28 百变星君Converter实现示例及背后原理探究

更新时间：2020-07-24 11:06:37

> 耐心是一切聪明才智的基础。——柏拉图

背景

图片来源网络

《百变星君》是由叶伟民导演，周星驰，梁咏琪，吴孟达等主演的奇幻科幻喜剧片，于 1995 年在香港上映。剧中周星驰科幻般的七十二变给观众留下了深刻的印象。在 Spring 的世界中，也有一位百变星君，它就是 Converter。

Spring 中的百变星君

类型转换的应用场景一般在 Web 应用中，如 Spring MVC 的流程中：

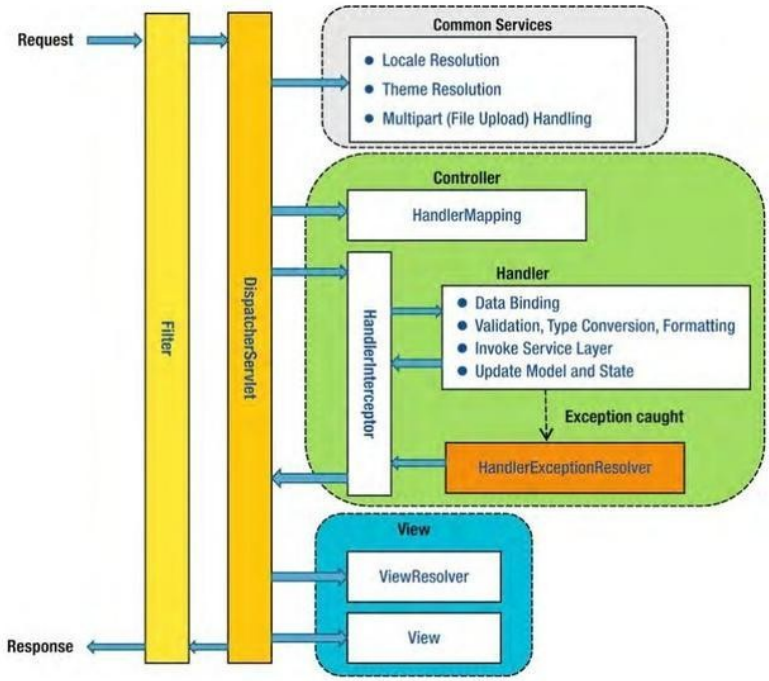
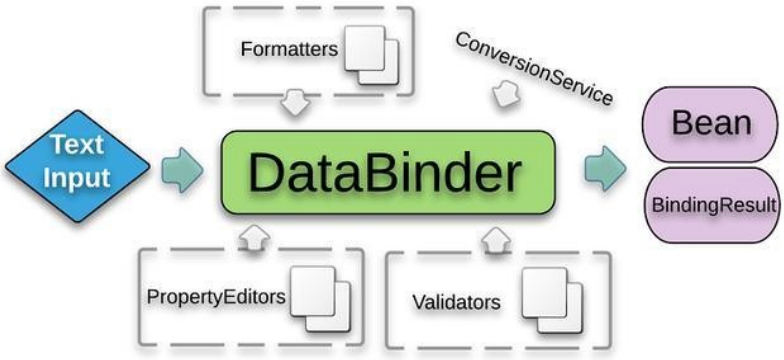


Figure 17-3. Spring MVC request life cycle

在前面的章节我们提到了数据绑定 DataBinder，它包括数据校验，数据格式转换等功能。其中，数据校验 Validator，属性编辑 PropertyEditor 都做了讲解，接下来我们开始看 ConversionService 和 Formatter。本节我们主要讲解 ConversionService，下节讨论 Formatter。



ConversionService 被设计成无状态对象,在容器启动时被实例化,在多线程间进行共享（线程安全）。在 Spring 应用中，可以自定义类型转换器。当需要框架进行类型转换时，Spring 会选择合适的类型转换器使用。你也可以注入 ConversionService 到 beans 或者直接调用。但 ConversionService 是如何实现数据格式转换的呢？

ConversionService 的默认实现 DefaultConversionService 内部封装了各种 Converter 的实现，为什么不直接使用 Converter 呢？我们知道不是线程安全的，而 ConversionService 是线程安全的。与 PropertyEditorSupport 不同的是，converter 可以将任意类型的输入转换为任意类型的输出。

探究原理

我们抛开复杂的应用，仅仅用最简单的例子来完成。

示例程序：

```java
package com.davidwang456.test;

import org.springframework.core.convert.support.DefaultConversionService;

import java.util.Collection;

public class ConversionServiceExample {
    public static void main (String[] args) {

        DefaultConversionService service = new DefaultConversionService();
        System.out.println(service);
        Collection<String> list = service.convert("Deb, Mike, Kim",Collection.class);
        System.out.println(list);
    }
}
```

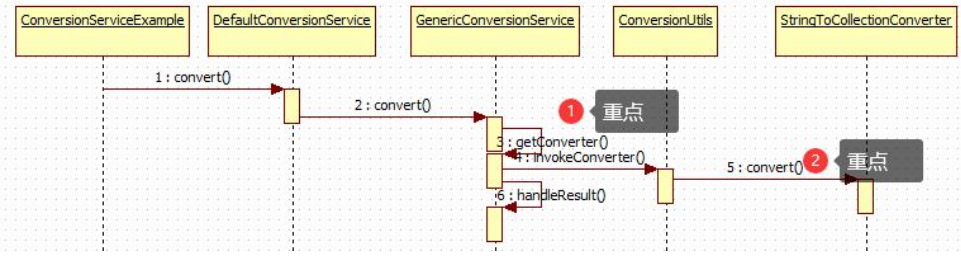
打印出 DefaultConversionService 的内容，它包含了很多类型的 Converter 实现：

> ConversionService converters =
>
> java.lang.Boolean -> java.lang.String : org.springframework.core.convert.support.ObjectToStringConverter@31221be2
>
> java.lang.Character -> java.lang.Number : org.springframework.core.convert.support.CharacterToNumberFactory@2e5d6d97
>
> ……
>
> ……

从上面可以看出 ConversionService 封装了各种 Converter 和 ConvertFactory。

深入原理

我们来 debug 进去看看：



重点 1：GenericConversionService.java#convert 方法

```java
@Override
@Nullable
public Object convert(@Nullable Object source, @Nullable TypeDescriptor sourceType, TypeDescriptor targetType) {
    Assert.notNull(targetType, "Target type to convert to cannot be null");
    if (sourceType == null) {
        Assert.isTrue(source == null, "Source must be [null] if source type == [null]");
        return handleResult(null, targetType, convertNullSource(null, targetType));
    }
    if (source != null && !sourceType.getObjectType().isInstance(source)) {
        throw new IllegalArgumentException("Source to convert from must be an instance of [" +
                sourceType + "]; instead it was a [" + source.getClass().getName() + "]");
    }
    GenericConverter converter = getConverter(sourceType, targetType);
    if (converter != null) {
        Object result = ConversionUtils.invokeConverter(converter, source, sourceType, targetType);
        return handleResult(sourceType, targetType, result);
    }
    return handleConverterNotFound(source, sourceType, targetType);
}
```

该方法根据输入数据的类型和输出数据的类型从注册的 Converter 中查找最符合条件的转换器。

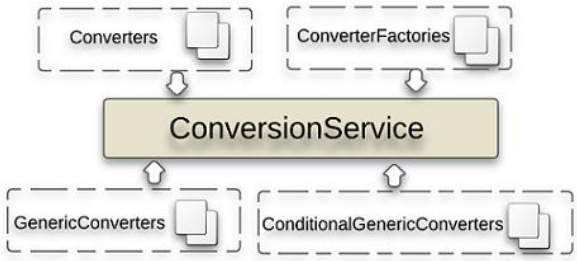
重点 2：StringToCollectionConverter.java#convert 方法

```
@Override
@Nullable
public Object convert(@Nullable Object source, TypeDescriptor sourceType, TypeDescriptor targetType) {
    if (source == null) {
        return null;
    }
    String string = (String) source;

    String[] fields = StringUtils.commaDelimitedListToStringArray(string);
    TypeDescriptor elementDesc = targetType.getElementTypeDescriptor();
    Collection<Object> target = CollectionFactory.createCollection(targetType.getType(),
            (elementDesc != null ? elementDesc.getType() : null), fields.length);

    if (elementDesc == null) {
        for (String field : fields) {
            target.add(field.trim());
        }
    }
    else {
        for (String field : fields) {
            Object targetElement = this.conversionService.convert(field.trim(), sourceType, elementDesc);
            target.add(targetElement);
        }
    }
    return target;
}
```

上面出现的参数类型 TypeDescriptor 是对于一种类型的封装，里面包含该种类型的值、实际类型等等信息。此时实现真正的类型转换，支持单个的数据类型转换，也支持多个数据的类型转换。

总结

Converter 不是线程安全的，ConversionService 被设计成无状态对象，在容器启动时被实例化，在多线程间进行共享（线程安全）。



ConversionService 封装了诸多 Converter、ConverterFactories、GenericConverter 和 ConditionalGnericConverter 的实现类，具有丰富的功能，不愧称之为 Spring 中的百变星君。

}

27 Spring MVC数据绑定 InitBinder揭秘

29 Spring MVC之类型转换 Converter