

08 SpringMVC-重定向 & 转发 & 处理静态资源

SpringMVC

一:forward && redirect 描述

- 1、一般控制器方法返回字符串类型的值会被当成逻辑视图名处理
- 2、如果返回的字符串中带 `forward:` 或 `redirect:` 前缀
- 3、SpringMVC 会对他们进行特殊处理：将 `forward:` 和 `redirect:` 当成指示符，其后的字符串作为 URL 来处理
 - `redirect:success.jsp` : 重定向操作
 - `forward:success.jsp` : 转发操作

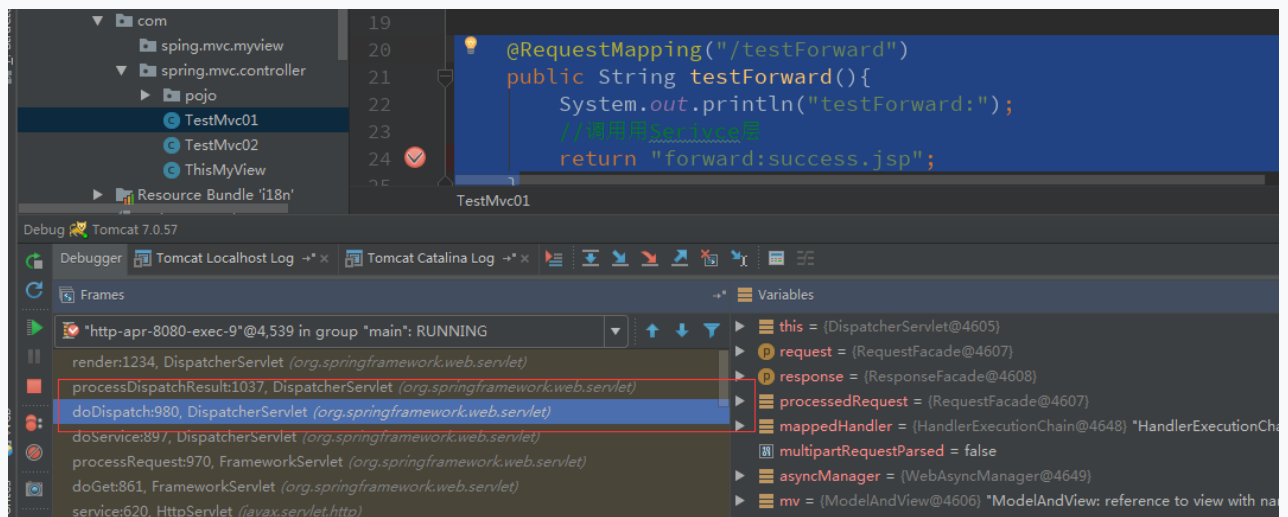
1、controller层

```
1.     @RequestMapping("/testForward")
2.     public String testForward() {
3.         System.out.println("testForward:");
4.         //调用用Service层
5.         return "forward:success.jsp";
6.     }
7.
8.     @RequestMapping("/testRredirect")
9.     public String testRredirect() {
10.        System.out.println("testRredirect:");
11.        //调用用Service层
12.        return "redirect:success.jsp";
13.    }
14.    =====
15.    <HR>
16.    <a href="testRredirect">testRredirect</a>
17.    <HR>
18.    <HR>
19.    <a href="testForward">testForward</a>
20.    <HR>
```

2、 forward: || redirect:源码解析

1、在请求的forward或者redirect处打上断点

```
1.     @RequestMapping("/testForward")
2.     public String testForward(){
3.         System.out.println("testForward:");
4.         //调用用Service层
5.         //打大花杠柳断点
6.         return "forward:success.jsp";
7.     }
```



2、在DispatcherServlet类中

```
1.     protected void doDispatch(HttpServletRequest request,...)
2.     {
3.         //其他代码省略.....
4.         /**
5.         打上断点，点击进入
6.         */
7.         this.processDispatchResult(processedRequest, response,
8. mappedHandler, mv, (Exception)dispatchException);
9.     } catch (Exception var22) {
10.         //其他代码省略.....
```

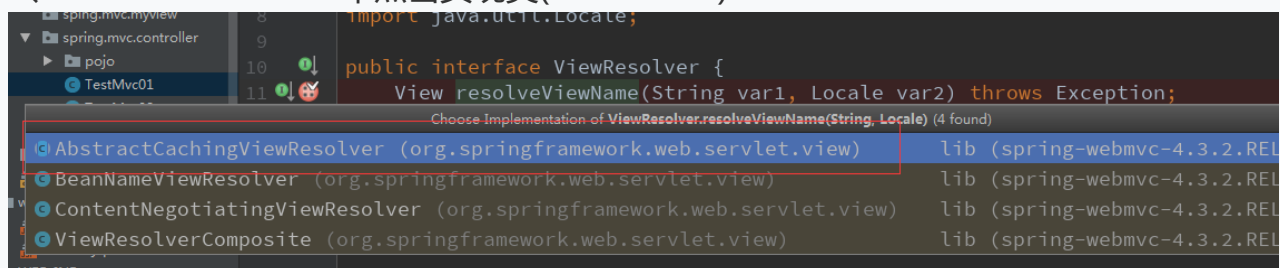
2、在processDispatchResult方法中

```
1. private void processDispatchResult(HttpServletRequest request,...) {
2.     //其他代码省略.....
3.     /**
4.     打上断点，点击进入
5.     */
6.     if (mv != null && !mv.wasCleared()) {
7.         this.render(mv, request, response);
8.         if (errorView) {
9.         }
10.    }
    //其他代码省略.....
```

2、在render方法中

```
1. protected void render(ModelAndView mv,.....
2.     //其他代码省略.....
3.     if (mv.isReference()) {
4.         /**
5.         打上断点，点击进入
6.         */
7.         view = this.resolveViewName(mv.getViewName(), mv.getModelInternal(), locale, request);
8.         if (view == null) {
9.             //其他代码省略.....
```

3、resolveViewName中点击实现类(Ctrl+Alt+B)



1、resolveViewName方法

```
1. public View resolveViewName(String viewName, Locale locale) throws
   Exception {
2.     if (!this.isCache()) {
```

```

3.         /**
4.         打上断点，点击进入
5.         */
6.         return this.createView(viewName, locale);
7.     } else {
8.         //其他代码省略.....

```

2、createView

```

1.     //Ctrl+Alt+B查询实现类
2.     protected View createView(String viewName, Locale locale) throws
    Exception {
3.         return this.loadView(viewName, locale);
4.     }

```

1、AbstractCachingViewResolver抽象类

- 1、UrlBasedViewResolver类中
- 2、进行判断请求的redirect还是forward

```

1.     protected View createView(String viewName, Locale locale) throws
    Exception {
2.         if (!this.canHandle(viewName, locale)) {
3.             return null;
4.         } else {
5.             String forwardUrl;
6.             if (viewName.startsWith("redirect:")) {
7.                 forwardUrl = viewName.substring("redirect:".length());
8.                 RedirectView view = new RedirectView(forwardUrl, this.isRedirectContextRelative(), this.isRedirectHttp10Compatible());
9.                 view.setHosts(this.getRedirectHosts());
10.                return this.applyLifecycleMethods(viewName, view);
11.            } else if (viewName.startsWith("forward:")) {
12.                forwardUrl = viewName.substring("forward:".length());
13.                return new InternalResourceView(forwardUrl);
14.            } else {
15.                return super.createView(viewName, locale);
16.            }
17.        }
18.    }

```

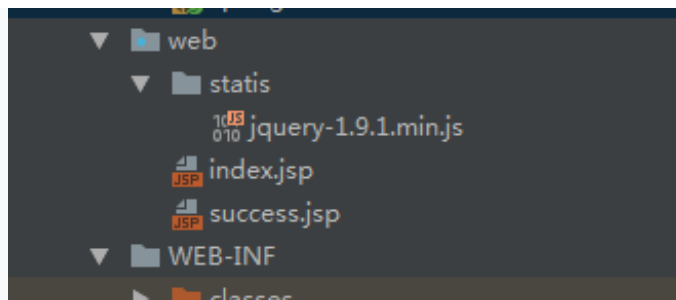
二:处理静态资源

- 1、SpringMVC 的配置文件中配置 `<mvc:default-servlet-handler/>` 的方式解决静态资源的问题
- 2、`<mvc:default-servlet-handler/>` 将在 SpringMVC 上下文中定义一个 `DefaultServletHttpRequestHandler`，它会对进入 `DispatcherServlet` 的请求进行筛查，如果发现是没有经过映射的请求
- 3、就将该请求交由WEB应用服务器默认的Servlet处理，如果不是静态资源的请求，才由 `DispatcherServlet` 继续处理
- 4、一般 WEB 应用服务器默认的 Servlet 的名称都是 `default`，若所使用的 WEB 服务器的默认 Servlet 名称不是 `default`，则需要通过 `default-servlet-name` 属性显式指定

```
1. //tomcat的xml中的进行处理
2. <servlet>
3.     <servlet-name>default</servlet-name>
4.     <servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>
5.     <init-param>
6.         <param-name>debug</param-name>
7.         <param-value>0</param-value>
8.     </init-param>
9.     <init-param>
10.        <param-name>listings</param-name>
11.        <param-value>false</param-value>
12.    </init-param>
13.    <load-on-startup>1</load-on-startup>
14. </servlet>
```

1、放入Jquery.jsjar包

- 1、导入jar包



2、请求<http://localhost:8080/statis/jquery-1.9.1.min.js>



HTTP Status 404 -

type Status report

message

description The requested resource is not available.

Apache Tomcat/7.0.57

3、在SpringMVC的xml中配置静态资源

1. `<!--配置静态资源-->`
2. `<mvc:default-servlet-handler/>`

4、请求



```
/*! jQuery v1.9.1 | (c) 2005, 2012 jQuery Foundation, Inc. | jquery.org/license
**/
@ sourceMappingURL=jquery.min.map
*(function(e,t){var n,r,i=typeof t,o=e.document,a=e.location,s=e.jQuery,u=e.$,l={},c=
[],p="1.9.1",f=c.concat,d=c.push,h=c.slice,g=c.indexOf,m=l.toString,y=l.hasOwnProperty,v=p.trim,b=function(e,t){return new b
\d+|)/.source,w=/\S+/g,T=/^[\s\uFEFF\xA0]+$/g,N=/^([^\w]|>)*#?([^\w]|>)*$/g,C=/^<(\w+)\s*\/?>(?:<\/\w+
["'\/\bfnrt]|u[\da-fA-F]{4})/g,A="/[^\r\n]*|true|false|null|-?(?:\d+\.\d+|(?=[eE])[+-]?\d+|j=|-ms-/,D=-([\da-z])/g
{(o.addEventListner||"load"===e.type||"complete"===o.readyState)&&(q(),b.ready())},q=function(){o.addEventListner?
(o.removeEventListner("DOMContentLoaded",H,!1),e.removeEventListner("load",H,!1)):o.detachEvent("onreadystatechange",H),e
[jquery:p,constructor:b,init:function(e,n,r){var i,a;if(!e)return this;if("string"===typeof e){if(i="<"===e.charAt(0)&&">"===
[null,e,null]:N.exec(e),!i||!i[1]&&n)return n|n.jquery?(n||r).find(e):this.constructor(n).find(e);if(i[1]){if(n=n.instanceo
n.ownerDocument||n:o,!0)),C.test(i[1])&&b.isPlainObject(n)}for(i in n)b.isFunction(this[i])?this[i](n[i]):this.attr(i,n[i]):
{if(a.id===i[2])return r.find(e);this.length=1,this[0]=a}return this.context=o,this.selector=e,this}return e.nodeType?(this.
(e.selector!==t&&(this.selector=e.selector,this.context=e.context),b.makeArray(e,this))),selector:"",length:0,size:function(
h.call(this)),get:function(e){return null==e?this.toArray():0>e?this[this.length+e]:this[e],pushStack:function(e){var t=b.m
t.prevObject=this,t.context=this.context,t},each:function(e,t){return b.each(this,e,t)},ready:function(e){return b.ready.pro
this.pushStack(h.apply(this,arguments))},first:function(){return this.eq(0)},last:function(){return this.eq(-1)},eq:function
this.pushStack(n>0&&t?[this[n]]:[]),map:function(e){return this.pushStack(b.map(this,function(t,n){return e.call(t,n,t)}
this.prevObject||this.constructor(null)},push:d,sort:[].sort,splice:[].splice},b.fn.init.prototype=b.fn,b.extend=b.fn.extend
{,u=1,l=arguments.length,c=1:for("boolean"===typeof s&&(c=s,s=arguments[1])||{,u=2),"object"===typeof s||b.isFunction(s)||s
in o)e=s[i],r=o[i],s=r&&(c&&r&&(b.isPlainObject(r)||b.isArray(r)))?(n=b.isArray(r))?(n?(n=!1,a=e&&b.isArray(e)?e:[]):a=e&&b.isPlainObjec
s),b.extend({noConflict:function(t){return e.$=b&&(e.$=u),t&&e.jquery===b&&(e.jquery=s),b},isReady:!1,readyWait:1,holdRead
{if(e===!0?!--b.readyWait:!b.isReady){if(!o.body)return setTimeout(b.ready);b.isReady=!0,e!==!0&&--b.readyWait>0||n.resolve
[b]),b.fn.trigger&&b(o).trigger("ready").off("ready")}},isFunction:function(e){return"function"===b.type(e)},isArray:Array.
```

