链滴

SpringMVC 回顾笔记 (十)---- 数据转换 & 数据格式化 & 数据校验

作者：pride

原文链接：https://ld246.com/article/1542526959868

来源网站：链滴

● 数据转换

大致流程:

1.Spring MVC 主框架将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SpringMVC上下文中的ConversionService组件进行数据

类型转换、数据格式化工作。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

基本知识点:

ConversionService是Spring类型转换体系的核心接口。

可以利用ConversionServiceFactoryBean 在Spring的IOC容器定义一个ConversionService.

Spring将自动识别出IOC容器中的ConversionService，并在Bean属性配置及Spring MVC处理方法参绑定等场合使用它进行数据的转换

可通过ConversionServiceFactoryBean的 converters 属性注册自定义的类型转换器

```
<!-- 配置ConversionService-->
   <bean id="conversionService" class="org.springframework.format.support.FormattingCon
ersionServiceFactoryBean">
   <property name="converters">
      <set>
         <ref bean="stringConverterEmployee"></ref>
      </set>
   </property>
   </bean>
```

Spring定义了3种类型的转换器接口，实现任意一个转换器接口都可以作为自定义转换器注册到ConversionServiceFactroyBean中:

-Converter<S，T>: 将S类型对象转为T类型对象

-ConverterFactory: 将相同系列多个“同质”Converter 封装在一起。如果希望将一种类型的对象转为另一种类型及其子类的对象（例如将String转换为Number及Number子类（Integer、Long、Double等）对象）可使用该转换器工厂类

-GenericConverter: 会根据源类对象及目标类对象所在的宿主类中的上下文信息进行类型转换

<mvc:annotation-driven conversion-service=“conversionService” />会将自定义的 ConversioService 注册到Spring MVC 的上下文中

● mvc:annotation-driven

<mvc:annotation-driven/>会自动注册RequestMappingHandlerMapping 、RequestMappingHandlerAdapter与ExceptionHandlerExceptionResolver 三个bean。

还将提供以下支持:

-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JSR303验证

-支持使用@RequestBody和@ResponseBody 注解

● @InitBinder

由 @InitBinder 标识的方法，可以对 WebDataBinder 对象进行初始化。WebDataBinder 是 DataBinder 的子类，用于完成由表单字段到 JavaBean 属性的绑定

• @InitBinder方法不能有返回值，它必须声明为void。

• @InitBinder方法的参数通常是是 WebDataBinder

● 数据等格式化

1. 数据格式化

 Spring 在格式化模块中定义了一个实现ConversionService 接口的 FormattingConversionService 实现类，该实现类扩展了 GenericConversionService，因此它既具有类型转换的功能又具有格式化的功能.

FormattingConversionService 拥有一个FormattingConversionServiceFactroyBean 工厂类，后用于在 Spring 上下文中构造前者.

其中FormattingConversionServiceFactroyBean 内部已经注册了：

– NumberFormatAnnotationFormatterFactroy：支持对数字类型的属性用 @NumberFormat 注解

– JodaDateTimeFormatAnnotationFormatterFactroy：支持对日期类型的属性使用 @DateTimeFormat 注解

• 装配了 FormattingConversionServiceFactroyBean 后，就可 以在 Spring MVC 入参绑定及模型据输出时使用注解驱动了。

<mvc:annotation-driven/> 默认创建的ConversionService 实例即为FormattingConversionServicFactroyBean

2. 日期格式化

@DateTimeFormat 注解可对java.util.Date、java.util.Calendar、java.long.Long 时间类型进行标：

– pattern 属性：类型为字符串。

指定解析/格式化字段数据的模式，如：” yyyy-MM-dd hh:mm:ss”

3. 数值格式化

@NumberFormat 可对类似数字类型的属性进行标注，它拥有两个互斥的属性：

– style：类型为 NumberFormat.Style。

用于指定样式类型，包括三种：Style.NUMBER（正常数字类型）、Style.CURRENCY（货币类型）、Style.PERCENT（百分数类型）

– pattern：类型为 String，自定义样式，

如patter="#,###";

● JSR303

JSR 303 是 Java 为 Bean 数据合法性校验提供的标准框架，它已经包含在 JavaEE 6.0 中.

JSR 303 通过在 Bean 属性上标注类似于 @NotNull、@Max等标准的注解指定校验规则，并通过标的验证接口对 Bean进行验证.

Spring 本身并没有提供 JSR303 的实现，所以必须将JSR303 的实现者的 jar 包放到类路径下,<mvc:a

notation-driven/> 会默认装配好一个LocalValidatorFactoryBean，通过在处理方法的入参上标注 valid 注解即可让 Spring MVC 在完成数据绑定后执行数据校验的工作.

Spring MVC 是通过对处理方法签名的规约来保存校验结果的：前一个表单/命令对象的校验结果保到随后的入参中，这个保存校验结果的入参必须是BindingResult 或Errors 类型通俗来说就是 @Valid 注解的属性后面一定是 BindingResult 不然会报错.

在页面上显示错误信息:

Spring MVC 除了会将表单/命令对象的校验结果保存到对应的 BindingResult 或 Errors 对象中外，会将所有校验结果保存到 “隐含模型” 隐含模型中的所有数据最终将通过 HttpServletRequest 的属列表暴露给 JSP 视图对象，因此在 JSP 中可以获取错误信息,在 JSP 页面上可通过 <form:errors path “*” >显示错误消息.

错误消息定制:

当一个属性校验失败后，校验框架会为该属性生成 4 个消息代码，这些代以校验注解类名为前缀，结合modleAttribute、属性名及属性类型名生成多个对应的消息代码例如 User 类中的 password 属性标准了一个 @Pattern 注解，当该属性值不满足 @Pattern 所定义规则时, 就会产生以下 4个错误代码：

– Pattern.user.password

– Pattern.password

– Pattern.java.lang.String

– Pattern

• 当使用 Spring MVC 标签显示错误消息时， Spring MVC 会查看WEB 上下文是否装配了对应的国化消息，如果没有，则显示默认的错误消息，否则使用国际化消息。

如何使用国际化定制消息:

1.使用国际化文件配置定制信息

2.在SpirngMVC中配置国际化信息文件

若数据类型转换或数据格式转换时发生错误，或该有的参数不存在，或调用处理方法时发生错误，都在隐含模型中创建错误消息。 其错误代码前缀说明如下：

– required：必要的参数不存在。如 @RequiredParam(“param1”)标注了一个入参，但是该参数存在

– typeMismatch：在数据绑定时，发生数据类型不匹配的问题

– methodInvocation：Spring MVC 在调用处理方法时发生了错误