[Bài đọc] Formatter

Formatter cũng hoạt động giống như converter, tức là chuyển đổi một kiểu dữ liệu sang kiểu dữ liệu khác. Tuy nhiên, kiểu dữ liệu nguồn của Formatter là String, trong khi đó converter có thể làm việc với bất cứ kiểu dữ liệu nguồn nào. Formatter phù hợp hơn với tầng web (web-tier), còn converter thì có thể sử dụng ở bất cứ tầng nào. Trong trường hợp muốn chuyển đổi dữ liệu nhập vào từ một trường input trong form trong một ứng dụng Spring MVC, chúng ta nên lựa chọn formatter hơn là converter.

Để tạo một formatter, chúng ta viết một lớp Java triển khai interface org.springframework.format.Formatter. Khai báo của interface này như sau:

**public interface** Formatter<T>

Trong đó, T đại diện cho kiểu dữ liệu đích. Interface này có hai phương thức mà các lớp cần triển khai, đó là parse() và print().

T parse(String text, java.util.Locale locale)  
String print(T object, java.util.Locale locale)

Phương thức*parse()* chuyển một String sang kiểu dữ liệu đích sử dụng một Locale xác định. Phương thức *print()* thì thực hiện công việc ngược lại, chuyển đổi từ một đối tượng của kiểu dữ liệu đích sang String.

Ví dụ sau đây sử dụng LocalDateFormatter để chuyển đổi String sang LocalDate, cũng thực hiện một nhiệm vụ giống như StringToLocalDateConverter ở phần trước.

Lớp The LocalDateFormatter:

**package** formatter;  
  
**import** java.text.ParseException;  
**import** java.time.LocalDate;  
**import** java.time.format.DateTimeFormatter;  
**import** java.time.format.DateTimeParseException;  
**import** java.util.Locale;  
**import** org.springframework.format.Formatter;  
  
**public class** LocalDateFormatter **implements** Formatter<LocalDate> {  
  
 **private** DateTimeFormatter **formatter**;  
 **private** String **datePattern**;  
  
 **public** LocalDateFormatter(String datePattern) {  
 **this**.**datePattern** = datePattern;  
 **formatter** = DateTimeFormatter.ofPattern(datePattern);  
 }  
  
 @Override  
 **public** String print(LocalDate date, Locale locale) {  
 **return** date.format(**formatter**);  
 }  
  
 @Override  
 **public** LocalDate parse(String s, Locale locale) **throws** ParseException {  
 **try** {  
 **return** LocalDate.parse(s,  
 DateTimeFormatter.ofPattern(**datePattern**));  
 } **catch** (DateTimeParseException e) {  
 *// the error message will be displayed in <form:errors>* **throw new** IllegalArgumentException(**"invalid date format. Please use this pattern\""** + **datePattern** + **"\""**);  
 }  
 }  
}

Để sử dụng một formatter trong một ứng dụng Spring MVC, chúng ta cần đăng ký nó sử dụng bean conversionService. Tên lớp của bean phải là org.springframework.format.support.FormattingConversionServiceFactoryBean. Đây là một lớp khác so với lớp mà chúng ta đã sử dụng cho phần converter. Bean này có thể có các thuộc tính converters và formatters để đăng ký các converter và formatter. Đây là một ví dụ:

File cấu hình:

<**beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:p="http://www.springframework.org/schema/p"  
 xmlns:mvc="http://www.springframework.org/schema/mvc"  
 xmlns:context="http://www.springframework.org/schema/context"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd  
 http://www.springframework.org/schema/mvc  
 http://www.springframework.org/schema/mvc/spring-mvc.xsd   
 http://www.springframework.org/schema/context  
 http://www.springframework.org/schema/context/spring-context.xsd"**>  
 <**context:component-scan base-package="controller"**/>  
 <**context:component-scan base-package="formatter"**/>  
 <**mvc:annotation-driven conversion-service="conversionService"**/>  
 <**mvc:resources mapping="/css/\*\*" location="/css/"**/>  
 <**mvc:resources mapping="/\*.html" location="/"**/>  
  
 <**bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>  
 <**property name="prefix" value="/WEB-INF/jsp/"** />  
 <**property name="suffix" value=".jsp"** />  
 </**bean**>  
  
 <**bean id="conversionService" class="org.springframework.format.support.FormattingConversionServiceFactoryBean"**>  
 <**property name="formatters"**>  
 <**set**>  
 <**bean class="formatter.LocalDateFormatter"**>  
 <**constructor-arg type="java.lang.String" value="MM-dd-yyyy"** />  
 </**bean**>  
 </**set**>  
 </**property**>  
 </**bean**>  
</**beans**>

Để ý là chúng ta cần thêm một thành phần component-scan cho formatter.

Để thử nghiệm formatter này, hãy đi đến đường dẫn:

http://localhost:8080/formatter-demo/add-employee

Sử dụng Registrar để đăng ký formatter

Chúng ta có thể sử dụng registrar để đănng ký một formatter. Chẳng hạn, đoạn mã sau sử dụng registrar để đăng ký formatter LocalDateFormatter.

Lớp MyFormatterRegistrar

**package** formatter;  
  
**import** org.springframework.format.FormatterRegistrar;  
**import** org.springframework.format.FormatterRegistry;  
  
**public class** MyFormatterRegistrar **implements** FormatterRegistrar {  
  
 **private** String **datePattern**;  
 **public** MyFormatterRegistrar(String datePattern) {  
 **this**.**datePattern** = datePattern;  
 }  
  
 @Override  
 **public void** registerFormatters(FormatterRegistry registry) {  
 registry.addFormatter(**new** LocalDateFormatter(**datePattern**));  
 *// register more formatters here* }  
}

Khi sử dụng registrar, chúng ta không cần phải đăng ký các formatter ở trong file cấu hình của Spring MVC. Thay vào đó, chúng ta đăng ký trong file cấu hình của registrar.

Ví dụ, đăng ký một registrar trong file springmvc-config.xml

*<?***xml version="1.0" encoding="UTF-8"***?>*<**beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:p="http://www.springframework.org/schema/p"  
 xmlns:mvc="http://www.springframework.org/schema/mvc"  
 xmlns:context="http://www.springframework.org/schema/context"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd  
 http://www.springframework.org/schema/mvc  
 http://www.springframework.org/schema/mvc/spring-mvc.xsd   
 http://www.springframework.org/schema/context  
 http://www.springframework.org/schema/context/spring-context.xsd"**>  
  
 <**context:component-scan base-package="controller"** />  
 <**context:component-scan base-package="service"** />  
 <**mvc:annotation-driven conversion-service="conversionService"** />  
  
 <**mvc:resources mapping="/css/\*\*" location="/css/"** />  
 <**mvc:resources mapping="/\*.html" location="/"** />  
  
 <**bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>  
 <**property name="prefix" value="/WEB-INF/jsp/"** />  
 <**property name="suffix" value=".jsp"** />  
 </**bean**>  
  
 <**bean id="conversionService" class="org.springframework.format.support.FormattingConversionServiceFactoryBean"**>  
 <**property name="formatterRegistrars"**>  
 <**set**>  
 <**bean class="formatter.MyFormatterRegistrar"**>  
 <**constructor-arg type="java.lang.String" value="MM-dd-yyyy"** />  
 </**bean**>  
 </**set**>  
 </**property**>  
 </**bean**>  
</**beans**>

Lựa chọn giữa Converter và Formatter

Một converter được sử dụng chung cho nhiều trường hợp, chuyển đổi bất cứ kiểu dữ liệu nào sang một kiểu dữ liệu khác, chẳng hạn như String sang LocalDate hoặc Long sang LocalDate. Converter có thể sử dụng ở bất cứ tầng nào của ứng dụng.

Trong khi đó, một formatter chỉ có thể chuyển đổi String sang một kiểu dữ liệu khác, chẳng hạn như String sang LocalDate. Không thể sử dụng formatter để chuyển từ Long sang LocalDate. Formatter phù hợp để sử dụng ở tầng web, do đó, trong một ứng dụng Spring MVC, chúng ta nên sử dụng formatter hơn là converter.

Last modified: Wednesday, 6 March 2019, 8:22 PM