

-->

【镜像mirrors，指核心部分的副本，maven的核心仓库的副本就是mirror】

此处指向英国的仓库，降低核心仓库负载，提高访问成功率

<mirrors>

<mirror>

<id>UK</id>

<name>UK Central</name>

<url>http://uk.maven.org/maven2</url>

<mirrorOf>central</mirrorOf>

</mirror>

</mirrors>

阿里云仓库

<id>alimaven</id>  
<name>aliyun maven</name>  
<url>http://maven.aliyun.com/nexus/content/groups/public/</url>  
<mirrorOf>central</mirrorOf>   
</mirror>

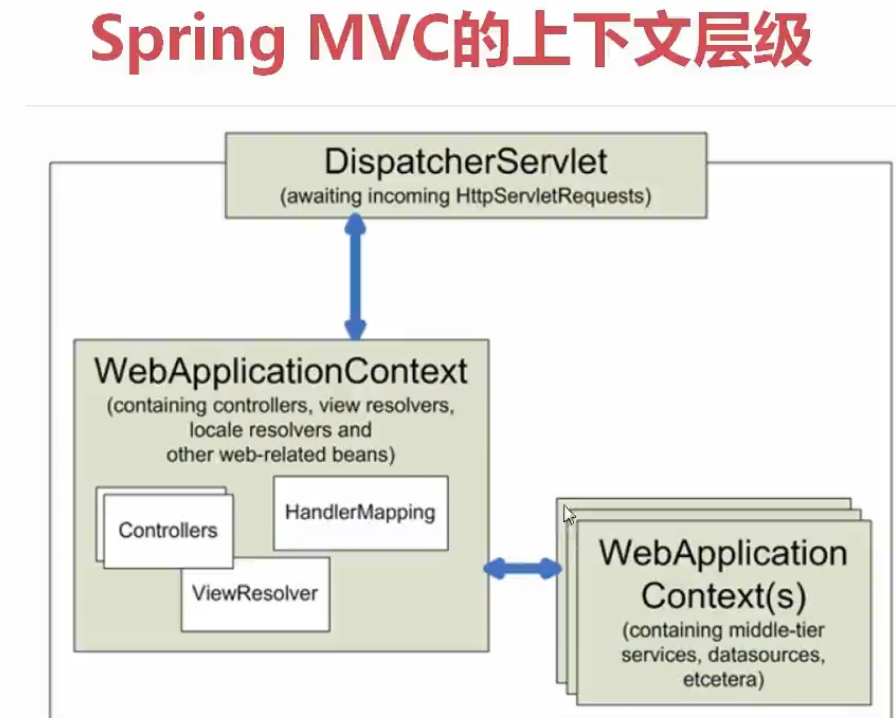
**OSChina**

<mirror><id>CN</id><name>OSChina Central</name> <url>http://maven.oschina.net/content/groups/public/</url> <mirrorOf>central</mirrorOf></mirror>

1. **Spring-MVC起步**

Web.Xml文件配置

<!-- Spring应用上下文， 理解层次化的ApplicationContext，spring/DispatcherServlet都有自己上下文的配置文件 -->



WebApplicationContext(s)

<!-- ContextLoaderListener上下文,提供了所有共享公共组件，不会局限在DispatcherServlet之中 -->

绑定Binding：将请求中的字段按照名字匹配的原则填入模型对象

玛德：就是将HTML表单中的name和后台的bean的属性绑定

1. **Spring-MVC-JSON**

JSON的出现让开发发生翻天覆地的变化，越来越多的系统采用单页面开发的模式，使前端和客户端分离的更彻底

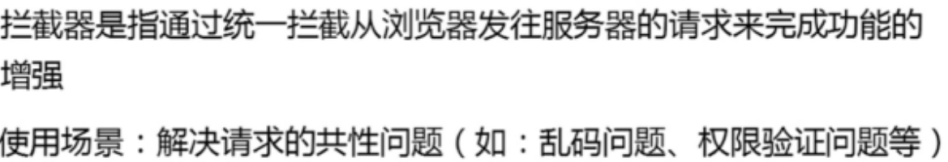
JSON是一种数据交换的格式，springMVC提供了JSON协同支持机制



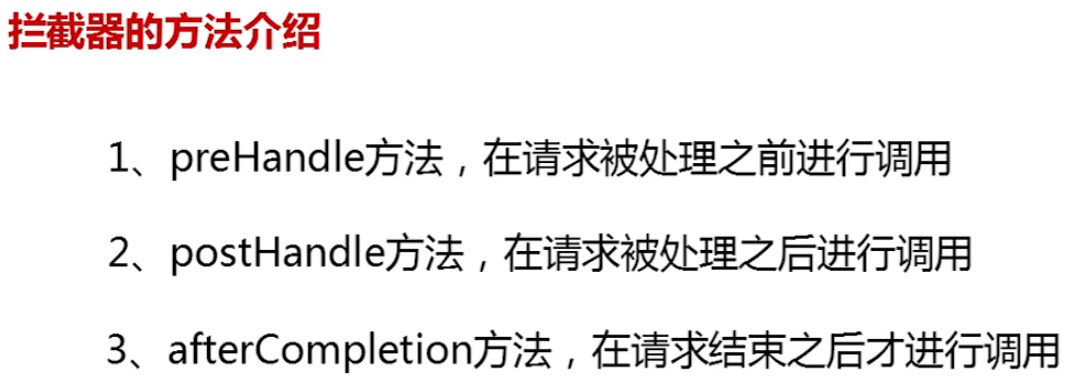
DATA就是model，当访问一个应用时需要的到model的呈现，很多大公司需要后端的APP进行大数据的分析或机器学习，此时JSON就是更好的选择

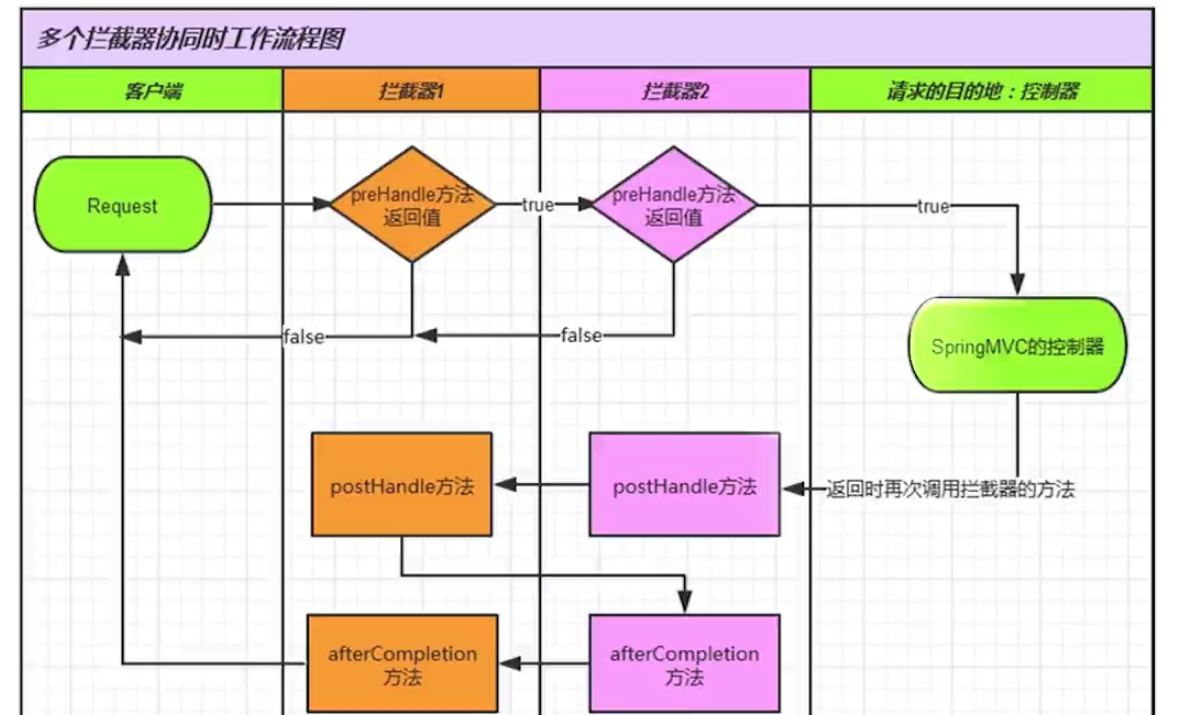
springMVC提供了ContentNegotiatingViewResolver机制来处理这种对相同数据不同呈现方式的应用场景，如果是用户需要得到HTML的数据呈现，那么将数据代理给JSPView，如果是机器需要JSON格式，那么将数据代理给JSONView

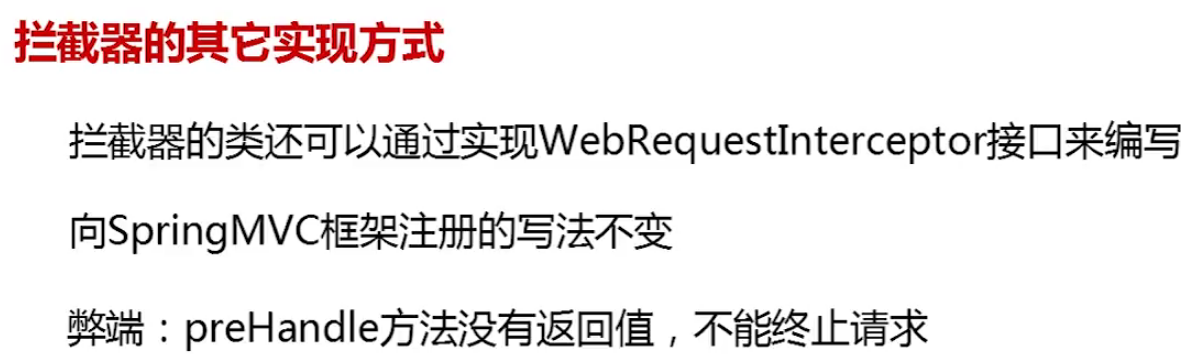
1. **Spring-MVC-拦截器**

  
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

如果都站在springmvc这个web框架所起的作用来看两者.那么 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。 >>在springmvc好像没有什么action吧.如果出了springmvc这个圈子,它的作用就大多了. 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。 明显不对. 过滤器只能在容器初始化时被调用?呵呵. 这个明显不是!springmvc里面的过滤器是继承OncePerRequestFilter.它代表什么意思呢?代表当前这一次http请求只经过一次filter. 下一次,下下一次.其它千千万次都会执行







只是三个方法的参数改变

