



当前位置: Java 技术驿站 (<http://cmsblogs.com>) > 死磕Java (<http://cmsblogs.com/?cat=189>) > 死磕 Spring (<http://cmsblogs.com/?cat=206>) > 正文

【死磕 Spring】—— IOC 之自定义类型转换器 (<http://cmsblogs.com/?p=3985>)

2019-01-20 分类: 死磕 Spring (<http://cmsblogs.com/?cat=206>) 阅读(6082) 评论(0)

原文出自: <http://cmsblogs.com> (<http://cmsblogs.com>)

在上篇文章中小编分析了 Spring ConversionService 类型转换体系, 相信各位都对其有了一个清晰的认识, 这篇博客将利用 ConversionService 体系来实现自己的类型转换器。

ConversionService 是 Spring 类型转换器体系中的核心接口, 它定义了是否可以完成转换 (`canConvert()`) 与 类型转换 (`convert()`) 两类接口。ConversionService 有三个子类, 每个子类针对不同的类型转换:

- Converter<S,T>:将 S 类型对象转为 T 类型对象。
- 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。
- 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换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。




如何自定义类型转换器? 分两步走:

1. 实现 Converter / GenericConverter / ConverterFactory 接口
2. 将该类注册到 ConversionServiceFactoryBean 中。

ConversionServiceFactoryBean 实现了 InitializingBean 接口实现 `afterPropertiesSet()`, 我们知道在 Bean 实例化 bean 阶段, Spring 容器会检查当前 bean 是否实现了 InitializingBean 接口, 如果是则执行相应的初始化方法。(关于 InitializingBean 详情请参考:【死磕 Spring】----- IOC 之 深入分析 InitializingBean 和 `init-method()`)。 `afterPropertiesSet()` 源码如下:

```
public void afterPropertiesSet() {  
    this.conversionService = createConversionService();  
    ConversionServiceFactory.registerConverters(this.converters, this.conversionService);  
}
```

首先调用 `createConversionService()` 初始化 `conversionService`, 然后调用 `ConversionServiceFactory.registerConverters()` 将定义的 `converters` 注入到类型转换体系中。`createConversionService()` 其实就是创建一个 `DefaultConversionService` 实例对象, 对于 `DefaultConversionService` 小编在上篇博客已经分析了, 如有不了解的请移步上篇博文。这里直接分析 `ConversionServiceFactory.registerConverters()`, 该方法是将定义的 `converter` 注册到目标 `ConverterRegistry` 中, 我们知道 `ConverterRegistry` 是一个 `Converter` 注册器, 他定义了一系列注册方法。

```

public static void registerConverters(@Nullable Set<?> converters, ConverterRegistry registry) {
    if (converters != null) {
        for (Object converter : converters) {
            if (converter instanceof GenericConverter) {
                registry.addConverter((GenericConverter) converter);
            }
            else if (converter instanceof Converter<?, ?>) {
                registry.addConverter((Converter<?, ?>) converter);
            }
            else if (converter instanceof ConverterFactory<?, ?>) {
                registry.addConverterFactory((ConverterFactory<?, ?>) converter);
            }
            else {
                throw new IllegalArgumentException("Each converter object must implement one of the " +
                    "Converter, ConverterFactory, or GenericConverter interfaces");
            }
        }
    }
}

```

调用 ConverterRegistry 的 addConverter() 方法将转换器注册到容器中。所以在我们使用 Spring 容器的时候，Spring 将会自动识别出 IOC 容器中注册的 ConversionService 并且在 bean 属性注入阶段使用自定义的转换器完成属性的转换了。

实例

定义 StudentConversionService 转换器：

```

public class StudentConversionService implements Converter<String,StudentService>{

    @Override
    public StudentService convert(String source) {
        if(StringUtils.hasLength(source)){
            String[] sources = source.split("#");

            StudentService studentService = new StudentService();
            studentService.setAge(Integer.parseInt(sources[0]));
            studentService.setName(sources[1]);

            return studentService;
        }
        return null;
    }
}


```

配置：

```
<bean id="conversionService"
      class="org.springframework.context.support.ConversionServiceFactoryBean">
  <property name="converters">
    <set>
      <ref bean="studentConversionService"/>
    </set>
  </property>
</bean>

<bean id="studentConversionService" class="org.springframework.core.conversion.StudentConversionService"/>

<bean id="student" class="org.springframework.core.conversion.Student">
  <property name="studentService" value="18#chenssy"/>
</bean>
```

 赞(6) 打赏

【公告】版权声明 (http://cmsblogs.com/?page_id=1908)

标签: Spring 源码解析 (<http://cmsblogs.com/?tag=spring-%e6%ba%90%e7%a0%81%e8%a7%a3%e6%9e%90>)

死磕Java (<http://cmsblogs.com/?tag=%e6%ad%bb%e7%a3%95java>)

死磕Spring (<http://cmsblogs.com/?tag=%e6%ad%bb%e7%a3%95spring>)

 **chenssy** (<http://cmsblogs.com/?author=1>)

不想当厨师的程序员不是好的架构师....

上一篇

【死磕 Spring】—— IOC 之深入分析 Bean 的类型转换体系 (<http://cmsblogs.com/?p=3983>)

下一篇

深入理解 Java 内存模型 (一) ——基础 (<http://cmsblogs.com/?p=3992>)

- 【死磕 Redis】—— 如何排查 Redis 中的慢查询 (<http://cmsblogs.com/?p=18352>)
- 【死磕 Redis】—— 发布与订阅 (<http://cmsblogs.com/?p=18348>)
- 【死磕 Redis】—— 布隆过滤器 (<http://cmsblogs.com/?p=18346>)
- 【死磕 Redis】—— 理解 pipeline 管道 (<http://cmsblogs.com/?p=18344>)
- 【死磕 Redis】—— 事务 (<http://cmsblogs.com/?p=18340>)
- 【死磕 Redis】—— Redis 的线程模型 (<http://cmsblogs.com/?p=18337>)
- 【死磕 Redis】—— Redis 通信协议 RESP (<http://cmsblogs.com/?p=18334>)
- 【死磕 Redis】—— 开篇 (<http://cmsblogs.com/?p=18332>)
- spring boot 源码解析11-ConfigurationClassPostProcessor类加载解析 (<http://cmsblogs.com/?p=9522>)

- spring boot 源码解析12-servlet容器的建立 (<http://cmsblogs.com/?p=9520>)



- spring boot 源码解析13-@ConfigurationProperties是如何生效的 (<http://cmsblogs.com/?p=9518>)
- spring boot 源码解析15-spring mvc零配置 (<http://cmsblogs.com/?p=9516>)
- spring boot 源码解析16-spring boot外置tomcat部署揭秘 (<http://cmsblogs.com/?p=9514>)
- spring boot 源码解析17-mvc自动化配置揭秘 (<http://cmsblogs.com/?p=9512>)
- spring boot 源码解析18-WebMvcAutoConfiguration自动化配置揭秘 (<http://cmsblogs.com/?p=9510>)

© 2014 - 2021 Java 技术驿站 (<http://cmsblogs.com>) 网站地图 (http://cmsblogs.com/sitemap_baidu.xml) |  https://www.cnzz.com/stat/website.php?web_id=5789174



湘ICP备14000180号-1 (<https://beian.miit.gov.cn/>)

>>> 网站已平稳运行: 2677 天 5 小时 19 分 30 秒