# SpringMVC执行流程分析

### org.springframework.web.servlet.DispatcherServlet

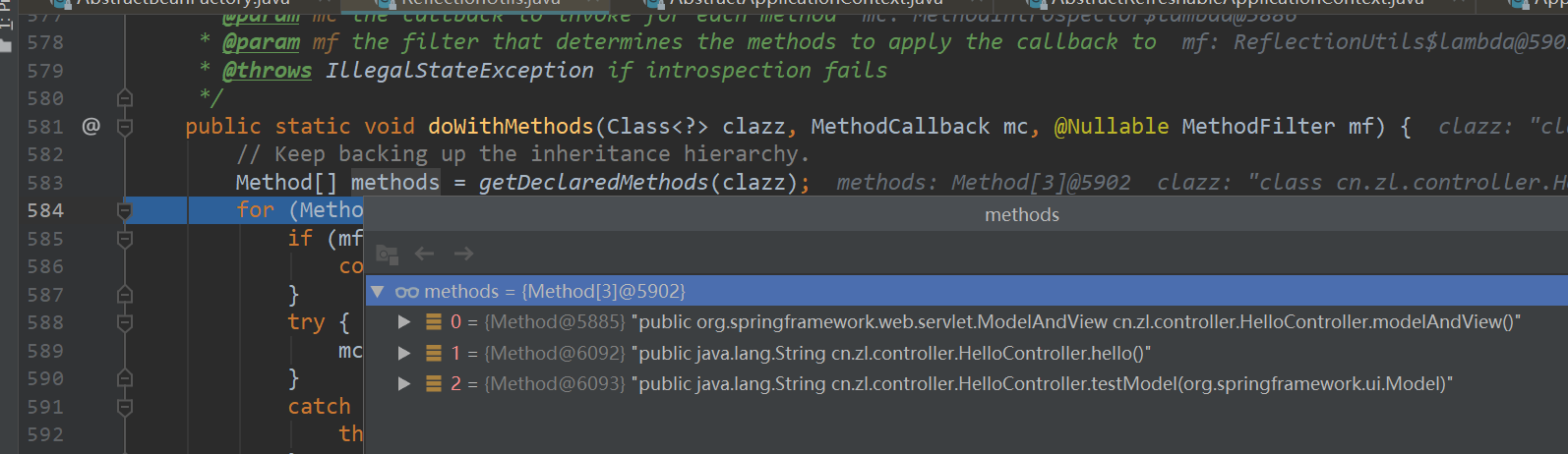
* 1. #doDispatch(HttpServletRequest request, HttpServletResponse response)
     1. org.springframework.web.servlet.DispatcherServlet#checkMultipart:
        1. Convert the request into a multipart request, and make multipart resolver available.
           1. 转换当前请求为 multipart请求，并使 multipart可用
        2. If no multipart resolver is set, simply use the existing request.
           1. 如果没有设置multipart解析器，则使用已存在的请求
     2. org.springframework.web.servlet.DispatcherServlet#getHandler
        1. Determine handler for the current request.
           1. 确定当前请求的处理程序。
     3. org.springframework.web.servlet.DispatcherServlet#getHandlerAdapter
        1. Determine handler adapter for the current request.
           1. 确定当前请求的适配器
     4. Process last-modified header, if supported by the handler.
        1. 处理最后修改的标头，如果处理程序支持。
     5. org.springframework.web.servlet.HandlerExecutionChain#applyPreHandle
        1. Apply preHandle methods of registered interceptors.
           1. 对已注册拦截器应用预(前置)处理方法
     6. org.springframework.web.servlet.HandlerAdapter#handle
        1. Actually invoke the handler.、
           1. 实际调用处理程序
        2. Use the given handler to handle this request.
           1. 使用给定的处理程序来处理此请求。
     7. org.springframework.web.servlet.DispatcherServlet#applyDefaultViewName
        1. org.springframework.web.servlet.DispatcherServlet#getDefaultViewName
           1. Translate the supplied request into a default view name.

将提供的请求转换为默认视图名称。

* + 1. org.springframework.web.servlet.HandlerExecutionChain#applyPostHandle
       1. Apply postHandle methods of registered interceptors.
          1. 对已注册的拦截器应用后置处理方法
    2. org.springframework.web.servlet.DispatcherServlet#processDispatchResult
       1. Handle the result of handler selection and handler invocation, which is either a ModelAndView or an Exception to be resolved to a ModelAndView.
          1. 把程序选择和程序徐调用的结果解析为ModelAndView或者异常

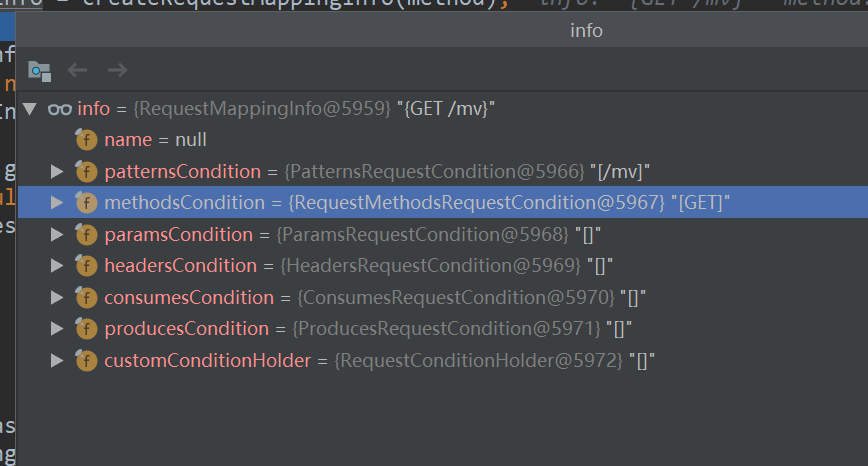
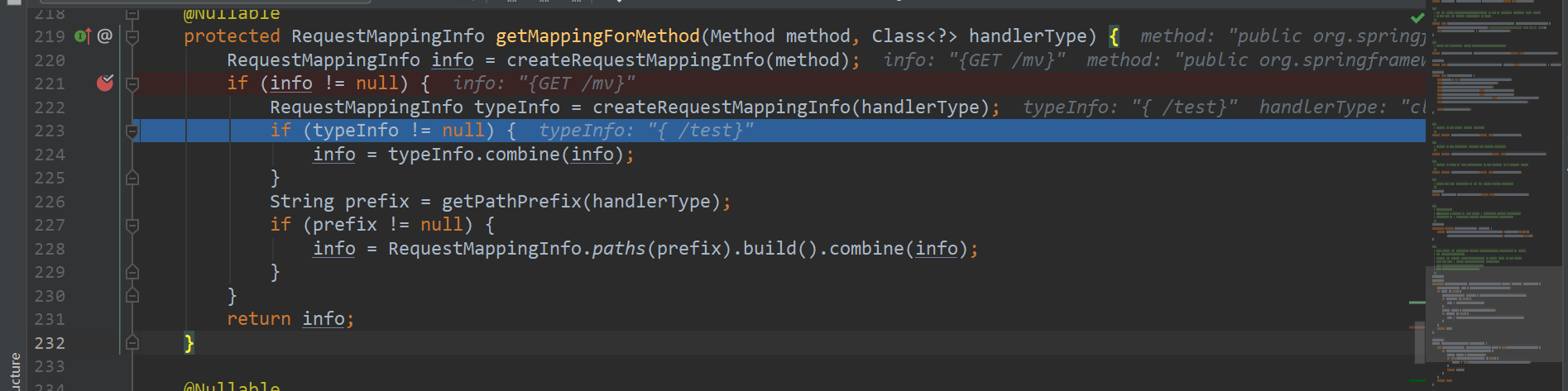
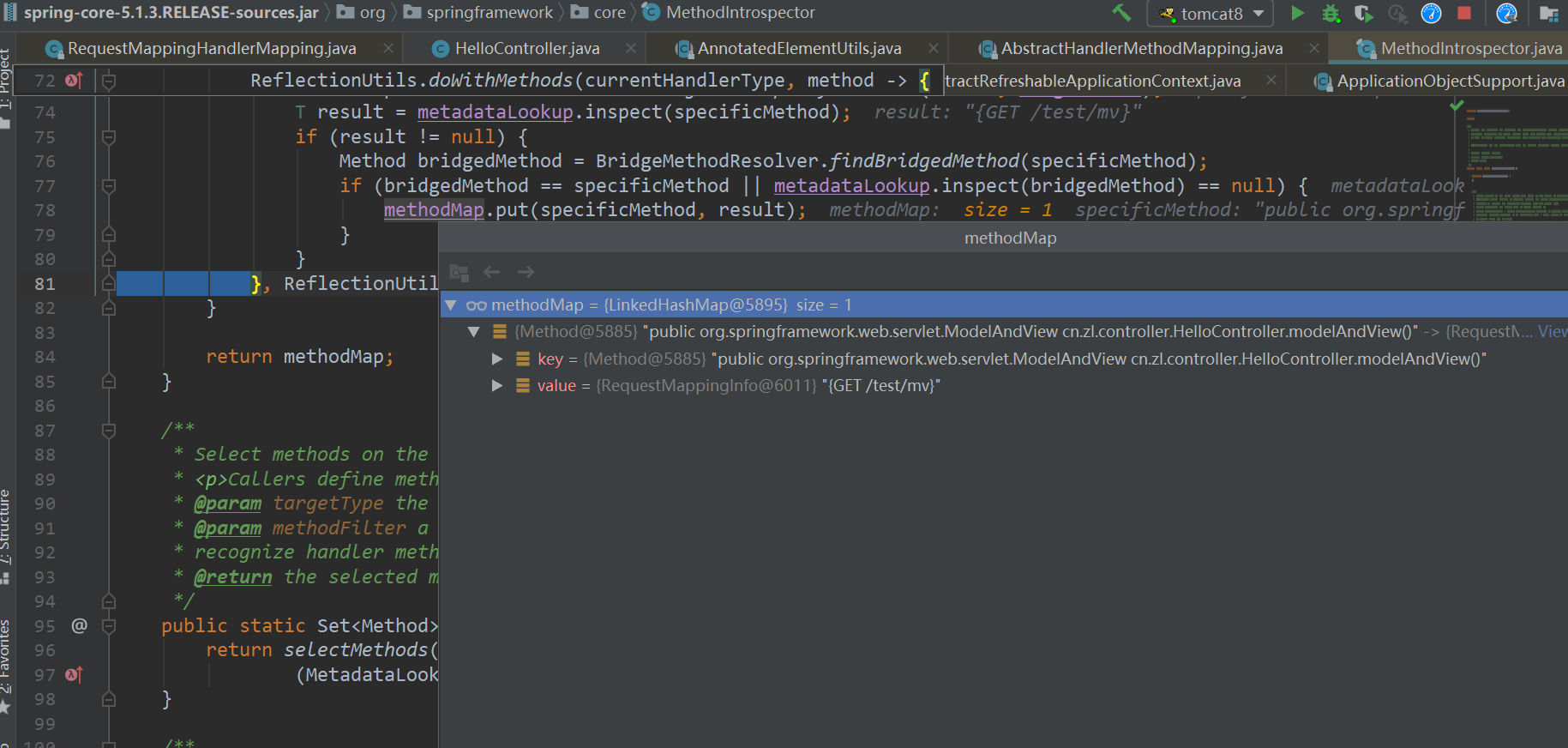
### SpringMvc扫描注解并注入的过程

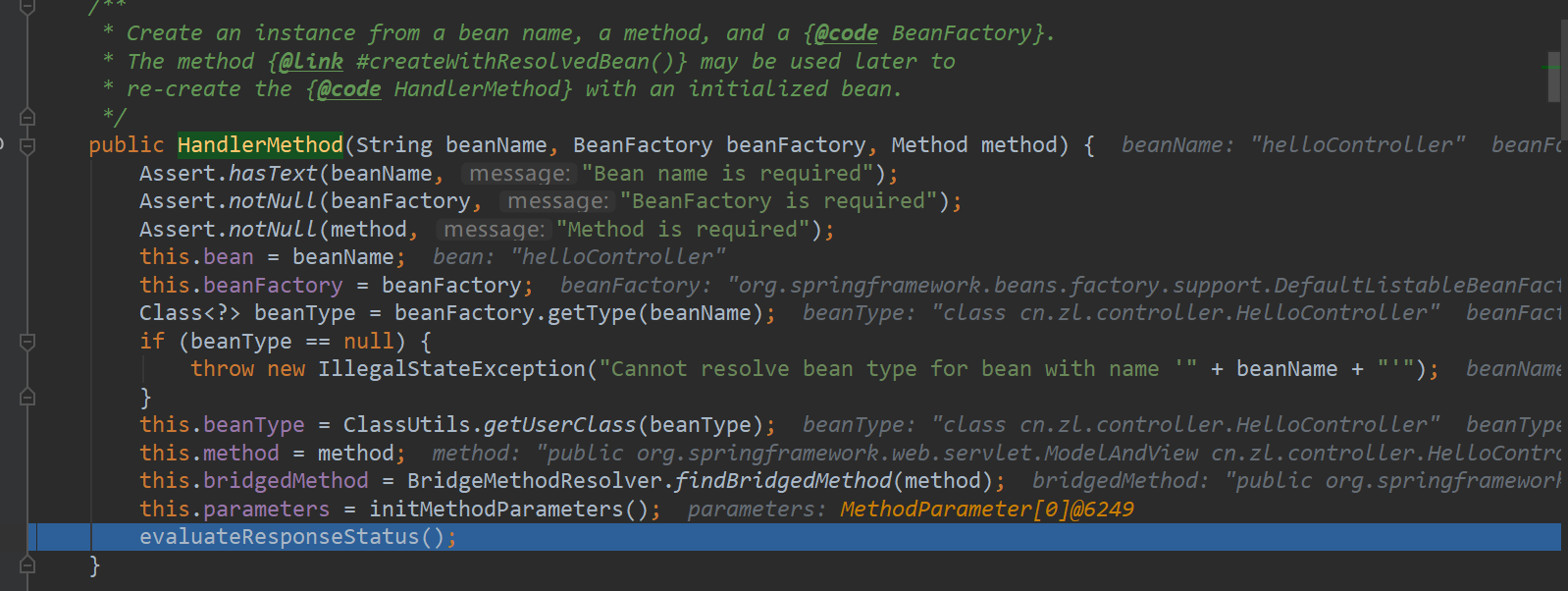
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping

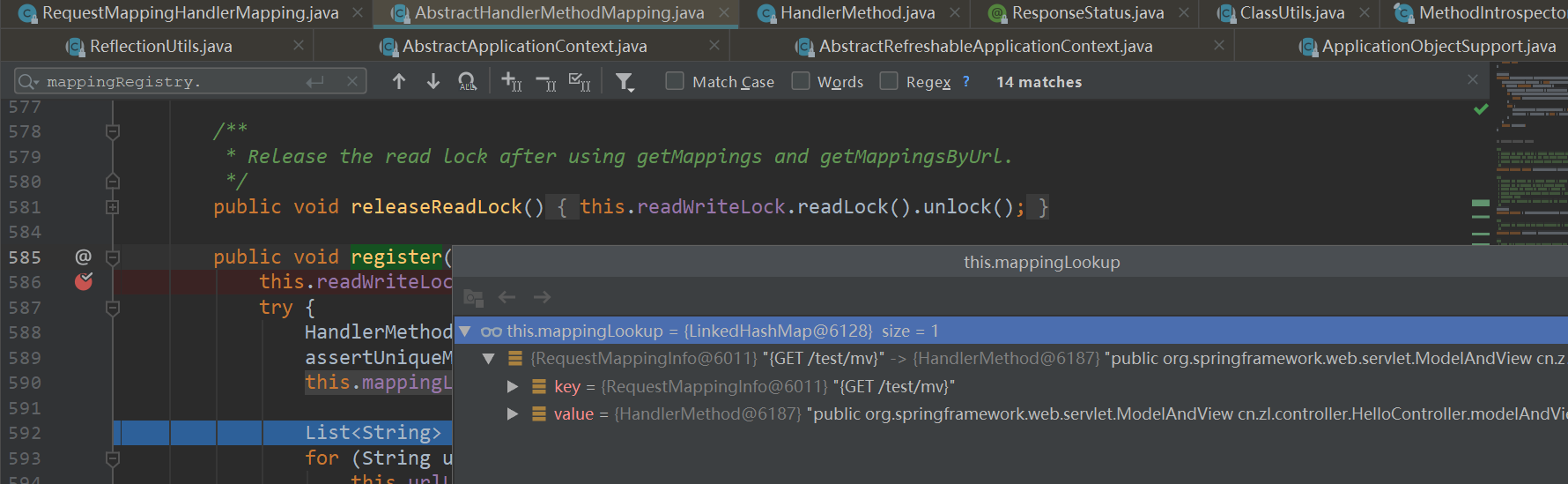
* 1. org.springframework.web.servlet.handler.AbstractHandlerMethodMapping#initHandlerMethods
     1. Scan beans in the ApplicationContext, detect and register handler methods.
        1. 在Application Context中扫描Bean，检测并注册handler方法
     2. org.springframework.web.servlet.handler.AbstractHandlerMethodMapping#processCandidateBean 处理候选bean
        1. Determine the type of the specified candidate bean and call {@link #detectHandlerMethods} if identified as a handler type. <p>This implementation avoids bean creation through checking {@link org.springframework.beans.factory.BeanFactory#getType} and calling {@link #detectHandlerMethods} with the bean name.
           1. 检测指定候选bean的类型且确定是一个处理类型则调用{@link #detectHandlerMethods}.
           2. 此实现通过检查{@link org.springframework.beans.factory.BeanFactory＃getType}并使用bean名称调用{@link #detectHandlerMethods}来避免创建bean。
  2. org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping#createRequestMappingInfo(java.lang.reflect.AnnotatedElement)
     1. Delegates to {@link #createRequestMappingInfo(RequestMapping, RequestCondition)}, supplying the appropriate custom {@link RequestCondition} depending on whether the supplied {@code annotatedElement} is a class or method.
        1. 

当前Controller所有的方法

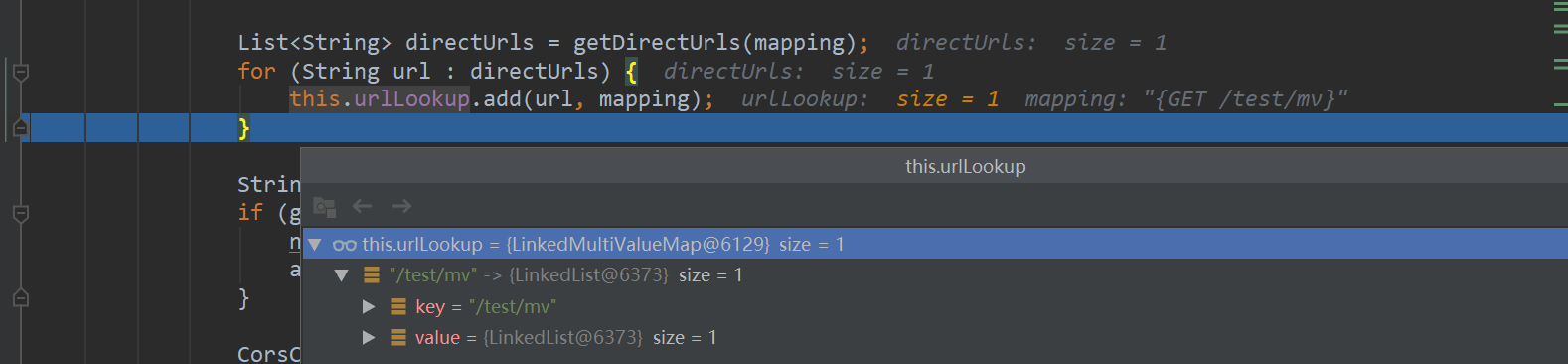
委托{@link #createRequestMappingInfo（RequestMapping，RequestCondition）}，提供适当的自定义{@link RequestCondition}，具体取决于提供的{@code annotatedElement}是类还是方法。

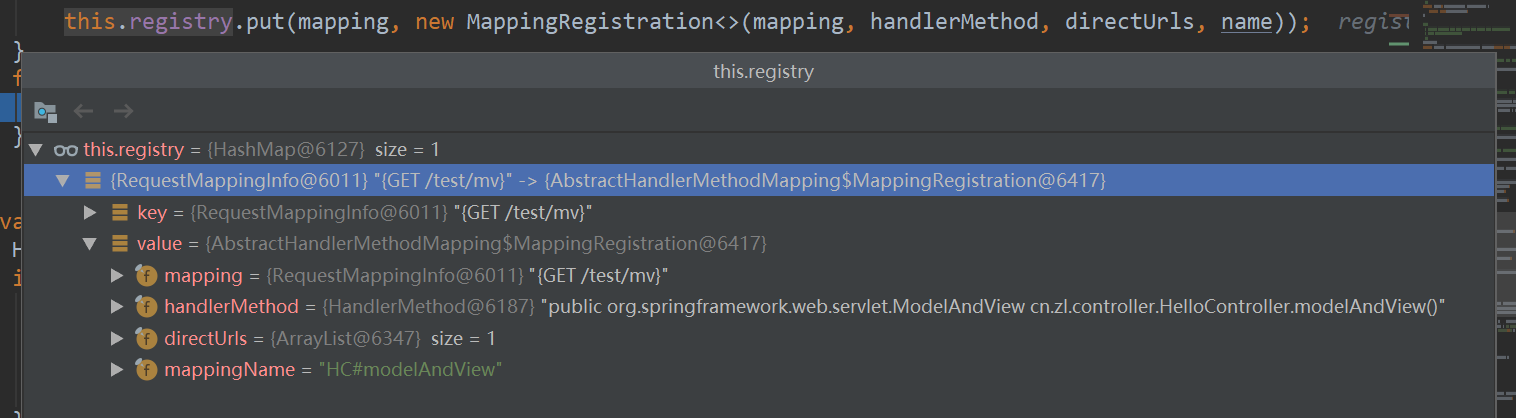
* + - 1. 
      2. 
      3. 通过LinkedHashMap来存储Controller方法名称和请求类型以及访问路径
  1. org.springframework.web.method.HandlerMethod#HandlerMethod(java.lang.String, org.springframework.beans.factory.BeanFactory, java.lang.reflect.Method)
     1. Create an instance from a bean name, a method, and a {@code BeanFactory}. The method {@link #createWithResolvedBean()} may be used later to re-create the {@code HandlerMethod} with an initialized bean.
        1. 从bean名称，方法和{@code BeanFactory}创建实例。 稍后可以使用方法{@link #createWithResolvedBean()}来使用初始化的bean重新创建{@code HandlerMethod}。

构建一个类的过程

* 1. org.springframework.web.method.HandlerMethod#evaluateResponseStatus
  2. org.springframework.web.servlet.handler.AbstractHandlerMethodMapping.MappingRegistry#register
     1. 

Controller方法的注解和方法的名称也是k-v，LinkedHashMap存储

* + 1. 最终路径映射如下
    2. 最终注册的HashMap的存储结果



值得一提的是，spring在此处使用重入锁避免多线程竞争

