30 Spring Core格式化Formatter实现示例及背后原理探究

更新时间：2020-08-06 13:41:43

> 机会不会上门来找人，只有人去找机会。——狄更斯

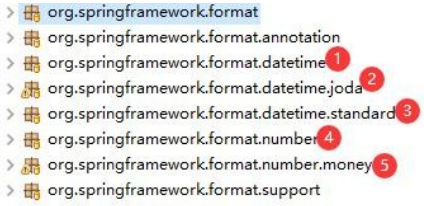
背景

有些数据需要格式化，比如说金额、日期等。传递的日期格式为 `yyyy-MM-dd` 或者 `yyyy-MM-dd hh:ss:mm`，这些是需要格式化的，对于金额也是如此， 比如，1 万元人民币，在正式场合往往要定写作￥10000.00，这些都要求把字符串按照一定的格式转换为日期或者金额。

为了对这些场景做出支持，Spring Context 提供了相关的 Formatter。这样就能满足我们对格式化数据的需求 了，它的内部实际是委托给 Converter 机制去实现的，我们需要定义的场合并不多，所以这里以学习用法为主。日期格式化器在 Spring MVC 中是由系统在启动时完成初始化的，所以并不需要进行干预，同时还提供注解 @DateTimeFormat 来进行日期格式的定义， 而采用注解 @NumberFormat 来进行数字的格式转换。这两个注解都有 pattern 属性，可以定义其格式，例如：`@DateTimeFormat(pattern="yyyy-MM-dd")` 和 `@NumberFormat(pattern="###,###.###")` 。

深入原理

我们先从源码包来看：



Spring 提供了丰富的 Formatter 实现，这些实现分布在如下包中：

org.springframework.format.**datetime**：提供了 **java.util.Date** 类型转换的 formatter；

org.springframework.format.**datetime.joda**：提供了 **Joda 日期和时间**类型转换 formatter；

org.springframework.format.**datetime.standard**：提供 **JSR-310 Java 8 java.time** 类型转换的 formatter；

org.springframework.format.**number**：提供了 **java.lang.Number 子类型**的类型转换 formatters；

org.springframework.format.**number.money**：提供了 **JSR 354 javax.money** 的类型转换 formatters。

实例验证

DefaultFormattingConversionService 实现了 FormattingConversionService，它内部包含了 Formatter 的各种默认实现。

```java
package com.davidwang456.test;

import org.springframework.core.convert.ConversionService;
import org.springframework.format.support.DefaultFormattingConversionService;

public class DefaultFormatterListExample {

    public static void main(String[] args) {
            ConversionService service =new DefaultFormattingConversionService();
            System.out.println(service);
    }
}
```

我们通过程序来打印出 DefaultFormattingConversionService 的默认 Formatter 实现，输出结果为：

```
ConversionService converters =
        java.lang.Boolean -> java.lang.String : org.springframework.core.convert.support.ObjectToStringConverter@1ddc4ec2
        java.lang.Character -> java.lang.Number : org.springframework.core.convert.support.CharacterToNumberFactory@9807454
        java.lang.Character -> java.lang.String : org.springframework.core.convert.support.ObjectToStringConverter@506e1b77
        java.lang.Enum -> java.lang.Integer : org.springframework.core.convert.support.EnumToIntegerConverter@30dae81
        java.lang.Enum -> java.lang.String : org.springframework.core.convert.support.EnumToStringConverter@b1bc7ed
        java.lang.Integer -> java.lang.Enum : org.springframework.core.convert.support.IntegerToEnumConverterFactory@7cd84586
        java.lang.Number -> java.lang.Character : org.springframework.core.convert.support.NumberToCharacterConverter@4fca772d
        java.lang.Number -> java.lang.Number : org.springframework.core.convert.support.NumberToNumberConverterFactory@312b1dae
        java.lang.Number -> java.lang.String : org.springframework.core.convert.support.ObjectToStringConverter@27bc2616
        java.lang.String -> java.lang.Boolean : org.springframework.core.convert.support.StringToBooleanConverter@3d494fbf
        java.lang.String -> java.lang.Character : org.springframework.core.convert.support.StringToCharacterConverter@3941a79c
        java.lang.String -> java.lang.Enum : org.springframework.core.convert.support.StringToEnumConverterFactory@133314b
        java.lang.String -> java.lang.Number : org.springframework.core.convert.support.StringToNumberConverterFactory@7530d0a
        java.lang.String -> java.nio.charset.Charset : org.springframework.core.convert.support.StringToCharsetConverter@70177ecd
        java.lang.String -> java.util.Currency : org.springframework.core.convert.support.StringToCurrencyConverter@66a29884
        java.lang.String -> java.util.Locale : org.springframework.core.convert.support.StringToLocaleConverter@1b2c6ec2
        java.lang.String -> java.util.Properties : org.springframework.core.convert.support.StringToPropertiesConverter@cc34f4d
        java.lang.String -> java.util.TimeZone : org.springframework.core.convert.support.StringToTimeZoneConverter@300ffa5d
        java.lang.String -> java.util.UUID : org.springframework.core.convert.support.StringToUUIDConverter@65b3120a
        java.nio.charset.Charset -> java.lang.String : org.springframework.core.convert.support.ObjectToStringConverter@1e80bfe8
        java.time.ZoneId -> java.util.TimeZone : org.springframework.core.convert.support.ZoneIdToTimeZoneConverter@1f17ae12
        java.time.ZonedDateTime -> java.util.Calendar : org.springframework.core.convert.support.ZonedDateTimeToCalendarConverter@4d405ef7
        java.util.Currency -> java.lang.String : org.springframework.core.convert.support.ObjectToStringConverter@4769b07b
        java.util.Locale -> java.lang.String : org.springframework.core.convert.support.ObjectToStringConverter@4edde6e5
        java.util.Properties -> java.lang.String : org.springframework.core.convert.support.PropertiesToStringConverter@17a7cec2
        java.util.UUID -> java.lang.String : org.springframework.core.convert.support.ObjectToStringConverter@6f539caf
        org.springframework.core.convert.support.ArrayToArrayConverter@2dda6444
        org.springframework.core.convert.support.ArrayToCollectionConverter@79fc0f2f
        org.springframework.core.convert.support.ArrayToObjectConverter@6e2c634b
        org.springframework.core.convert.support.ArrayToStringConverter@378fd1ac
        org.springframework.core.convert.support.ByteBufferConverter@76fb509a
        org.springframework.core.convert.support.ByteBufferConverter@76fb509a
        org.springframework.core.convert.support.ByteBufferConverter@76fb509a
        org.springframework.core.convert.support.ByteBufferConverter@76fb509a
        org.springframework.core.convert.support.CollectionToArrayConverter@50040f0c
        org.springframework.core.convert.support.CollectionToCollectionConverter@5e9f23b4
        org.springframework.core.convert.support.CollectionToObjectConverter@7106e68e
        org.springframework.core.convert.support.CollectionToStringConverter@7e6cbb7a
        org.springframework.core.convert.support.FallbackObjectToStringConverter@c4437c4
        org.springframework.core.convert.support.IdToEntityConverter@6193b845,org.springframework.core.convert.support.ObjectToObjectConverter@2e817b38
        org.springframework.core.convert.support.MapToMapConverter@4783da3f
        org.springframework.core.convert.support.ObjectToArrayConverter@37a71e93
        org.springframework.core.convert.support.ObjectToCollectionConverter@7eda2dbb
        org.springframework.core.convert.support.ObjectToOptionalConverter@433c675d
        org.springframework.core.convert.support.ObjectToOptionalConverter@433c675d
        org.springframework.core.convert.support.ObjectToOptionalConverter@433c675d
        org.springframework.core.convert.support.StreamConverter@6576fe71
        org.springframework.core.convert.support.StreamConverter@6576fe71
        org.springframework.core.convert.support.StreamConverter@6576fe71
        org.springframework.core.convert.support.StreamConverter@6576fe71
        org.springframework.core.convert.support.StringToArrayConverter@49097b5d
        org.springframework.core.convert.support.StringToCollectionConverter@7c3df479
```

既然 DefaultFormattingConversionService 包含了如此多的 Formatter，我们就来练练手，使用一下吧：

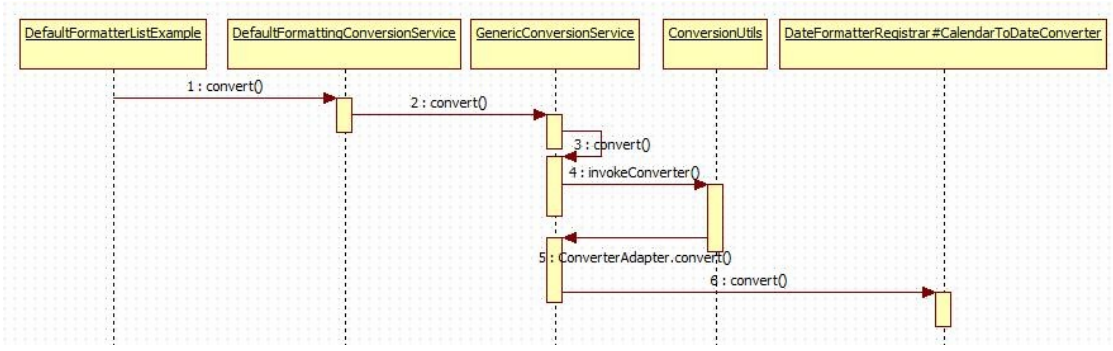
```
package com.davidwang456.test;

import java.util.Calendar;
import java.util.Date;
import java.util.Locale;

import org.springframework.core.convert.ConversionService;
import org.springframework.format.Formatter;
import org.springframework.format.datetime.DateFormatter;
import org.springframework.format.number.NumberStyleFormatter;
import org.springframework.format.support.DefaultFormattingConversionService;

public class DefaultFormatterListExample {

    public static void main(String[] args) {
        //date formatter
        //Formatter dtform = new DateFormatter();
        //System.out.println(dtform.print(new Date(), Locale.CHINESE));

        //System.out.println(dtform.print(new Date(), Locale.JAPANESE));

        //number formatter
        //Formatter numform = new NumberStyleFormatter();
        //System.out.println(numform.print(10, Locale.CHINESE));
        //System.out.println(numform.print(10.5, Locale.JAPANESE));

           ConversionService service =
                   new DefaultFormattingConversionService();
            System.out.println(service);
          Date date= service.convert(Calendar.getInstance(), Date.class);
          System.out.println(date);
    }
}
```

第一步：打印出 DefaultFormattingConversionService 包含的默认 Formatter 实现；

第二步：利用 DefaultFormattingConversionService 将 Calendar 转换为 Date 类型。

深入 Formatter 原理

debug 进去，发现代码执行如下：



重点 GenericConversionService#getConverter，converterCache 是一个 ConcurrentReferenceHashMap 结构，缓存了众多的 Converter，getConverter 根据输入数据类型和输出数据类型确定合适的 Converter。

```java
/**
 * Hook method to lookup the converter for a given sourceType/targetType pair.
 * First queries this ConversionService's converter cache.
 * On a cache miss, then performs an exhaustive search for a matching converter.
 * If no converter matches, returns the default converter.
 * @param sourceType the source type to convert from
 * @param targetType the target type to convert to
 * @return the generic converter that will perform the conversion,
 * or {@code null} if no suitable converter was found
 * @see #getDefaultConverter(TypeDescriptor, TypeDescriptor)
 */
@Nullable
protected GenericConverter getConverter(TypeDescriptor sourceType, TypeDescriptor targetType) {
    ConverterCacheKey key = new ConverterCacheKey(sourceType, targetType);
    GenericConverter converter = this.converterCache.get(key);
    if (converter != null) {
        return (converter != NO_MATCH ? converter : null);
    }

    converter = this.converters.find(sourceType, targetType);
    if (converter == null) {
        converter = getDefaultConverter(sourceType, targetType);
    }

    if (converter != null) {
        this.converterCache.put(key, converter);
        return converter;
    }

    this.converterCache.put(key, NO_MATCH);
    return null;
}
```

如何自定义格式转换的 Formatter 呢？只要实现 Formatter 接口即可。Formatter 主要是两个方法，通过 print 方法能将结果按照一定的格式输出字符串。通过 parse 方法能够将满足一定格式的字符串转换为对象。

```java
package com.davidwang456.test;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

import org.springframework.format.Formatter;

public final class DateFormatter11 implements Formatter<Date> {

    private String pattern;

    public DateFormatter11(String pattern) {
        this.pattern = pattern;
    }

    public String print(Date date, Locale locale) {
        if (date == null) {
            return "";
        }
        return getDateFormat(locale).format(date);
    }

    public Date parse(String formatted, Locale locale) throws ParseException {
        if (formatted.length() == 0) {
            return null;
        }
        return getDateFormat(locale).parse(formatted);
    }

    protected DateFormat getDateFormat(Locale locale) {
        DateFormat dateFormat = new SimpleDateFormat(this.pattern, locale);
        dateFormat.setLenient(false);
        return dateFormat;
    }

}
```

然后注册到相应的 ConversionService 中，以 DefaultFormattingConversionService，可以修改源码如下：

```java
/**
 * Add formatters appropriate for most environments: including number formatters,
 * JSR-354 Money & Currency formatters, JSR-310 Date-Time and/or Joda-Time formatters,
 * depending on the presence of the corresponding API on the classpath.
 * @param formatterRegistry the service to register default formatters with
 */
public static void addDefaultFormatters(FormatterRegistry formatterRegistry) {
    // Default handling of number values
    formatterRegistry.addFormatterForFieldAnnotation(new NumberFormatAnnotationFormatterFactory());
    formatterRegistry.addFormatter(new DateFormatter11("yyyy-mm-dd"));
    // Default handling of monetary values
    if (jsr354Present) {
        formatterRegistry.addFormatter(new CurrencyUnitFormatter());
        formatterRegistry.addFormatter(new MonetaryAmountFormatter());
        formatterRegistry.addFormatterForFieldAnnotation(new Jsr354NumberFormatAnnotationFormatterFactory());
    }

    // Default handling of date-time values

    // just handling JSR-310 specific date and time types
    new DateTimeFormatterRegistrar().registerFormatters(formatterRegistry);

    if (jodaTimePresent) {
        // handles Joda-specific types as well as Date, Calendar, Long
        new JodaTimeFormatterRegistrar().registerFormatters(formatterRegistry);
    }
    else {
        // regular DateFormat-based Date, Calendar, Long converters
```

```
        new DateFormatterRegistrar().registerFormatters(formatterRegistry);
    }
}
```

总结

Converter 和 Formatter 看上去差别不大，但其实作用是不同的，Converter 更强调从一种类型转换为另外一种类型，Formatter 更注重不同的展现形式，如传递的日期格式为 `yyyy-MM-dd` 或者 `yyyy-MM-dd hh:ss:mm`， 这些是需要格式化的。

}

29 Spring MVC之类型转换 Converter

31 Converter还是Formatter?