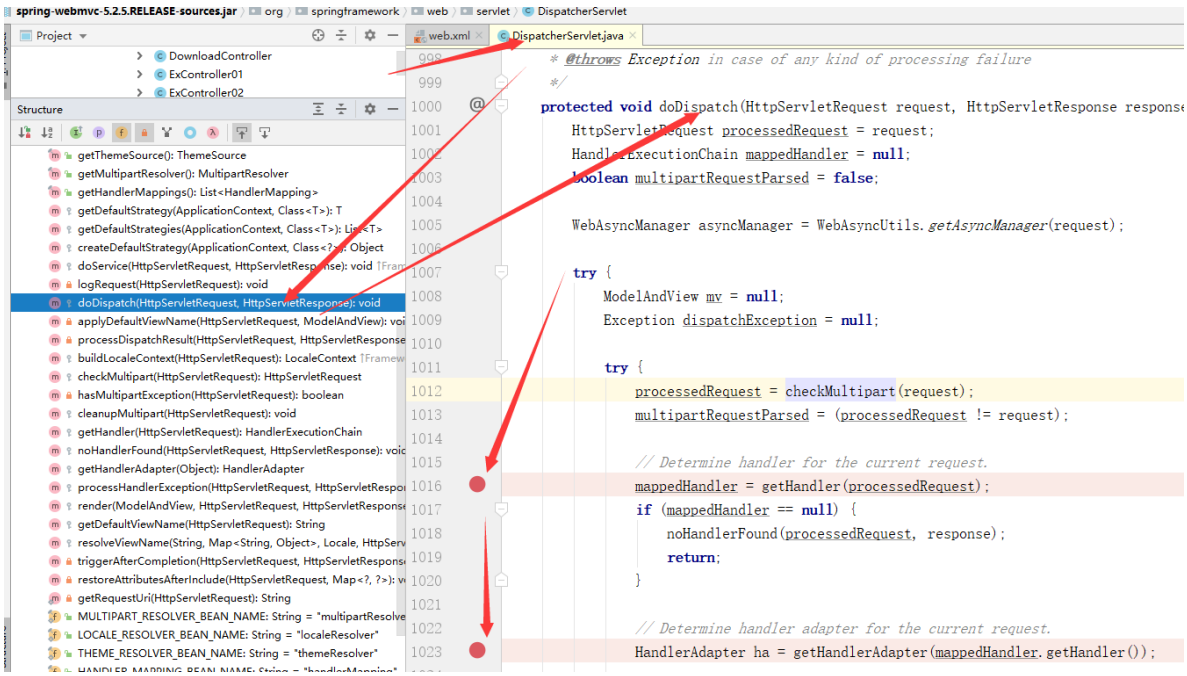
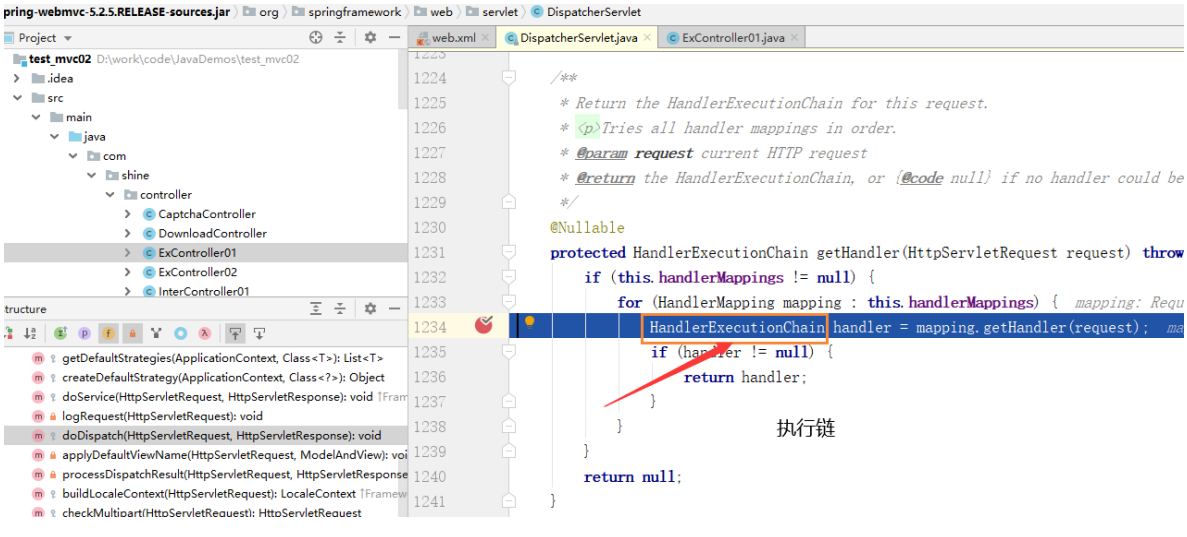
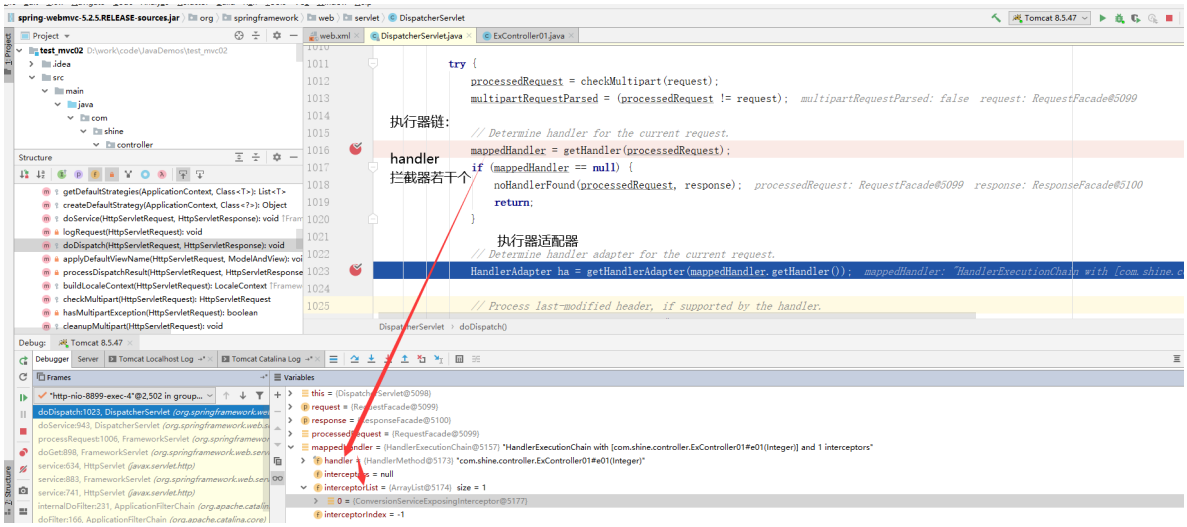
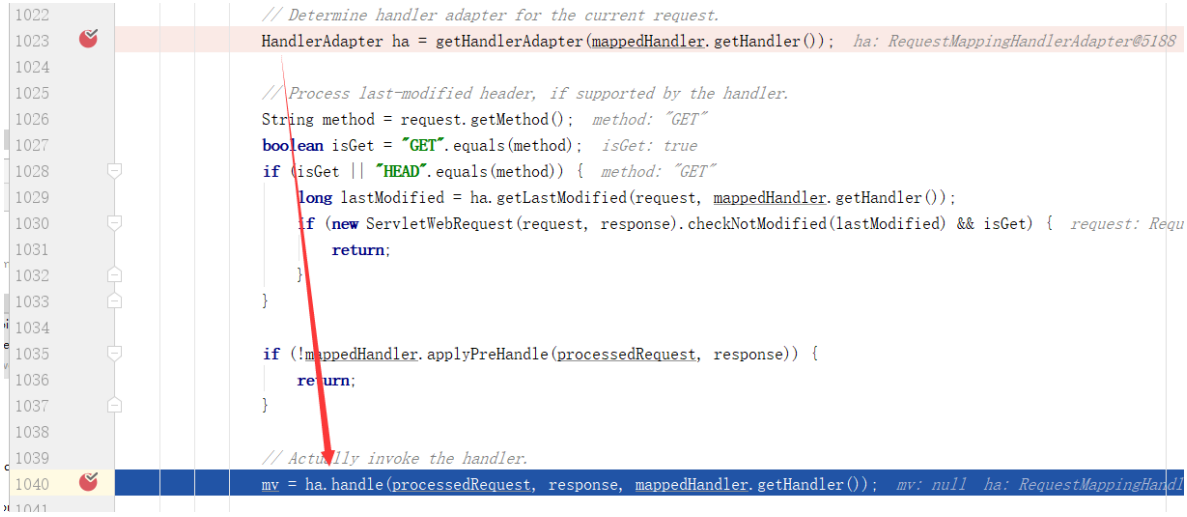
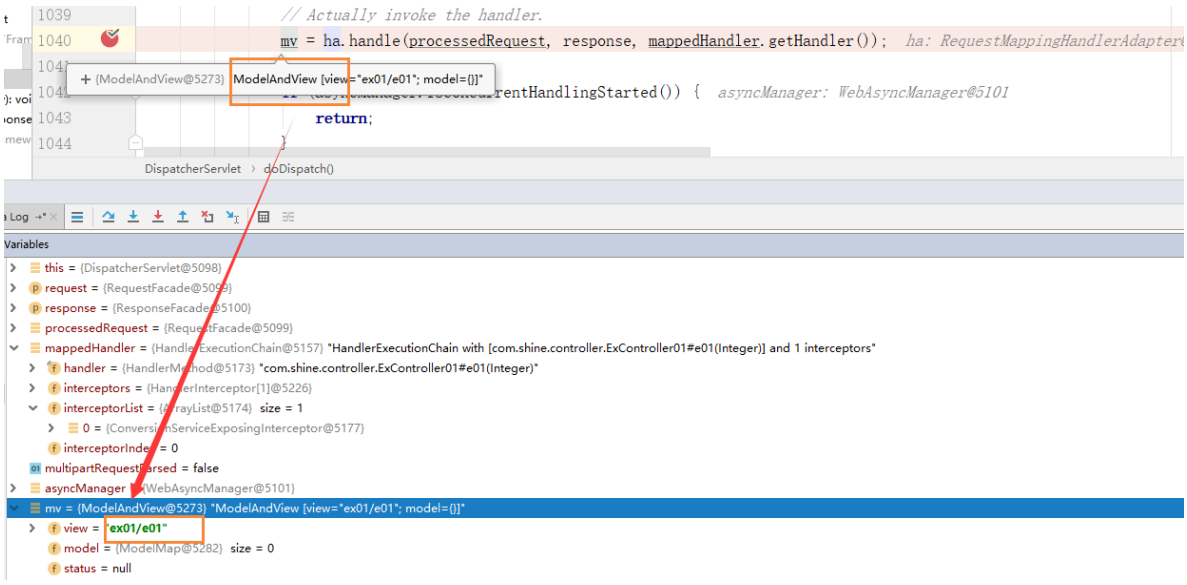
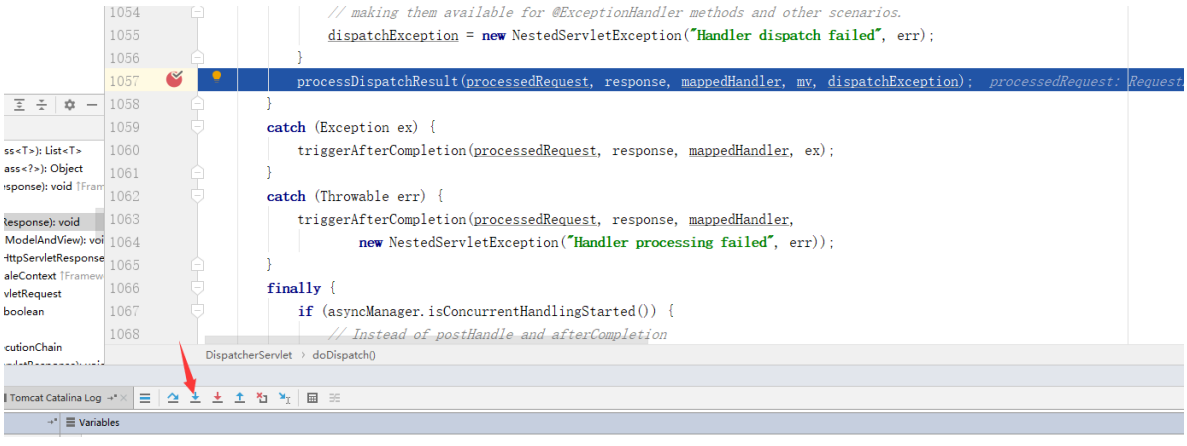

spring-webmvc-5.2.5.RELEASE-sources.jar
org
springframework
web
servlet
DispatcherServlet
Project
Structure
web.xml
DispatcherServlet.java
DownloadController
ExController01
ExController02
getThemeSource(): ThemeSource
getMultipartResolver(): MultipartResolver
getHandlerMappings(): List<HandlerMapping>
getDefaultStrategy(ApplicationContext, Class<T>): T
getDefaultStrategies(ApplicationContext, Class<T>): List<T>
createDefaultStrategy(ApplicationContext, Class<?>): Object
doService(HttpServletRequest, HttpServletResponse): void
logRequest(HttpServletRequest): void
doDispatch(HttpServletRequest, HttpServletResponse): void
applyDefaultViewName(HttpServletRequest, ModelAndView): voi
processDispatchResult(HttpServletRequest, HttpServletResponse
buildLocaleContext(HttpServletRequest): LocaleContext
checkMultipart(HttpServletRequest): HttpServletRequest
hasMultipartException(HttpServletRequest): boolean
cleanupMultipart(HttpServletRequest): void
getHandler(HttpServletRequest): HandlerExecutionChain
noHandlerFound(HttpServletRequest, HttpServletResponse): voi
getHandlerAdapter(Object): HandlerAdapter
processHandlerException(HttpServletRequest, HttpServletRespo
render(ModelAndView, HttpServletRequest, HttpServletRespons
getDefaultViewName(HttpServletRequest): String
resolveViewName(String, Map<String, Object>, Locale, HttpServ
triggerAfterCompletion(HttpServletRequest, HttpServletRespons
restoreAttributesAfterInclude(HttpServletRequest, Map<?, ?>):
getRequestUri(HttpServletRequest): String
MULTIPART_RESOLVER_BEAN_NAME: String = "multipartResolve
LOCALE_RESOLVER_BEAN_NAME: String = "localeResolver"
THEME_RESOLVER_BEAN_NAME: String = "themeResolver"
* @throws Exception in case of any kind of processing failure
protected void doDispatch(HttpServletRequest request, HttpServletResponse response
HttpServletRequest processedRequest = request;
HandlerExecutionChain mappedHandler = null;
boolean multipartRequestParsed = false;
WebAsyncManager asyncManager = WebAsyncUtils.getAsyncManager(request);
try {
ModelAndView mv = null;
Exception dispatchException = null;
try {
processedRequest = checkMultipart(request);
multipartRequestParsed = (processedRequest != request);
// Determine handler for the current request.
mappedHandler = getHandler(processedRequest);
if (mappedHandler == null) {
noHandlerFound(processedRequest, response);
return;
}
// Determine handler adapter for the current request.
HandlerAdapter ha = getHandlerAdapter(mappedHandler.getHandler());

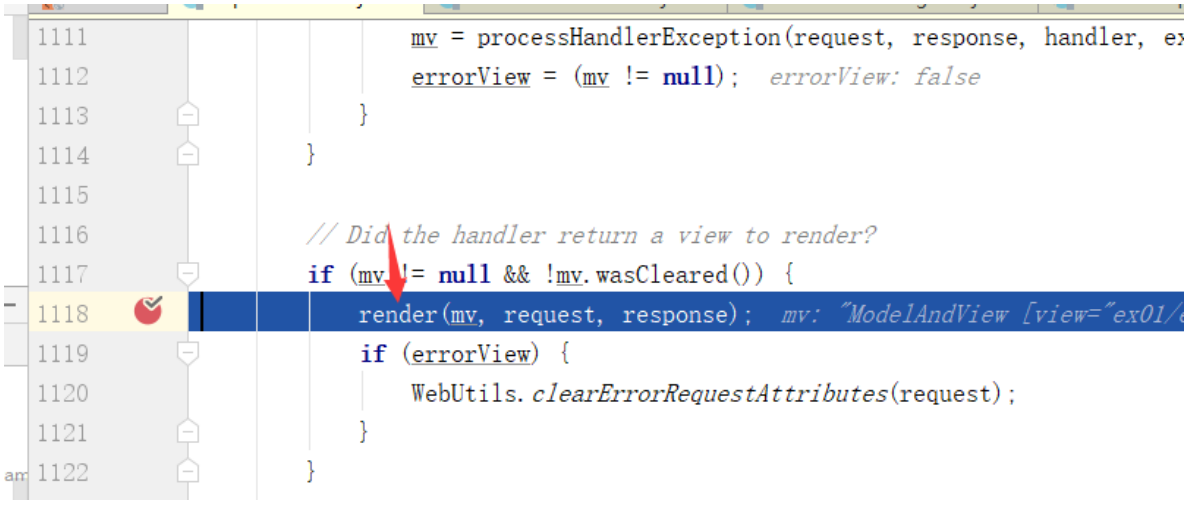

spring-webmvc-5.2.5.RELEASE-sources.jar
org
springframework
web
servlet
DispatcherServlet
Project
structure
web.xml
DispatcherServlet.java
ExController01.java
test_mvc02 D:\work\code\JavaDemos\test_mvc02
.idea
src
main
java
com
shine
controller
CaptchaController
DownloadController
ExController01
ExController02
InterController01
getDefaultStrategies(ApplicationContext, Class<T>): List<T>
createDefaultStrategy(ApplicationContext, Class<?>): Object
doService(HttpServletRequest, HttpServletResponse): void
logRequest(HttpServletRequest): void
doDispatch(HttpServletRequest, HttpServletResponse): void
applyDefaultViewName(HttpServletRequest, ModelAndView): voi
processDispatchResult(HttpServletRequest, HttpServletResponse
buildLocaleContext(HttpServletRequest): LocaleContext
checkMultipart(HttpServletRequest): HttpServletRequest
/**
* Return the HandlerExecutionChain for this request.
* <p>Tries all handler mappings in order.
* @param request current HTTP request
* @return the HandlerExecutionChain, or {@code null} if no handler could be
*/
@Nullable
protected HandlerExecutionChain getHandler(HttpServletRequest request) throw
if (this.handlerMappings != null) {
for (HandlerMapping mapping : this.handlerMappings) {
HandlerExecutionChain handler = mapping.getHandler(request);
if (handler != null) {
return handler;
}
}
}
return null;
}
执行链


try {
processedRequest = checkMultipart(request);
multipartRequestParsed = (processedRequest != request);
multipartRequestParsed: false
request: RequestFacade@5099
执行器链:
// Determine handler for the current request.
mappedHandler = getHandler(processedRequest);
handler
拦截器若干个
if (mappedHandler == null) {
noHandlerFound(processedRequest, response);
processedRequest: RequestFacade@5099
response: ResponseFacade@5100
return;
}
执行器适配器
// Determine handler adapter for the current request.
HandlerAdapter ha = getHandlerAdapter(mappedHandler.getHandler());
mappedHandler: "HandlerExecutionChain with [com.shine.c
// Process last-modified header, if supported by the handler.
DispatcherServlet
doDispatch()
Tomcat 8.5.47
Debug:
Debugger
Server
Tomcat Localhost Log
Tomcat Catalina Log
Frames
Variables
"http-nio-8899-exec-4"@2,502 in group...
doDispatch:1023, DispatcherServlet (org.springframework.web
doService:943, DispatcherServlet (org.springframework.web.s
processRequest:1006, FrameworkServlet (org.springframewor
doGet:898, FrameworkServlet (org.springframework.web.serv
service:634, HttpServlet (javax.servlet.http)
service:883, FrameworkServlet (org.springframework.web.serv
service:741, HttpServlet (javax.servlet.http)
internalDoFilter:231, ApplicationFilterChain (org.apache.catali
doFilter:166, ApplicationFilterChain (org.apache.catalina.core)
this = (DispatcherServlet@5098)
request = (RequestFacade@5099)
response = (ResponseFacade@5100)
processedRequest = (RequestFacade@5099)
mappedHandler = (HandlerExecutionChain@5157) "HandlerExecutionChain with [com.shine.controller.ExController01#e01(Integer)] and 1 interceptors"
handler = (HandlerMethod@5173) "com.shine.controller.ExController01#e01(Integer)"
interceptors = null
interceptorList = (ArrayList@5174) size = 1
0 = (ConversionServiceExposingInterceptor@5177)
interceptorIndex = -1


// Determine handler adapter for the current request.
HandlerAdapter ha = getHandlerAdapter(mappedHandler.getHandler());
// Process last-modified header, if supported by the handler.
String method = request.getMethod();
boolean isGet = "GET".equals(method);
if (isGet || "HEAD".equals(method)) {
long lastModified = ha.getLastModified(request, mappedHandler.getHandler());
if (new ServletWebRequest(request, response).checkNotModified(lastModified) && isGet) {
return;
}
}
if (!mappedHandler.applyPreHandle(processedRequest, response)) {
return;
}
// Actually invoke the handler.
mv = ha.handle(processedRequest, response, mappedHandler.getHandler());


// Actually invoke the handler.
mv = ha.handle(processedRequest, response, mappedHandler.getHandler());
{ModelAndView@5273} ModelAndView [view="ex01/e01"; model={}]
return;
DispatcherServlet › doDispatch()
Variables
this = {DispatcherServlet@5098}
request = {RequestFacade@5099}
response = {ResponseFacade@5100}
processedRequest = {RequestFacade@5099}
mappedHandler = {HandlerExecutionChain@5157} "HandlerExecutionChain with [com.shine.controller.ExController01#e01(Integer)] and 1 interceptors"
handler = {HandlerMethod@5173} "com.shine.controller.ExController01#e01(Integer)"
interceptors = {HandlerInterceptor[1]@5226}
interceptorList = {ArrayList@5174} size = 1
0 = {ConversionServiceExposingInterceptor@5177}
multipartRequestParsed = false
asyncManager = {WebAsyncManager@5101}
mv = {ModelAndView@5273} "ModelAndView [view="ex01/e01"; model={}]"
view = "ex01/e01"
model = {ModelMap@5282} size = 0
status = null


// making them available for @ExceptionHandler methods and other scenarios.
dispatchException = new NestedServletException("Handler dispatch failed", err);
processDispatchResult(processedRequest, response, mappedHandler, mv, dispatchException);
catch (Exception ex) {
triggerAfterCompletion(processedRequest, response, mappedHandler, ex);
catch (Throwable err) {
triggerAfterCompletion(processedRequest, response, mappedHandler,
new NestedServletException("Handler processing failed", err));
finally {
if (asyncManager.isConcurrentHandlingStarted()) {
// Instead of postHandle and afterCompletion
DispatcherServlet › doDispatch()
Tomcat Catalina Log
Variables


mv = processHandlerException(request, response, handler, ex
errorView = (mv != null); errorView: false
// Did the handler return a view to render?
if (mv != null && !mv.wasCleared()) {
render(mv, request, response); mv: "ModelAndView [view="ex01/
if (errorView) {
WebUtils.clearErrorRequestAttributes(request);

```
View view;
String viewName = mv.getViewName();   viewName: "ex01/e01"
if (viewName != null) {
    // We need to resolve the view name.
    view = resolveViewName(viewName, mv.getModelInternal(), locale, request);
    if (view == null) {
        throw new ServletException("Could not resolve view with name '" + mv.getViewNa
                "' in servlet with name '" + getServletName() + "'");
```

```
@Nullable
protected View resolveViewName(String viewName, @Nullable Map<String, Object> model,   viewName: "ex01/e01"  model:  size = 0
        Locale locale, HttpServletRequest request) throws Exception {   locale: "zh_CN"  request: RequestFacade@5099
```

根据视图名称解析得到视图对象

```
    if (this.viewResolvers != null) {
        for (ViewResolver viewResolver : this.viewResolvers) {   viewResolver: InternalResourceViewResolver@5675  viewResolvers:  size = 1
            View view = viewResolver.resolveViewName(viewName, locale);   view: "org.springframework.web.servlet.view.InternalResourceView
            if (view != null) {
                return view;   view: "org.springframework.web.servlet.view.InternalResourceView: name 'ex01/e01'; URL [/ex01/e01.jsp]"
            }
        }
    }
    return null;
}
```

```
if (viewName != null) {
    // We need to resolve the view name.
    view = resolveViewName(viewName, mv.getModelInternal(), locale, request);   viewName: "ex01/e01"  mv: "ModelAndVi
    if (view == null) {   view: "org.springframework.web.servlet.view.InternalResourceView: name 'ex01/e01'; URL [/ex
        throw new ServletException("Could not resolve view with name '" + mv.getViewName() +
                "' in servlet with name '" + getServletName() + "'");
    }
}
else {
    // No need to lookup: the ModelAndView object contains the actual View object.
    view = mv.getView();
```

DispatcherServlet › render()

Variables

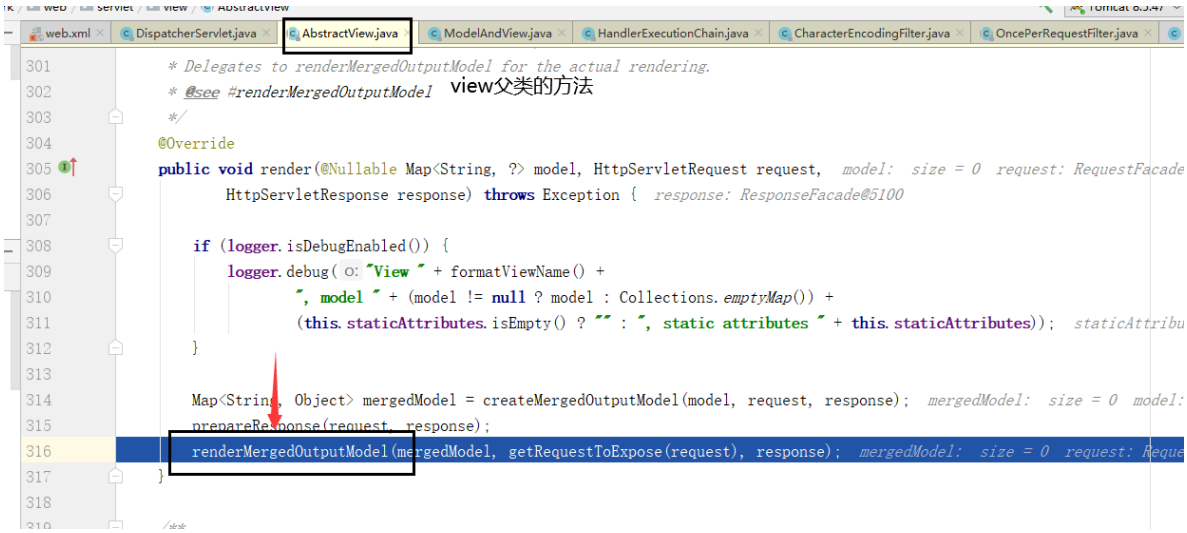
```
response = {ResponseFacade@5100}
locale = {Locale@5638} "zh_CN"
viewName = "ex01/e01"
view = {InternalResourceView@5680} "org.springframework.web.servlet.view.InternalResourceView: name 'ex01/e01'; URL [/ex01/e01.jsp]"
    alwaysInclude = false
    preventDispatchLoop = true
    url = "/ex01/e01.jsp"
    contentType = "text/html;charset=ISO-8859-1"
    requestContextAttribute = null
    staticAttributes = {LinkedHashMap@5704}  size = 0
    exposePathVariables = true
    exposeContextBeansAsAttributes = false
    exposedContextBeanNames = null
    beanName = "ex01/e01"
```

```
            response.setStatus(mv.getStatus().value());
        }
        view.render(mv.getModelInternal(), request, response);   view: "org.springframework.web.servlet.view.InternalResou
    }
    catch (Exception ex) {
        if (logger.isDebugEnabled()) {
            logger.debug( o: "Error rendering view [" + view + "]", ex);
        }
        throw ex;
    }
}
```

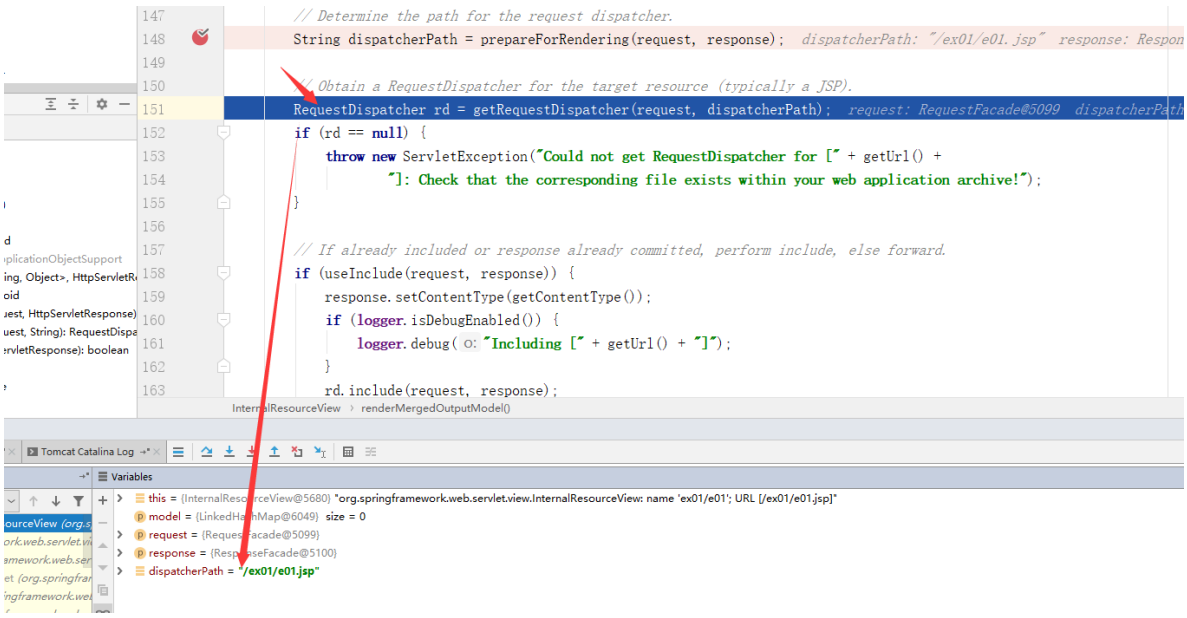
DispatcherServlet › render()

Variables

```
this = {DispatcherServlet@5098}
mv = {ModelAndView@5584} "ModelAndView [view="ex01/e01"; model={}]"
request = {RequestFacade@5099}
response = {ResponseFacade@5100}
locale = {Locale@5638} "zh_CN"
viewName = "ex01/e01"
view = {InternalResourceView@5680} "org.springframework.web.servlet.view.InternalResourceView: name 'ex01/e01'; URL [/ex01/e01.jsp]"
    alwaysInclude = false
```


web.xml
DispatcherServlet.java
AbstractView.java
ModelAndView.java
HandlerExecutionChain.java
CharacterEncodingFilter.java
OncePerRequestFilter.java
view父类的方法
* Delegates to renderMergedOutputModel for the actual rendering.
* @see #renderMergedOutputModel
@Override
public void render(@Nullable Map<String, ?> model, HttpServletRequest request, model: size = 0 request: RequestFacade
HttpServletResponse response) throws Exception { response: ResponseFacade@5100
if (logger.isDebugEnabled()) {
logger.debug(o: "View " + formatViewName() +
", model " + (model != null ? model : Collections.emptyMap()) +
(this.staticAttributes.isEmpty() ? "" : ", static attributes " + this.staticAttributes)); staticAttribu
}
Map<String, Object> mergedModel = createMergedOutputModel(model, request, response); mergedModel: size = 0 model:
prepareResponse(request, response);
renderMergedOutputModel(mergedModel, getRequestToExpose(request), response); mergedModel: size = 0 request: Reque
}


web
servlet
view
InternalResourceView
web.xml
DispatcherServlet.java
FrameworkServlet.java
AbstractView.java
InternalResourceView.java
ModelAndView.java
protected void renderMergedOutputModel(
Map<String, Object> model, HttpServletRequest request, HttpServletResponse response) throw
// Expose the model object as request attributes.
exposeModelAsRequestAttributes(model, request); model: size = 0
// Expose helpers as request attributes, if any.
exposeHelpers(request);
// Determine the path for the request dispatcher.
String dispatcherPath = prepareForRendering(request, response); request: RequestFacade@5099
// Obtain a RequestDispatcher for the target resource (typically a JSP).
RequestDispatcher rd = getRequestDispatcher(request, dispatcherPath);
if (rd == null) {
@Nullable
protected RequestDispatcher getRequestDispatcher(HttpServletRequest request, String path) { request: Requ
return request.getRequestDispatcher(path); request: RequestFacade@5099 path: "/ex01/e01.jsp"
}


// Determine the path for the request dispatcher.
String dispatcherPath = prepareForRendering(request, response); dispatcherPath: "/ex01/e01.jsp" response: Respon
// Obtain a RequestDispatcher for the target resource (typically a JSP).
RequestDispatcher rd = getRequestDispatcher(request, dispatcherPath); request: RequestFacade@5099 dispatcherPath
if (rd == null) {
throw new ServletException("Could not get RequestDispatcher for [" + getUrl() +
"]: Check that the corresponding file exists within your web application archive!");
}
// If already included or response already committed, perform include, else forward.
if (useInclude(request, response)) {
response.setContentType(getContentType());
if (logger.isDebugEnabled()) {
logger.debug(o: "Including [" + getUrl() + "]");
}
rd.include(request, response);
InternalResourceView › renderMergedOutputModel()
Tomcat Catalina Log
Variables
this = {InternalResourceView@5680} "org.springframework.web.servlet.view.InternalResourceView: name 'ex01/e01'; URL [/ex01/e01.jsp]"
model = {LinkedHashMap@6049} size = 0
request = {RequestFacade@5099}
response = {ResponseFacade@5100}
dispatcherPath = "/ex01/e01.jsp"