04 SpringMVC-数据模型

`SpringMVC`

一：处理模型数据

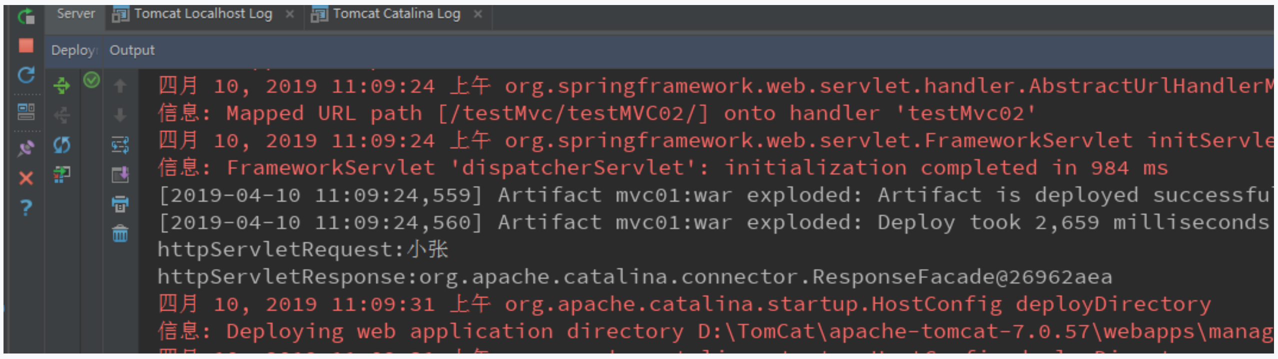
> 1、将数据返回到页面，页面进行显示出来，这时就需要模型进行处理对应的数据

二：ModelAndView

> 1、可以进行包含视图和模型信息
> 2、controller层

```
@RequestMapping("/testModelAndView")
    public ModelAndView testModelAndView(){
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.addObject("msg",new Date());
        modelAndView.setViewName("success");
        return modelAndView;
    }
======页面请求=========
 <a href="testModelAndView">testModelAndView</a>
=======success.jsp页面获取信息==========
 msg:${requestScope.msg}
```

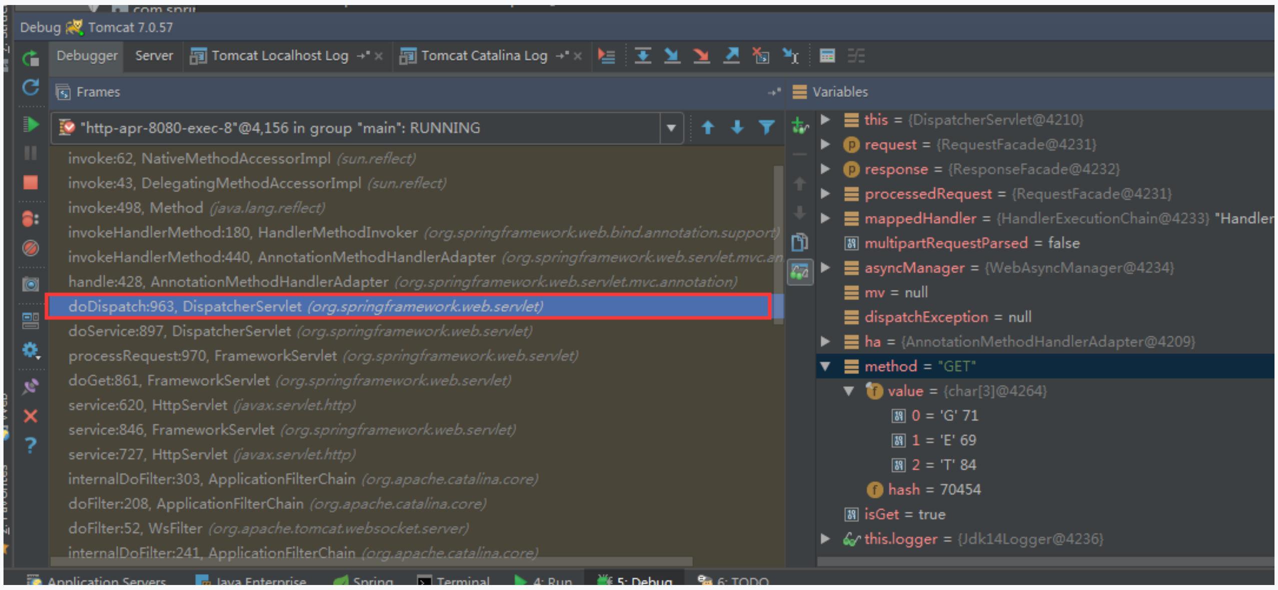
> 3、请求的结果



```
四月 10, 2019 11:09:24 上午 org.springframework.web.servlet.handler.AbstractUrlHandlerM
信息: Mapped URL path [/testMvc/testMVC02/] onto handler 'testMvc02'
四月 10, 2019 11:09:24 上午 org.springframework.web.servlet.FrameworkServlet initServle
信息: FrameworkServlet 'dispatcherServlet': initialization completed in 984 ms
[2019-04-10 11:09:24,559] Artifact mvc01:war exploded: Artifact is deployed successfu
[2019-04-10 11:09:24,560] Artifact mvc01:war exploded: Deploy took 2,659 milliseconds
httpServletRequest:小张
httpServletResponse:org.apache.catalina.connector.ResponseFacade@26962aea
四月 10, 2019 11:09:31 上午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deploying web application directory D:\TomCat\apache-tomcat-7.0.57\webapps\manag
```

1、内部原理

1、在代码`modelAndView.addObject("msg",new Date());`处进行DUG运行
2、点击DUG视图中doDispatch



```
"http-apr-8080-exec-8"@4,156 in group "main": RUNNING
invoke:62, NativeMethodAccessorImpl (sun.reflect)
invoke:43, DelegatingMethodAccessorImpl (sun.reflect)
invoke:498, Method (java.lang.reflect)
invokeHandlerMethod:180, HandlerMethodInvoker (org.springframework.web.bind.annotation.support)
invokeHandlerMethod:440, AnnotationMethodHandlerAdapter (org.springframework.web.servlet.mvc.an
handle:428, AnnotationMethodHandlerAdapter (org.springframework.web.servlet.mvc.annotation)
doDispatch:963, DispatcherServlet (org.springframework.web.servlet)
doService:897, DispatcherServlet (org.springframework.web.servlet)
processRequest:970, FrameworkServlet (org.springframework.web.servlet)
doGet:861, FrameworkServlet (org.springframework.web.servlet)
service:620, HttpServlet (javax.servlet.http)
service:846, FrameworkServlet (org.springframework.web.servlet)
service:727, HttpServlet (javax.servlet.http)
internalDoFilter:303, ApplicationFilterChain (org.apache.catalina.core)
doFilter:208, ApplicationFilterChain (org.apache.catalina.core)
doFilter:52, WsFilter (org.apache.tomcat.websocket.server)
internalDoFilter:241, ApplicationFilterChain (org.apache.catalina.core)
```

```
this = {DispatcherServlet@4210}
request = {RequestFacade@4231}
response = {ResponseFacade@4232}
processedRequest = {RequestFacade@4231}
mappedHandler = {HandlerExecutionChain@4233} "Handler
multipartRequestParsed = false
asyncManager = {WebAsyncManager@4234}
mv = null
dispatchException = null
ha = {AnnotationMethodHandlerAdapter@4209}
method = "GET"
    value = {char[3]@4264}
        0 = 'G' 71
        1 = 'E' 69
        2 = 'T' 84
    hash = 70454
isGet = true
this.logger = {Jdk14Logger@4236}
```

3、进入DispatcherServlet类中有个doDispatch方法

```
    protected void doDispatch(HttpServletRequest request, HttpServletResponse response) throws Exception {
        HttpServletRequest processedRequest = request;
        HandlerExecutionChain mappedHandler = null;
        boolean multipartRequestParsed = false;
        WebAsyncManager asyncManager = WebAsyncUtils.getAsyncManager(request);

        try {
            try {
                ModelAndView mv = null;
//............代码省略
/**
获取模型视图
*/
                    mv = ha.handle(processedRequest, response, mappedHandler.getHandler());
                    if (asyncManager.isConcurrentHandlingStarted()) {
                        return;
                    }
    //............代码省略
  /**
    进行处理结果---点击进入
   */
this.processDispatchResult(processedRequest, response, mappedHandler,
mv, (Exception)dispatchException);
```

4、点击进入this.processDispatchResult()

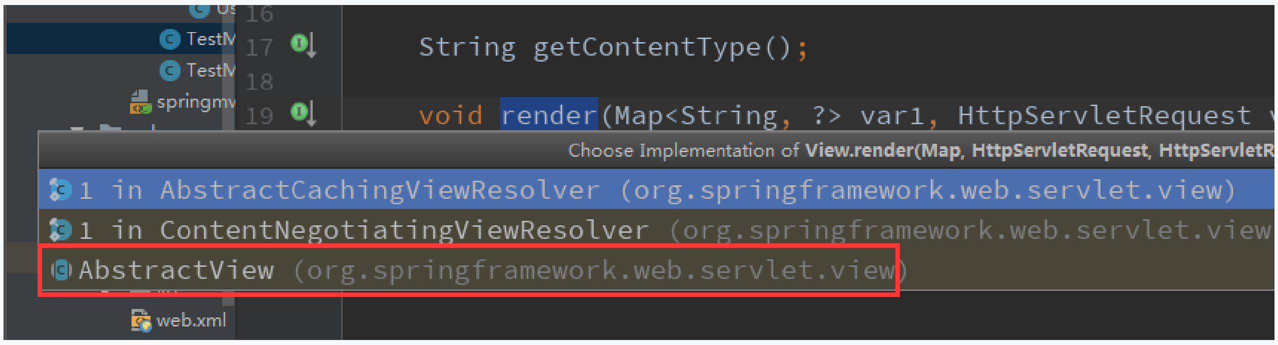
```
 private void processDispatchResult(HttpServletRequest request,
HttpServletResponse response, HandlerExecutionChain mappedHandler,
ModelAndView mv, Exception exception) throws Exception {
        boolean errorView = false;
  //............代码省略
 if (mv != null && !mv.wasCleared()) {
 //进行渲染视图--点击进入
            this.render(mv, request, response);
            if (errorView) {
                WebUtils.clearErrorRequestAttributes(request);
            }
  //............代码省略
```

5、点击进入his.render(....)

```
protected void render(ModelAndView mv, HttpServletRequest request, Htt
pServletResponse response) throws Exception {
        Locale locale = this.localeResolver.resolveLocale(request);
     //............代码省略
     try {
            if (mv.getStatus() != null) {
                response.setStatus(mv.getStatus().value());
            }
           //进行渲染视图--点击进入
            view.render(mv.getModelInternal(), request, response);
       //............代码省略
```

5、点击进入his.render(....)

```
public interface View {
        //............代码省略
        //Ctrl+Alt+B  返回实现类
    void render(Map<String, ?> var1, HttpServletRequest var2, HttpServletResponse var3) throws Exception;
}
```



```
    String getContentType();

    void render(Map<String, ?> var1, HttpServletRequest v
```

Choose Implementation of **View.render(Map, HttpServletRequest, HttpServletR**

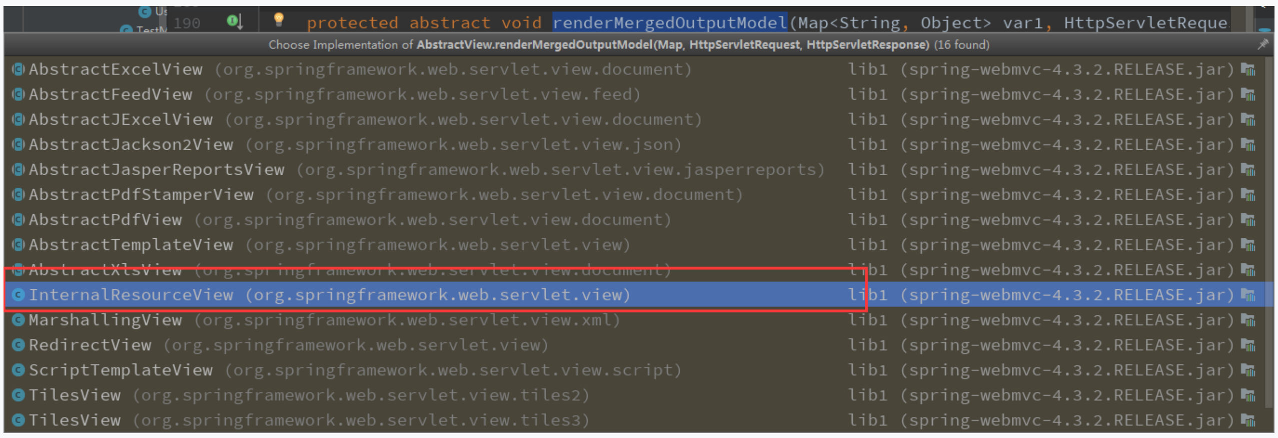
```
1 in AbstractCachingViewResolver (org.springframework.web.servlet.view)
1 in ContentNegotiatingViewResolver (org.springframework.web.servlet.view
AbstractView (org.springframework.web.servlet.view)
```

5、点击进入AbstractView

```
public void render(Map<String, ?> model, HttpServletRequest request, HttpServletResponse response) throws Exception {
        //............代码省略
        //进行输出结果---点击进行
         this.renderMergedOutputModel(mergedModel, this.getRequestToExpose(request), response);
    }
```

6、抽象类renderMergedOutputModel

```
 //Ctrl+Alt+B  返回实现类 InternalResourceView
    protected abstract void renderMergedOutputModel(Map<String, Object> var1, HttpServletRequest var2, HttpServletResponse var3) throws Exception;
```



```
190     protected abstract void renderMergedOutputModel(Map<String, Object> var1, HttpServletReque
Choose Implementation of AbstractView.renderMergedOutputModel(Map, HttpServletRequest, HttpServletResponse) (16 found)
AbstractExcelView (org.springframework.web.servlet.view.document)            lib1 (spring-webmvc-4.3.2.RELEASE.jar)
AbstractFeedView (org.springframework.web.servlet.view.feed)                 lib1 (spring-webmvc-4.3.2.RELEASE.jar)
AbstractJExcelView (org.springframework.web.servlet.view.document)           lib1 (spring-webmvc-4.3.2.RELEASE.jar)
AbstractJackson2View (org.springframework.web.servlet.view.json)             lib1 (spring-webmvc-4.3.2.RELEASE.jar)
AbstractJasperReportsView (org.springframework.web.servlet.view.jasperreports) lib1 (spring-webmvc-4.3.2.RELEASE.jar)
AbstractPdfStamperView (org.springframework.web.servlet.view.document)       lib1 (spring-webmvc-4.3.2.RELEASE.jar)
AbstractPdfView (org.springframework.web.servlet.view.document)              lib1 (spring-webmvc-4.3.2.RELEASE.jar)
AbstractTemplateView (org.springframework.web.servlet.view)                  lib1 (spring-webmvc-4.3.2.RELEASE.jar)
AbstractXlsView (org.springframework.web.servlet.view.document)              lib1 (spring-webmvc-4.3.2.RELEASE.jar)
InternalResourceView (org.springframework.web.servlet.view)                  lib1 (spring-webmvc-4.3.2.RELEASE.jar)
MarshallingView (org.springframework.web.servlet.view.xml)                   lib1 (spring-webmvc-4.3.2.RELEASE.jar)
RedirectView (org.springframework.web.servlet.view)                          lib1 (spring-webmvc-4.3.2.RELEASE.jar)
ScriptTemplateView (org.springframework.web.servlet.view.script)             lib1 (spring-webmvc-4.3.2.RELEASE.jar)
TilesView (org.springframework.web.servlet.view.tiles2)                      lib1 (spring-webmvc-4.3.2.RELEASE.jar)
TilesView (org.springframework.web.servlet.view.tiles3)                      lib1 (spring-webmvc-4.3.2.RELEASE.jar)
```

6、renderMergedOutputModel

```
    protected void renderMergedOutputModel(Map<String, Object> model, HttpServletRequest request, HttpServletResponse response) throws Exception {
    //将模型放入到请求域中，点击
        this.exposeModelAsRequestAttributes(model, request);
```

7、exposeModelAsRequestAttributes

```
    protected void exposeModelAsRequestAttributes(Map<String, Object> model, HttpServletRequest request) throws Exception {
        Iterator var3 = model.entrySet().iterator();
//使用的循环将map进行遍历，在将对应的value放入到域中
        while(var3.hasNext()) {
            Entry<String, Object> entry = (Entry)var3.next();
            String modelName = (String)entry.getKey();
            Object modelValue = entry.getValue();
            if (modelValue != null) {
            //放入到域中
                request.setAttribute(modelName, modelValue);
```

2、返回DispatcherServlet

1、返回DispatcherServlet中的render查询视图

```
//点击模型mv.getModelInternal()返回的是model
view.render(mv.getModelInternal(), request, response);
```

> 2、返回的model

```
    protected Map<String, Object> getModelInternal() {
        return this.model;
    }
```

> 3、返回Controller中的ModelAndView中ADD方法(点击)

```
       ModelAndView modelAndView = new ModelAndView();
       //点击
         modelAndView.addObject("msg",new Date());
```

```
     public ModelAndView addObject(String attributeName, Object attributeValue) {
     //在点击getModelMap()返回的也是一个model
         this.getModelMap().addAttribute(attributeName, attributeValue);
         return this;
     }
```

```
     public ModelMap getModelMap() {
         if (this.model == null) {
             this.model = new ModelMap();
         }
         return this.model;
     }
```

> 4、ModelAndView中的Model中的数据会进行通过遍历的方式，进行放入到request域对象中。

三：Map模型系列

> 1、Spring MVC 在内部使用了一个org.springframework.ui.Model 接口存储模型数据
>
> 2、Spring MVC 在调用方法前会创建一个隐含的模型对象作为模型数据的存储容器

> 3、如果方法传入的参数是 Map 或 Model 类型，Spring MVC 会将隐含模型的引用传递给这些参数
>
> 4、在方法体内，开发者可以通过这个参数对象访问到模型中的所有数据，也可以向模型中添加新的属性数据

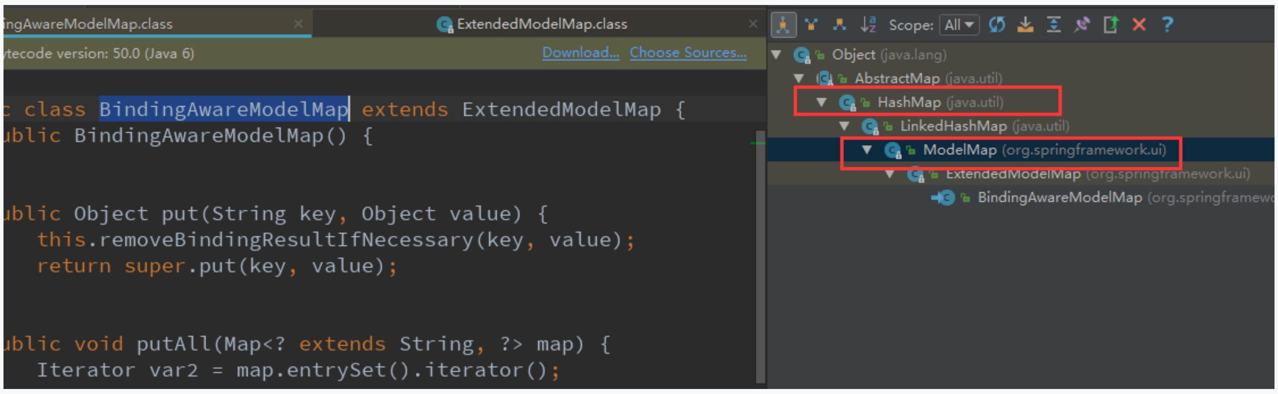
1、controller层

> 1、使用Map类型来进行编辑模型

```
    /**
     * 使用map来作为模型
     * @param map
     * @return
     */
    @RequestMapping("/testMap")
    public String testMap(Map<String, Object> map){
        System.out.println(map.getClass().getName());
        map.put("key",new Date());
        return "success";
    }
    /**
     * 使用ModelMap 来作为模型
     * @param map
     * @return
     */
    @RequestMapping("/testModelMap")
    public String testModelMap(ModelMap map){
        map.addAttribute("key",new Date());
        return "success";
    }

    /**
     * 使用model来作为模型
     * @param model
     * @return
     */
    @RequestMapping("/testModel")
    public String testModel(Model model){

        model.addAttribute("key", new Date());//传入Request域
```

```
            return "success";
        }

==============请求页面=================
   <a href="testModelMap">testModelMap</a>
   <HR>
   <a href="testMap">testMap</a>
   <HR>
   <a href="testModel">testModel</a>
=============获取值页面==========================
<HR>
key:${requestScope.key}
```

2、Map模型继承图，打印出map类地址名称



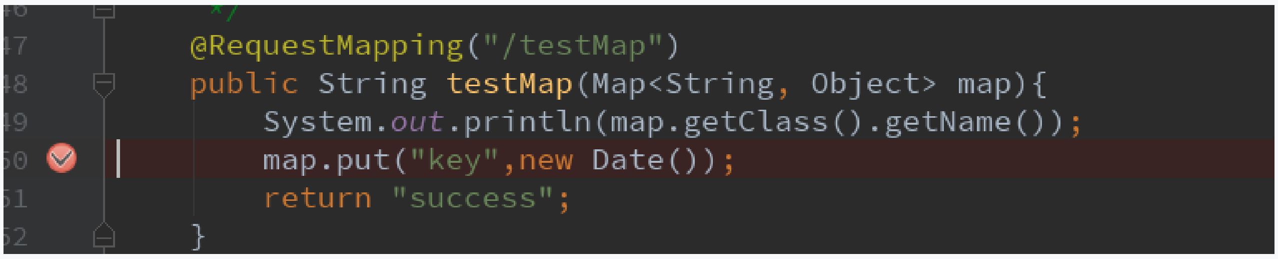
```
ingAwareModelMap.class        ExtendedModelMap.class
ytecode version: 50.0 (Java 6)        Download...  Choose Sources...

c class BindingAwareModelMap extends ExtendedModelMap {
ublic BindingAwareModelMap() {


ublic Object put(String key, Object value) {
   this.removeBindingResultIfNecessary(key, value);
   return super.put(key, value);


ublic void putAll(Map<? extends String, ?> map) {
   Iterator var2 = map.entrySet().iterator();
```

```
Scope: All
▼ Object (java.lang)
  ▼ AbstractMap (java.util)
    ▼ HashMap (java.util)
      ▼ LinkedHashMap (java.util)
        ▼ ModelMap (org.springframework.ui)
          ▼ ExtendedModelMap (org.springframework.ui)
              BindingAwareModelMap (org.springframew
```

3、Model是ExtendedModelMap的实现接口

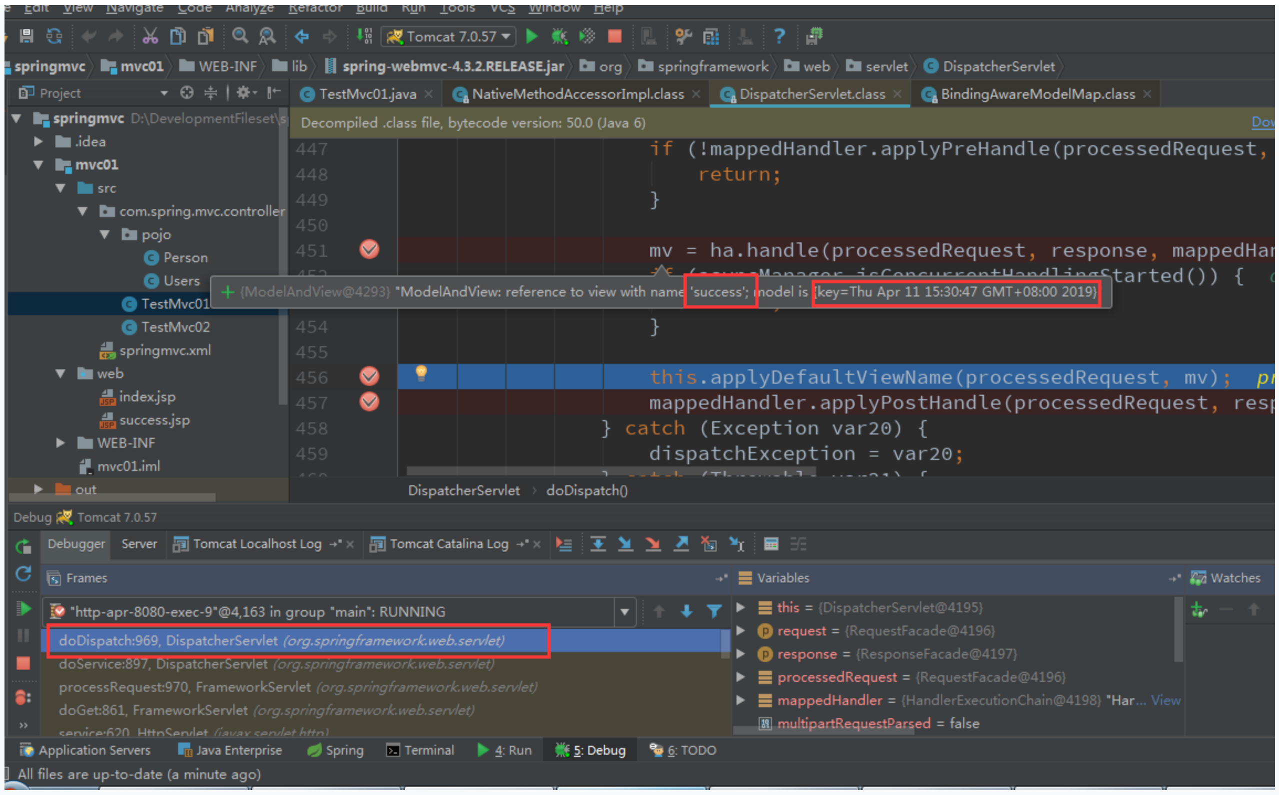
2、原理

1、在传入模型处打入DUG断点(如map.put("key",new Date());)



```
        */
    @RequestMapping("/testMap")
    public String testMap(Map<String, Object> map){
        System.out.println(map.getClass().getName());
        map.put("key",new Date());
        return "success";
    }
```

2、查询执行结果



Decompiled .class file, bytecode version: 50.0 (Java 6)

```
                if (!mappedHandler.applyPreHandle(processedRequest,
                    return;
                }

                mv = ha.handle(processedRequest, response, mappedHan
                if (asyncManager.isConcurrentHandlingStarted()) {

                }

                this.applyDefaultViewName(processedRequest, mv);
                mappedHandler.applyPostHandle(processedRequest, resp
            } catch (Exception var20) {
                dispatchException = var20;
```

```
{ModelAndView@4293} "ModelAndView: reference to view with name 'success'; model is {key=Thu Apr 11 15:30:47 GMT+08:00 2019}
```

DispatcherServlet › doDispatch()

```
"http-apr-8080-exec-9"@4,163 in group "main": RUNNING
doDispatch:969, DispatcherServlet (org.springframework.web.servlet)
doService:897, DispatcherServlet (org.springframework.web.servlet)
processRequest:970, FrameworkServlet (org.springframework.web.servlet)
doGet:861, FrameworkServlet (org.springframework.web.servlet)
service:620, HttpServlet (javax.servlet.http)
```

```
this = {DispatcherServlet@4195}
request = {RequestFacade@4196}
response = {ResponseFacade@4197}
processedRequest = {RequestFacade@4196}
mappedHandler = {HandlerExecutionChain@4198} "Har... View
multipartRequestParsed = false
```

All files are up-to-date (a minute ago)

四：@SessionAttributes

> 1、若希望在多个请求之间共用某个模型属性数据，则可以在控制器类上标注一个@SessionAttributes，而不在方法上使用
>
> 2、Spring MVC 将在模型中对应的属性暂存到 HttpSession 中
>
> 3、@SessionAttributes除了可以通过属性名指定需要放到会话中的属性外，还可以通过模型属性的对象类型指定哪些模型属性需要放到会话中

1、controller层

```
@SessionAttributes(value = {"sessionKey","sessionKey1"},types = {String.class,Date.class})
@Controller
public class TestMvc01 {
    @RequestMapping("/testSessionAttributes")
    public String testSessionAttributes(ModelMap map){
        map.addAttribute("sessionKey",new Date());
        map.addAttribute("sessionKey1",new Date());
        map.addAttribute("name","小王");

           RequestAttributes requestAttributes = RequestContextHolder.getRequestAttributes();
        Object sessionKey = requestAttributes.getAttribute("sessionKey", RequestAttributes.SCOPE_SESSION);
        System.out.println("date"+sessionKey);


        return "success";
    }
```

1、页面代码

```
================请求页面===================
  <a href="testSessionAttributes">testSessionAttributes</a>
  <HR>
============获取值的页面===============
sessionScope-sessionKey1:${sessionScope.sessionKey1}
<HR>
requestScope-sessionKey1:${requestScope.sessionKey1}
<HR>
sessionScope-sessionKey:${sessionScope.sessionKey}
<HR>
requestScope-sessionKey:${requestScope.sessionKey}
<HR>
sessionScope-name:${sessionScope.name}
<HR>
requestScope-name:${requestScope.name}
<HR>
```