

# 第十五章：Spring 类型转换

小马哥 · mercyblitz



扫码试看/订阅

《小马哥讲 Spring 核心编程思想》视频课程

# Spring 类型转换

---

1. Spring 类型转换的实现
2. 使用场景
3. 基于 JavaBeans 接口的类型转换
4. Spring 内建 PropertyEditor 扩展
5. 自定义 PropertyEditor 扩展
6. Spring PropertyEditor 的设计缺陷
7. Spring 3.0 通用类型转换接口
8. Spring 内建类型转换器
9. Converter 接口的局限性
10. GenericConverter 接口



# Spring 类型转换

---

- 11. 优化 GenericConverter 接口
- 12. 扩展 Spring 类型转换器
- 13. 统一类型转换服务
- 14. ConversionService 作为依赖
- 15. 面试题精选



# Spring 类型转换的实现

- 基于 JavaBeans 接口的类型转换实现 [spring 3.0 之前](#)
  - 基于 `java.beans.PropertyEditor` 接口扩展
- Spring 3.0+ 通用类型转换实现

## 使用场景

- 场景分析

场景	基于 JavaBeans 接口的类型转换实现	Spring 3.0+ 通用类型转换实现
数据绑定	YES	YES
BeanWrapper	YES	YES
Bean 属性类型装换	YES	YES
外部化属性类型转换	NO	YES

# 基于 JavaBeans 接口的类型转换

- 核心职责
  - 将 String 类型的内容转化为目标类型的对象  
局限性:只适用 string
- 扩展原理
  - Spring 框架将文本内容传递到 PropertyEditor 实现的 setAsText(String) 方法
  - PropertyEditor#setAsText(String) 方法实现将 String 类型转化为目标类型的对象
  - 将目标类型的对象传入 PropertyEditor#setValue(Object) 方法
  - PropertyEditor#setValue(Object) 方法实现需要临时存储传入对象
  - Spring 框架将通过 PropertyEditor#getValue() 获取类型转换后的对象

## Spring 內建 PropertyEditor 扩展

- 內建扩展（org.springframework.beans.propertyeditors 包下）

转换场景	实现类
String -> Byte 数组	org.springframework.beans.propertyeditors.ByteArrayPropertyEditor
String -> Char	org.springframework.beans.propertyeditors.CharacterEditor
String -> Char 数组	org.springframework.beans.propertyeditors.CharArrayPropertyEditor
String -> Charset	org.springframework.beans.propertyeditors.CharsetEditor
String -> Class	org.springframework.beans.propertyeditors.ClassEditor
String -> Currency	org.springframework.beans.propertyeditors.CurrencyEditor
...	...



# 自定义 PropertyEditor 扩展

- 扩展模式
  - 扩展 `java.beans.PropertyEditorSupport` 类
- 实现 `org.springframework.beans.PropertyEditorRegistrar`
  - 实现 `registerCustomEditors(org.springframework.beans.PropertyEditorRegistry)` 方法
  - 将 `PropertyEditorRegistrar` 实现注册为 Spring Bean
- 向 `org.springframework.beans.PropertyEditorRegistry` 注册自定义 `PropertyEditor` 实现
  - 通用类型实现 `registerCustomEditor(Class<?>, PropertyEditor)`
  - Java Bean 属性类型实现：`registerCustomEditor(Class<?>, String, PropertyEditor)`  
`pojo`    属性路径

# Spring PropertyEditor 的设计缺陷

- 违反职责单一原则
  - java.beans.PropertyEditor 接口职责太多，除了类型转换，还包括 Java Beans 事件和 Java GUI 交互
- java.beans.PropertyEditor 实现类型局限
  - 来源类型只能为 java.lang.String 类型
- java.beans.PropertyEditor 实现缺少类型安全
  - 除了实现类命名可以表达语义，实现类无法感知目标转换类型  
不是强类型的，依赖类名表达含义(如StringToPropertiesPropertyEditor)

## Spring 3 通用类型转换接口

- 类型转换接口 - `org.springframework.core.convert.converter.Converter<S,T>`
  - 泛型参数 S：来源类型，参数 T：目标类型
  - 核心方法：T convert(S)
- 通用类型转换接口 - `org.springframework.core.convert.converter.GenericConverter`
  - 核心方法：convert(Object,TypeDescriptor,TypeDescriptor)
  - 配对类型：org.springframework.core.convert.converter.GenericConverter.ConvertiblePair
  - 类型描述：org.springframework.core.convert.TypeDescriptor

# Spring 內建类型转换器

- 內建扩展

转换场景	实现类所在包名 (package)
日期/时间相关	org.springframework.format.datetime
Java 8 日期/时间相关	org.springframework.format.datetime.standard
通用实现	org.springframework.core.convert.support

## Converter 接口的局限性

- 局限一：缺少 Source Type 和 Target Type 前置判断 S→T有可能无法转换
  - 应对：增加 `org.springframework.core.convert.converter.ConditionalConverter` 实现
- 局限二：仅能转换单一的 Source Type 和 Target Type
  - 应对：使用 `org.springframework.core.convert.converter.GenericConverter` 代替支持复合类型

## GenericConverter 接口

- `org.springframework.core.convert.converter.GenericConverter`

核心要素	说明
使用场景	用于“复合”类型转换场景，比如 Collection、Map、数组等
转换范围	<code>Set&lt;ConvertiblePair&gt; getConvertibleTypes()</code>
配对类型	<code>org.springframework.core.convert.converter.GenericConverter.ConvertiblePair</code>
转换方法	<code>convert(Object, TypeDescriptor, TypeDescriptor)</code>
类型描述	<code>org.springframework.core.convert.TypeDescriptor</code>

GenericConverter 可以和 Converter 配合使用 (如 CollectionToArrayConverter)

# 优化 GenericConverter 接口

- GenericConverter 局限性
  - 缺少 Source Type 和 Target Type 前置判断
  - 单一类型转换实现复杂
- GenericConverter 优化接口 - ConditionalGenericConverter
  - 复合类型转换: `org.springframework.core.convert.converter.GenericConverter`
  - 类型条件判断: `org.springframework.core.convert.converter.ConditionalConverter`

## 扩展 Spring 类型转换器

- 实现转换器接口
  - `org.springframework.core.convert.converter.Converter`
  - `org.springframework.core.convert.converter.ConverterFactory`
  - `org.springframework.core.convert.converter.GenericConverter`
- 注册转换器实现
  - 通过 `ConversionServiceFactoryBean` Spring Bean
  - 通过 `org.springframework.core.convert.ConversionService` API



## 统一类型转换服务

- `org.springframework.core.convert.ConversionService`

实现类型	说明
<code>GenericConversionService</code>	通用 <code>ConversionService</code> 模板实现，不内置转化器实现
<code>DefaultConversionService</code>	基础 <code>ConversionService</code> 实现，内置常用转化器实现
<code>FormattingConversionService</code>	通用 <code>Formatter</code> + <code>GenericConversionService</code> 实现，不内置转化器和 <code>Formatter</code> 实现
<code>DefaultFormattingConversionService</code>	<code>DefaultConversionService</code> + 格式化 实现（如：JSR-354 Money & Currency, JSR-310 Date-Time）

## ConversionService 作为依赖

- 类型转换器底层接口 - `org.springframework.beans.TypeConverter`
  - 起始版本: Spring 2.0
  - 核心方法 - `convertIfNecessary` 重载方法
  - 抽象实现 - `org.springframework.beans.TypeConverterSupport`
  - 简单实现 - `org.springframework.beans.SimpleTypeConverter`

## ConversionService 作为依赖

- 类型转换器底层抽象实现 - `org.springframework.beans.TypeConverterSupport`
  - 实现接口 - `org.springframework.beans.TypeConverter`
  - 扩展实现 - `org.springframework.beans.PropertyEditorRegistrySupport`
  - 委派实现 - `org.springframework.beans.TypeConverterDelegate`

## ConversionService 作为依赖

- 类型转换器底层委派实现 - `org.springframework.beans.TypeConverterDelegate`
  - 构造来源 - `org.springframework.beans.AbstractNestablePropertyAccessor` 实现
    - `org.springframework.beans.BeanWrapperImpl`
  - 依赖 - `java.beans.PropertyEditor` 实现
    - 默认内建实现 - `PropertyEditorRegistrySupport#registerDefaultEditors`
  - 可选依赖 - `org.springframework.core.convert.ConversionService` 实现

## 面试题

**沙雕面试题** - Spring 类型转换实现有哪些?

答:

1. 基于 JavaBeans PropertyEditor 接口实现
2. Spring 3.0+ 通用类型转换实现



我真的没笑

## 面试题



### 996 面试题 - Spring 类型转换器接口有哪些？

答：

- 类型转换接口 - `org.springframework.core.convert.converter.Converter`
- 通用类型转换接口 - `org.springframework.core.convert.converter.GenericConverter`
- 类型条件接口 - `org.springframework.core.convert.converter.ConditionalConverter`
- 综合类型转换接口 -  
`org.springframework.core.convert.converter.ConditionalGenericConverter`

# 面试题

**劝退面试题** - TypeDescriptor 是如何处理泛型？

答：答案下章揭晓





扫码试看/订阅

《小马哥讲 Spring 核心编程思想》视频课程