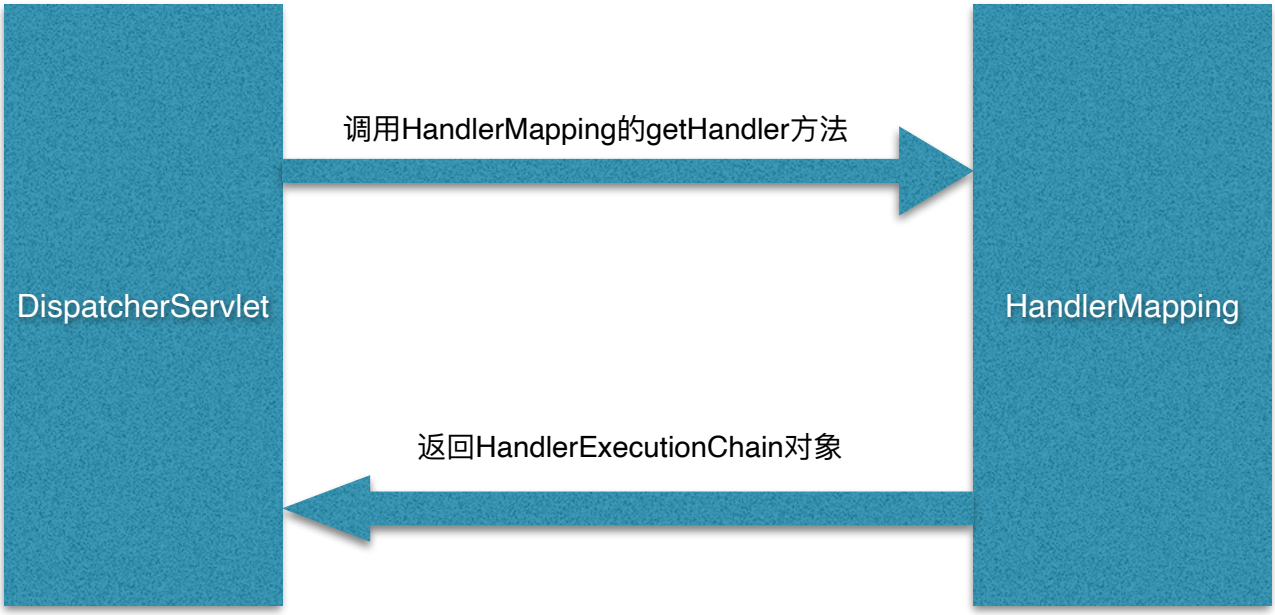
Spring MVC源码阅读-1

DispatcherServlet就是一个普通的servlet，它继承自HttpServlet，是整个框架的入口，同时也是整个框架的指挥中心。在doDispatch方法中，DispatcherServlet调用SpringMvc框架的其它组成部分，完成对整个请求的处理。

DispatcherServlet调用的第一个框架组件是HandlerMapping。HandlerMapping的作用是根据request对象找到对应的请求处理器handler。如下图所示：DispatcherServlet调用HandlerMapping的getHandler方法，传入参数request对象，HandlerMapping会返回处理request对象的handler。



上面的HandlerMapping其实只是一个接口，默认配置下有两个HandlerMapping的实现类可用，分别是BeanNameUrlHandlerMapping和DefaultAnnotationHandlerMapping。DispatcherServlet会依次遍历自己拥有的HandlerMapping，直到能解析出对应的handler为止。如果所有的HandlerMapping都无法找到处理request的Handler，就会报错。

综上所述：HandlerMapping的作用就是告诉DispatcherServlet，这个http请求该由哪个handler进行处理。这个handler其实就是我们编写的@Controller类中的对应方法。

HandlerExecutionChain

上面HandlerMapping反回的对象并不仅仅是一个简单的handler对象，而是一个HandlerExecutionChain对象，顾名思义，这个对象表示一个执行顺序链。其中包括我们自己编写的handler。HandlerExecutionChain对象包括如下内容：

1.真正的handler对象(这也就是@Controller对象)

2.HandlerInterceptor对象的列表(分别有array和list两个列表，它们的内容相同)。

对于HandlerInterceptor对象，顾名思义，就是handler拦截器的意思。也就是说，在调用真正的handler之前，会先调用HandlerInterceptor。

HandlerInterceptor是一个接口，它只提供的三个方法：preHandler、postHandle和afterCompletion。这三个方法的解释如下(来源于网络，其实有错误)：

（1）preHandle方法，顾名思义，该方法将在请求处理之前进行调用。SpringMVC 中的Interceptor是链式的调用的，在一个应用中或者说在一个请求中可以同时存在多个Interceptor 。每个Interceptor的调用会依据它的声明顺序依次执行，而且最先执行的都是

Interceptor中的preHandle方法，所以可以在这个方法中进行一些前置初始化操作或者是对当前请求的一个预处理，也可以在这个方法中进行一些判断来决定请求是否要继续进行下去。该方法的返回值是布尔值Boolean类型的，当它返回为false时，表示请求结束，后续的Interceptor和Controller都不会再执行；当返回值为true时就会继续调用下一个Interceptor的preHandle方法，如果已经是最后一个Interceptor的时候就会是调用当前请求的Controller方法。

（2）postHandle方法，由preHandle方法的解释我们知道这个方法包括后面要说到的afterCompletion方法都只能是在当前所属的Interceptor的preHandle方法的返回值为true 时才能被调用。postHandle方法，顾名思义就是在当前请求进行处理之后，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。postHandle方法被调用的方向跟preHandle是相反的，也就是说先声明的Interceptor的postHandle方法反而会后执行。

（3）afterCompletion方法，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。顾名思义，该方法将在整个请求结束之后，也就是DispatcherServlet渲染了对应的视图之后执行。这个方法的主要作用是用于进行资源清理工作的。

上面对这三个方法的解释来自于网络，有少许错误。比如说HandlerExecutionChain中按顺序包含A、B、C三个HandlerInterceptor。A和B的preHandler方法都返回true，而C得preHandle方法返回false，则@Controller方法将不会执行，三个拦截器的postHandle方法也不会被执行，但是A和B的afterCompletion方法会被执行(先B后A)。这和上面的描述有区别。

```
boolean applyPreHandle(HttpServletRequest request, HttpServletResponse response) {
      HandlerInterceptor[] interceptors = getInterceptors();
      if (!ObjectUtils.isEmpty(interceptors)) {
            for (int i = 0; i < interceptors.length; i++) {
                  HandlerInterceptor interceptor = interceptors[i];
                  if (!interceptor.preHandle(request, response, this.handler)) {
                        triggerAfterCompletion(request, response, null);
                        return false;
                  }
                  this.interceptorIndex = i;
            }
      }
      return true;
}
void triggerAfterCompletion(HttpServletRequest request, HttpServletResponse response,
Exception ex) {
      HandlerInterceptor[] interceptors = getInterceptors();
      if (!ObjectUtils.isEmpty(interceptors)) {
            for (int i = this.interceptorIndex; i >= 0; i--) {
                  HandlerInterceptor interceptor = interceptors[i];
                  try {
                        interceptor.afterCompletion(request, response, this.handler, ex);
                  } catch (Throwable ex2) {
                        logger.error("exception", ex2);
                  }
            }
      }
}
```

综上所述：HandlerExecutionChain既包含了目标handler，又包含了不定数量的HandlerInterceptor。每个HandlerInterceptor都包含preHandle、postHandler和afterCompletion三个方法。handler和HandlerInterceptor中的三个方法都将由DispatcherServlet进行适时的调用。

HandlerAdapter

经过上面的过程，DispatcherServlet已经知道了该用哪个handler处理这个request对象。DispatcherServlet将通过HandlerAdapter调用目标handler。

DispatcherServlet可以拥有多个HandlerAdapter，它会依次遍历每个HandlerAdapter，直到找到一个HandlerAdapter可以支持目标handler的调用为止。

执行过程

现在，DispatcherServlet已经拿到了HandlerExecutionChain和HandlerAdapter两个对象，它执行下面的过程。

1.按顺序调用HandlerExecutionChain中HandlerInterceptor的preHandle方法，如果有任何一个preHandle方法返回false，不再执行后续的任何代码(包括handler)，直接return(其实在return前会以逆序执行那些已经执行的preHandle方法返回true的HandlerInterceptor的afterCompletion方法)。如果所有的方法都返回true，则进入第2个步骤。

2.通过HandlerAdapter对象执行handler，取得执行结果是ModelAndView类型的对象mv。

3.以逆序的方式调用HandlerExecutionChain中所有的HandlerInterceptor的postHandler方法。

4.DispatcherServlet调用ViewResolver，根据viewName、model、locale,request四个参数解析相应的View。DispatcherServlet可能拥有多个ViewResolver，它会依次遍历每个ViewResolver尝试传入上面四个参数，直到能够解析出一个view为止。默认下DispatcherServlet只有一个ViewResolver--InternalResourceViewResolver。

5.拿到对应的View之后，调用view的render方法绘制视图。

6.在View绘制完成之后，调用HandlerExecutionChain中所有的HandlerInterceptor的afterCompletion方法。

7.结束

接触到的几个类：

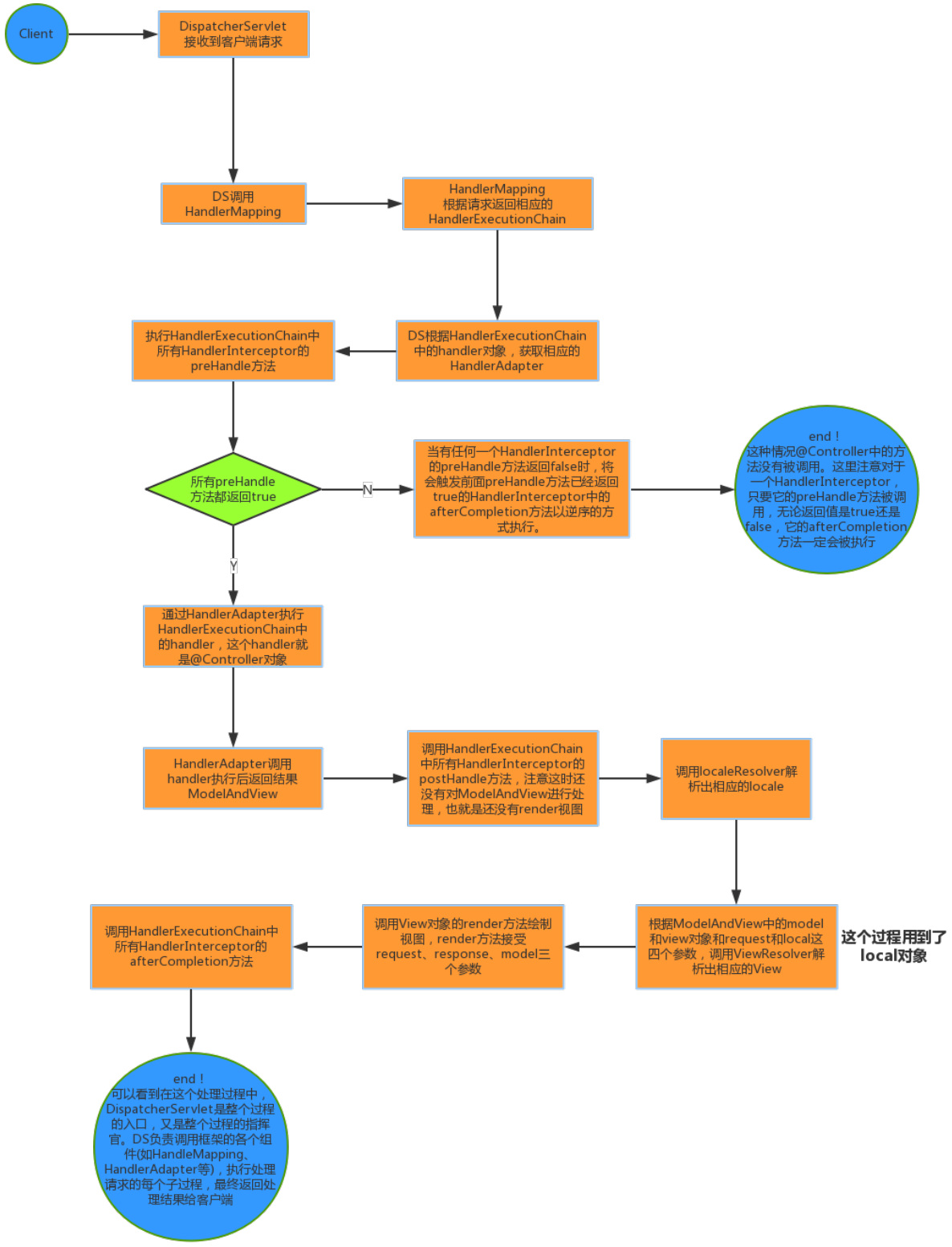
1.DispatcherServlet
2.HandlerMapping
    BeanNameUrlHandlerMapping
    DefaultAnnotationHandlerMapping
3.HandlerExecutionChain
4.HandlerInterceptor.
5.HandlerAdapter
    HttpRequestHandlerAdapter
    SimpleControllerHandlerAdapter
    AnnotationMethodHandlerAdapter
6.ViewResolver

InternalResourceViewResolver

7.View

8.localeResolver

AcceptHeaderLocaleResolver



DispatcherServlet.properties文件的内容

```
# Default implementation classes for DispatcherServlet's strategy
interfaces.
# Used as fallback when no matching beans are found in the
DispatcherServlet context.
# Not meant to be customized by application developers.

org.springframework.web.servlet.LocaleResolver=org.springframework.web.se
rvlet.i18n.AcceptHeaderLocaleResolver

org.springframework.web.servlet.ThemeResolver=org.springframework.web.ser
vlet.theme.FixedThemeResolver

org.springframework.web.servlet.HandlerMapping=org.springframework.web.se
rvlet.handler.BeanNameUrlHandlerMapping,\

org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMa
pping

org.springframework.web.servlet.HandlerAdapter=org.springframework.web.se
rvlet.mvc.HttpRequestHandlerAdapter,\
      org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter,
\

org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAda
pter

org.springframework.web.servlet.HandlerExceptionResolver=org.springframew
ork.web.servlet.mvc.annotation.AnnotationMethodHandlerExceptionResolver,\

org.springframework.web.servlet.mvc.annotation.ResponseStatusExceptionRes
olver,\

org.springframework.web.servlet.mvc.support.DefaultHandlerExceptionResolv
er

org.springframework.web.servlet.RequestToViewNameTranslator=org.springfra
mework.web.servlet.view.DefaultRequestToViewNameTranslator

org.springframework.web.servlet.ViewResolver=org.springframework.web.serv
let.view.InternalResourceViewResolver

org.springframework.web.servlet.FlashMapManager=org.springframework.web.s
ervlet.support.SessionFlashMapManager
```

HttpMessageConverter

@Value 读取外部文件的值
@Valid
HandlerInterceptorAdapter

自动添加到HandlerExecutionChain中的Interceptor
ConversionServiceExposingInterceptor