

09 SpringMVC-数据转换 & 数据格式化 & 数据校验

SpringMVC

一：数据绑定流程

- 1、. Spring MVC 将 ServletRequest 对象及目标方法的入参实例传递给 WebDataBinderFactory 实例，以创建 DataBinder 实例对象
- 2、DataBinder 调用装配在 Spring MVC 上下文中的 ConversionService 组件进行数据类型转换、数据格式化工作并将 Servlet 中的请求信息填充到入参对象中
- 3、调用 Validator 组件对已经绑定了请求消息的入参对象进行数据合法性校验，并最终生成数据绑定结果 BindingData 对象
- 4、Spring MVC 抽取 BindingResult 中的入参对象和校验错误对象，将它们赋给处理方法的响应入参

二:源码查询

1、controller层代码

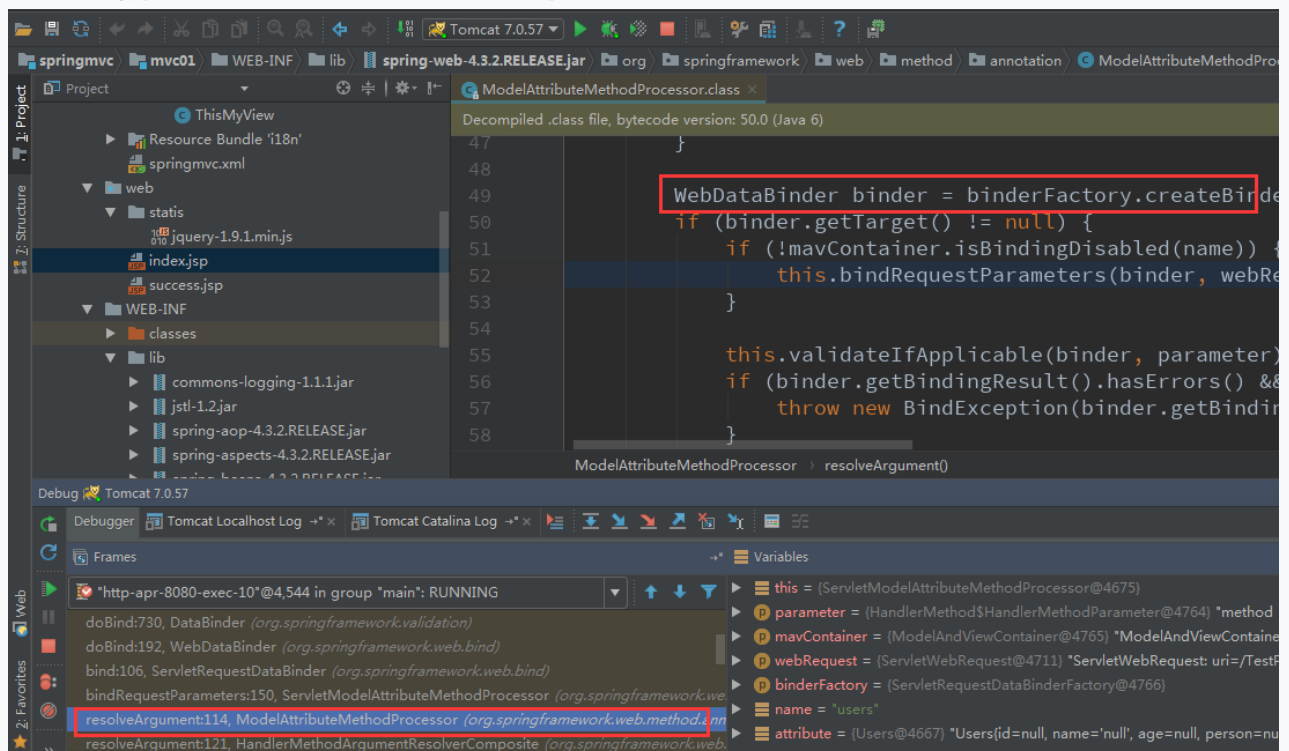
```
1.      @RequestMapping("/TestPojo")
2.      public String TestPojo(Users users){
3.          System.out.println("users:"+users);
4.          return "success";
5.      }
6.      =====页面代码=====
7.      <form ACTION="TestPojo" METHOD="post">
8.          name:<INPUT TYPE="text" NAME="name"><br/>
9.          age:<INPUT TYPE="text" NAME="age"><br/>
10.         <INPUT TYPE="submit" VALUE="提交"><br/>
11.     </form>
```

```

12.  =====实体类DUG=====
13.     public Integer getAge() {
14.         //在此处DUG
15.         return age;
16.     }
17.     public void setAge(Integer age) {
18.         //在此处DUG
19.         this.age = age;
20.     }

```

2、点击源码查询WebDataBinder对象



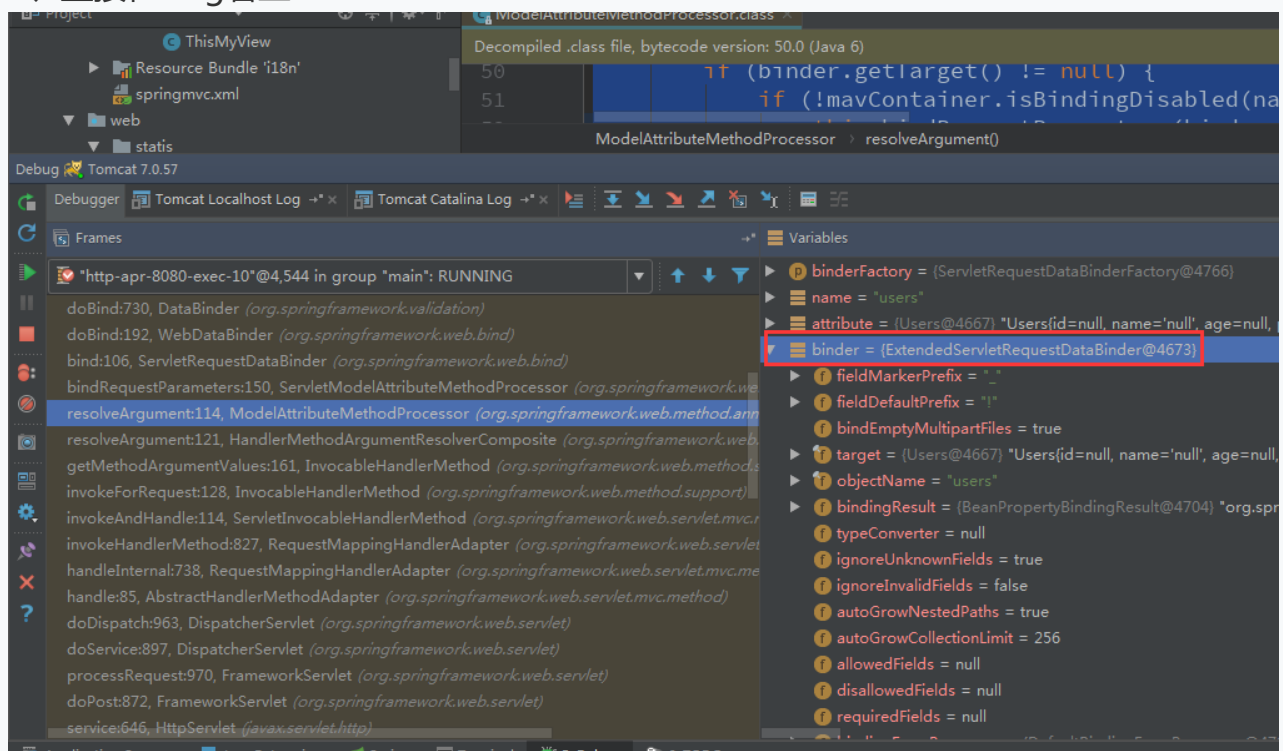
```

1.     //获取对象并传入request请求, 和目标类型对象参数与Model属性名
2.     WebDataBinder binder = binderFactory.createBinder(webRequest,
3.         attribute, name);
4.     if (binder.getTarget() != null) {
5.         if (!mavContainer.isBindingDisabled(name)) {
6.             //数据类型的绑定
7.             this.bindRequestParameters(binder, webRequest);
8.         }
9.         //数据的效验和格式转换
10.        this.validateIfApplicable(binder, parameter);

```

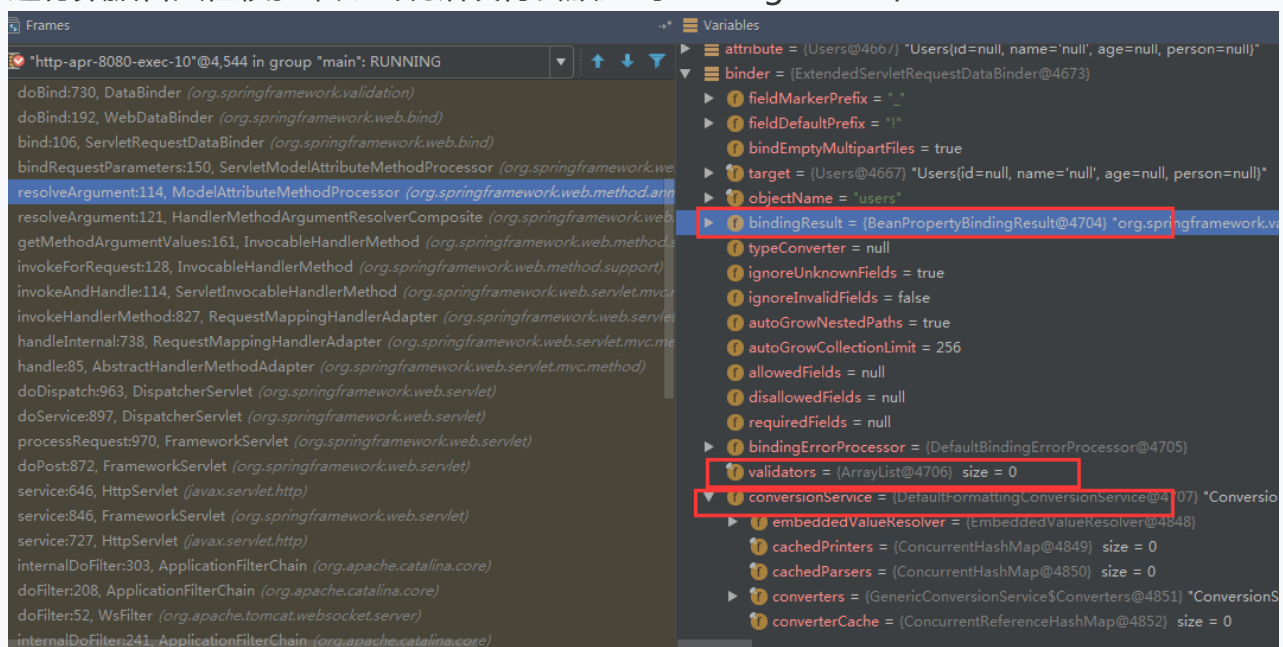
3、那么数据类型是怎么绑定类内？点击 this.bindRequestParameters(源码费劲)

4、直接在Dug看上binder



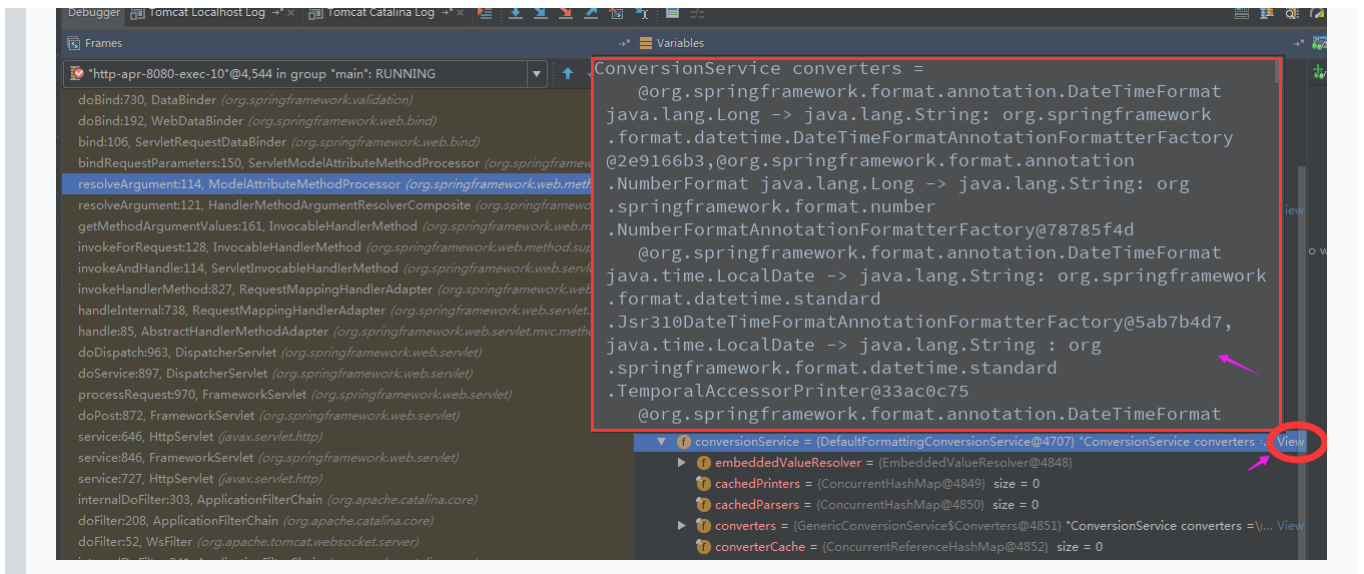
5、可以看到ConversionService 组件进行数据类型转换、数据格式化工作

6、Validator 组件对已经绑定了请求消息的入参对象进行数据合法性校验，如出现错误将会放入到BindingResult中



7、查询ConversionService转换的类型，可以看到右很多如下,有

个StringToNumberConverterFactory 类字符串转换数字



```
1. ConversionService converters =
2. //其他省略.....
3.     java.lang.String -> java.lang.Boolean : org.spring.....
4.     java.lang.String -> java.lang.Character : org.spr.....
5.     java.lang.String -> java.lang.Enum : org.springfr.....
6.     java.lang.String -> java.lang.Number : org.spring.....
7.     java.lang.String -> java.nio.charset.Charset : or.....
8.     java.lang.String -> java.time.Duration: org.sprin.....
9.     java.lang.String -> java.time.Instant: org.spring.....
10.    java.lang.String -> java.time.MonthDay: org.sprin.....
11.    java.lang.String -> java.time.Period: org.springf.....
12.    java.lang.String -> java.time.YearMonth: org.spri.....
13.    java.lang.String -> java.util.Currency : org.spri.....
14.    java.lang.String -> java.util.Locale : org.spring.....
15.    java.lang.String -> java.util.Properties : org.sp.....
16.    java.lang.String -> java.util.TimeZone : org.spri.....
17.    java.lang.String -> java.util.UUID : org.springfr.....
18.    //其他省略.....
```

1、StringToNumberConverterFactory类

1、字符串转换数字源码

```
1.     public <T extends Number> Converter<String, T> getConverter(Class<T
2.     > targetType) {
3.         //打DUG断点，可进行查询.....
4.         return new StringToNumberConverterFactory.StringToNumber(target
```

```
4.         Type);  
           }
```

三:自定义类型转换器

- 1、ConversionService 是 Spring 类型转换体系的核心接口
- 2、可以利用 ConversionServiceFactoryBean 在 Spring 的 IOC 容器中定义一个 ConversionService ; Spring 将自动识别出IOC 容器中的 ConversionService
- 3、并在 Bean 属性配置及Spring MVC 处理方法入参绑定等场合使用它进行数据的转换
- 4、可通过 ConversionServiceFactoryBean 的 converters 属性注册自定义的类型转换器

1、代码

1、controller层

```
1.         @RequestMapping("/testConversionService")  
2.         public String testConversionService(@RequestParam("users") Users u  
3.         sers) {  
4.             System.out.println("字符串转换为对象:"+users);  
5.             //调用用Service层  
6.             return "success";  
7.         }  
8.         =====页面=====  
9.         <form ACTION="testConversionService" METHOD="post">  
10.            <!--输入的字符串:12,是,12-->  
11.            users:<INPUT TYPE="text" NAME="users"><br/>  
12.            <INPUT TYPE="submit" VALUE="提交"><br/>  
            </form>
```

2、创建一个类自定义一个ConversionServiceBean

```

1. package com.spring.mvc.controller;
2. import com.spring.mvc.controller.pojo.Users;
3. import org.springframework.core.convert.converter.Converter;
4. import org.springframework.stereotype.Component;
5.
6. @Component
7. public class ThisConversionService implements Converter<String,Users> {
8.     @Override
9.     public Users convert(String s) {
10.         if(s!=null){
11.             /**
12.              页面输入的参数，必须与这里一致(12,是,12)
13.              */
14.             String[] split = s.split(",");
15.             if(split!=null && split.length==3){
16.                 Users users = new Users();
17.                 users.setAge(Integer.parseInt(split[2]));
18.                 users.setName(split[1]);
19.                 System.out.println("users"+users);
20.                 return users;
21.             }
22.         }
23.         return null;
24.     }
25. }

```

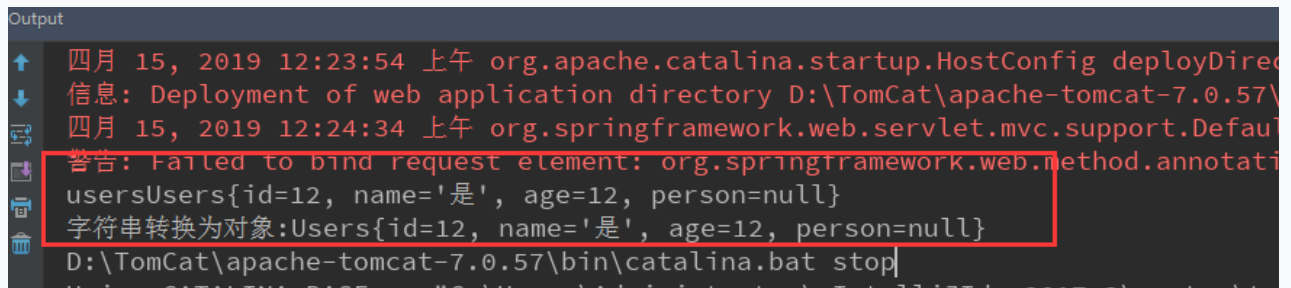
3、配置SpringMVC.xml文件进行注入自定义的Bean

```

1.      <!--这里需要进行指定，你配置的conversionServiceBean-->
2.      <mvc:annotation-driven conversion-service="conversionService"></mvc
:annotation-driven>
3.
4.      <!--配置ConversionServiceFactoryBean -->
5.      <bean
6.      class="org.springframework.context.support.ConversionServiceFactoryBean"
7.          id="conversionService">
8.          <property name="converters">
9.              <set>
10.                  <ref bean="thisConversionService"/>
11.              </set>
12.          </property>
13.      </bean>

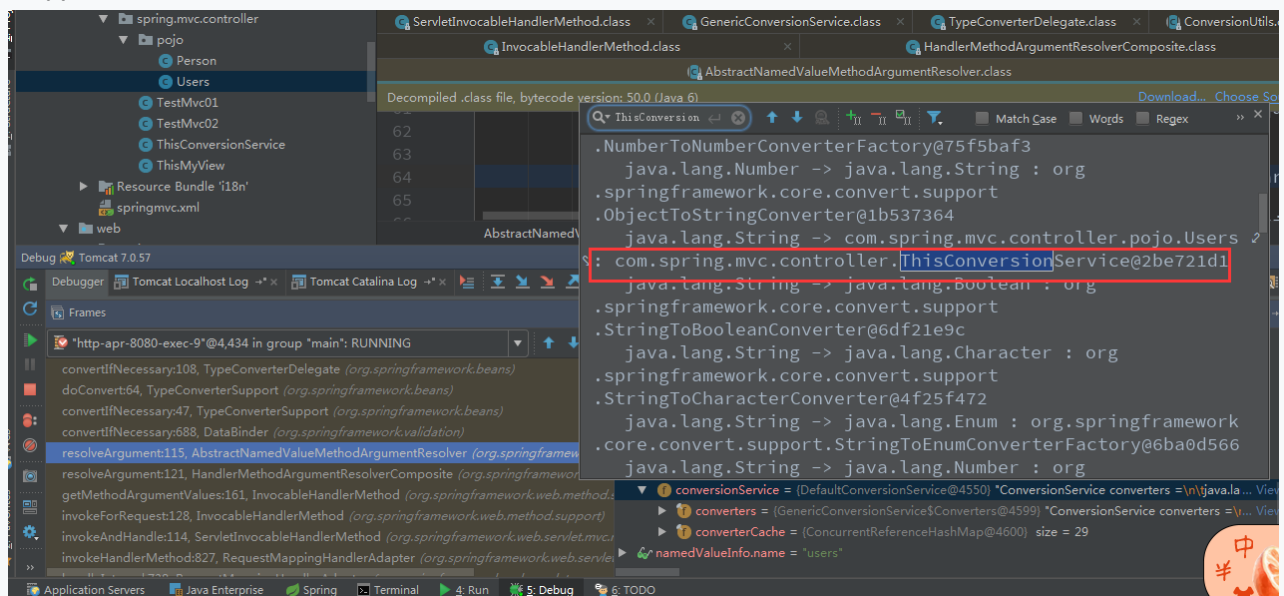
```

4、测试



```
四月 15, 2019 12:23:54 上午 org.apache.catalina.startup.HostConfig deployDirec
信息: Deployment of web application directory D:\TomCat\apache-tomcat-7.0.57\
四月 15, 2019 12:24:34 上午 org.springframework.web.servlet.mvc.support.Default
警告: Failed to bind request element: org.springframework.web.method.annotati
usersUsers{id=12, name='是', age=12, person=null}
字符串转换为对象:Users{id=12, name='是', age=12, person=null}
D:\TomCat\apache-tomcat-7.0.57\bin\catalina.bat stop
```

5、之前的ConversionServiceBean一样的使用，可用使用DUG断点查询，断点和之前一样



四:mvc:annotation-driven描述

1、<mvc:annotation-driven /> 会自动注册RequestMappingHandlerMapping、RequestMappingHandlerAdapter、ExceptionHandlerExceptionResolver 这个鞋 Bean

2、并且还提供支持

---支持使用 ConversionService 实例对表单参数进行类型转换

---支持使用 @NumberFormat annotation、@DateTimeFormat

注解完成数据类型的格式化

---支持使用 @Valid 注解对 JavaBean 实例进行 JSR 303 验证

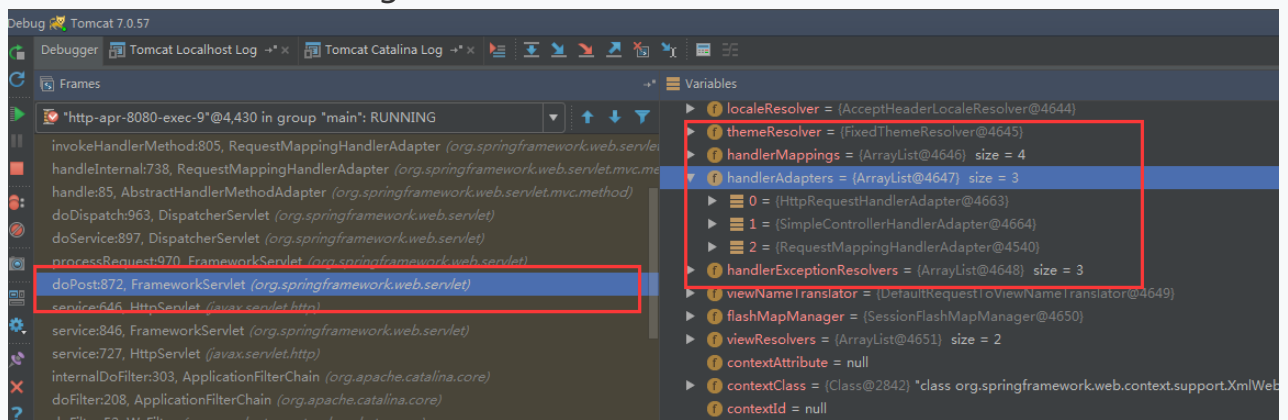
---支持使用 @RequestBody 和 @ResponseBody 注解

1、验证

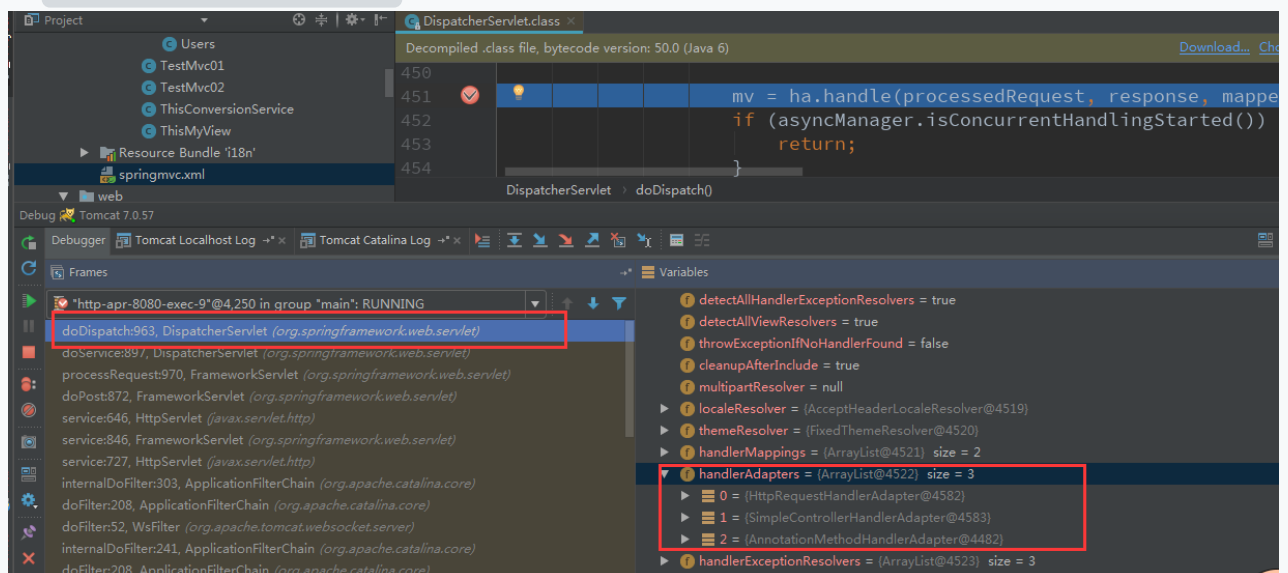
1、如我刚刚配置了自定义的转换q

1. `<mvc:default-servlet-handler/>`
2. `<mvc:annotation-driven conversion-service="conversionService"></mvc:annotation-driven>`

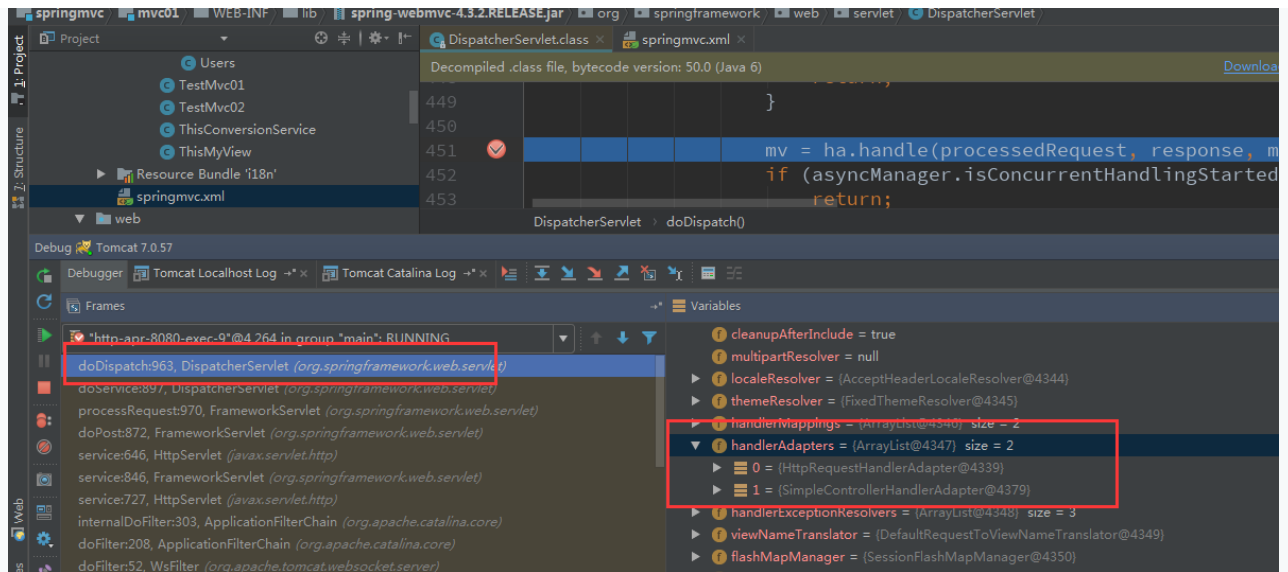
2、还是在实体类上的set get 方法上加上DUG进行测试查询源码



3、注释掉 `<mvc:default-servlet-handler/>`和 `<mvc:annotation-driven` 注意 `<mvc:view-controller` 也的注释掉



4、配置了 `<mvc:default-servlet-handler/>` 但没有配置 `<mvc:annotation-driven/>`



5、既配置了 `<mvc:default-servlet-handler/>` 又配置 `<mvc:annotation-driven/>`

