芋道源码 —— 知识星球

我是一段不羁的公告！
记得给芳芳这 3 个项目加油，添加一个 STAR 噢。
https://github.com/YunaiV/SpringBoot-Labs
https://github.com/YunaiV/onemall
https://github.com/YunaiV/ruoyi-vue-pro
2019-09-04
Spring

【死磕 Spring】—— IoC 之自定义类型转换器

本文主要基于 Spring 5.0.6.RELEASE

摘要：原创出处 http://cmsblogs.com/?p=todo 「小明哥」，谢谢！

作为「小明哥」的忠实读者，「老芳芳」略作修改，记录在理解过程中，参考的资料。

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(...)`） 两类接口方法。

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。

ConversionServiceFactoryBean

那么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()` 方法，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()` 方法，代码如下：

```
// ConversionServiceFactoryBean.java

@Override
public void afterPropertiesSet() {
    // 创建 DefaultConversionService 对象
    this.conversionService = createConversionService();
    // 注册到 ConversionServiceFactory 中
    ConversionServiceFactory.registerConverters(this.converters, this.conversionService);
}
```

首先调用 `#createConversionService()` 方法，初始化 `conversionService`。代码如下：

```
// ConversionServiceFactoryBean.java

protected GenericConversionService createConversionService() {
 return new DefaultConversionService();
}
```

然后调用 `ConversionServiceFactory#registerConverters(Set<?> converters, ConverterRegistry registry)` 方法，将定义的 `converters` 注入到类型转换体系中。代码如下：

```
// ConverterRegistry.java

public static void registerConverters(@Nullable Set<?> converters, ConverterRegistry registry) {
 if (converters != null) {
        // 遍历 converters 数组，逐个注解
     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");
            }
        }
    }
}
```

- 我们知道 ConverterRegistry 是一个 Converter 注册器，他定义了一系列注册方法。
- 通过调用 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。

示例

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>
```

运行结果：

脑补一下~哈哈哈哈

文章目录