26 Spring Core 数据绑定之DataBinder实现示例及背后原理探究

更新时间：2020-08-04 18:32:07

> 完成工作的方法，是爱惜每一分钟。——达尔文

背景

在 Spring 配置文件中使用 <value.../> 元素来为这两个属性指定属性值，如下面的例子所示：

```
<bean id="pojoBean" class="com.davidwang456.test.POJOBean">
    <property name="name" value="davidwang456"></property>
    <property name="age" value="10"></property>
    <property name="name" value="2019-12-24"></property>
</bean>
```

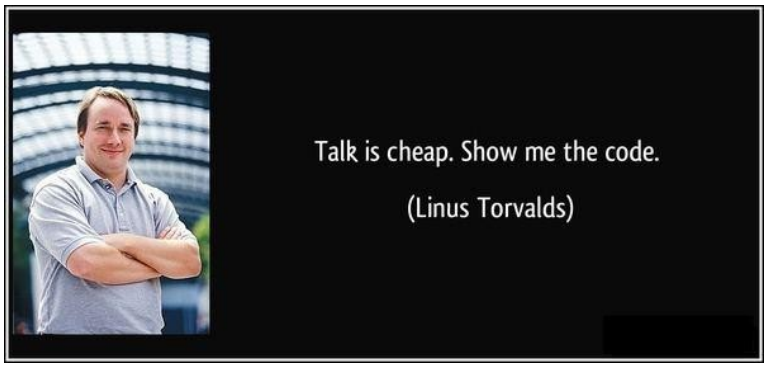
上面的 XML 是如何实现的呢？

探究原理

为了探究上面的原理，我们暂且忘掉上面 XML 配置方式，如何使用编程方式来做到对属性指定相应的值呢？

有两种方式可以做到：

（1）使用 BeanWrapper 实现属性指定（上篇）；
（2）使用 DataBinder 来实现属性指定。



示例程序，Java Bean 如下：

```
package com.davidwang456.test;

import java.util.Date;

import lombok.Data;
@Data
public class POJOBean {
    private int age;
    private String name;
    private Date birthday;
    @Override
    public String toString () {
        return "POJOBean{name=" + name+",age="+age+",birthday="+birthday.toGMTString()+ '}';
    }
}
```

测试类：

```
package com.davidwang456.test;

import java.util.Date;

import org.springframework.beans.MutablePropertyValues;
import org.springframework.validation.DataBinder;

public class DataBinderExample {
    public static void main (String[] args) {
        MutablePropertyValues mpv = new MutablePropertyValues();
        mpv.add("name", "davidwang456");
        mpv.add("age", "10x");
        mpv.add("birthday", new Date());
        DataBinder db = new DataBinder(new POJOBean());
        db.bind(mpv);

        db.getBindingResult()
                    .getAllErrors()
                    .stream()
                    .forEach(System.out::println);
    }
}
```

现在我们可以开始 debug 了。

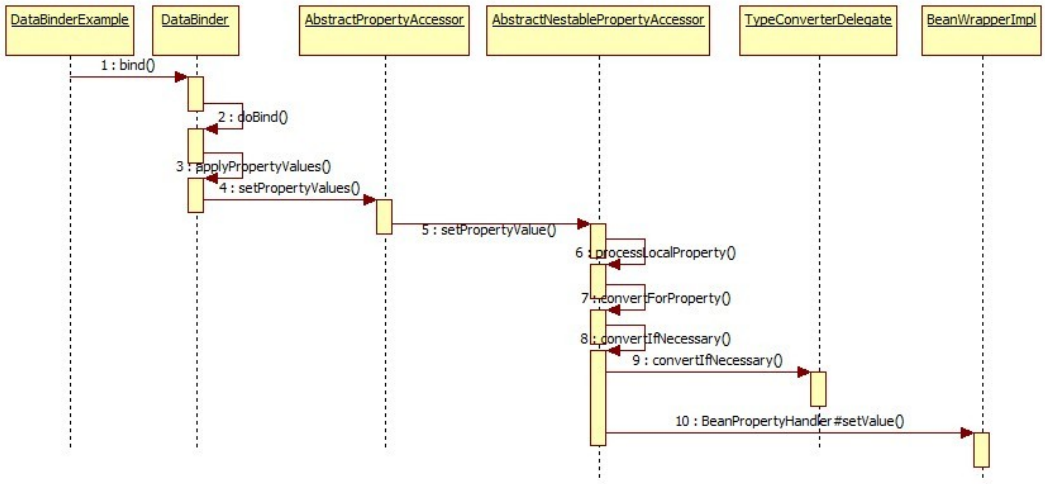
原理

通过一层层 debug，DataBinder 通过 doBind() 方法来做属性的检查和赋值：

1. 检查属性是否为可以绑定的属性，并删除不是可以绑定的属性；
2. 检查缺少字段错误：“required”；

3. 设置属性值到对应的属性上。

其完整的调用链如下图所示：



最底层是通过 BeanWrapperImpl 实现，中间商是 TypeConverterDelegate。

重点：BeanWrapperImpl.java #setValue：

```
@Override
public void setValue(final @Nullable Object value) throws Exception {
    final Method writeMethod = (this.pd instanceof GenericTypeAwarePropertyDescriptor ?
            ((GenericTypeAwarePropertyDescriptor) this.pd).getWriteMethodForActualAccess() :
            this.pd.getWriteMethod());
    if (System.getSecurityManager() != null) {
        AccessController.doPrivileged((PrivilegedAction<Object>) () -> {
            ReflectionUtils.makeAccessible(writeMethod);
            return null;
        });
        try {
            AccessController.doPrivileged((PrivilegedExceptionAction<Object>) () ->
                    writeMethod.invoke(getWrappedInstance(), value), acc);
        }
        catch (PrivilegedActionException ex) {
            throw ex.getException();
        }
    }
    else {
        ReflectionUtils.makeAccessible(writeMethod);
        writeMethod.invoke(getWrappedInstance(), value);
    }
}
}
```

最后，使用反射的方式触发属性方法 set 的执行。

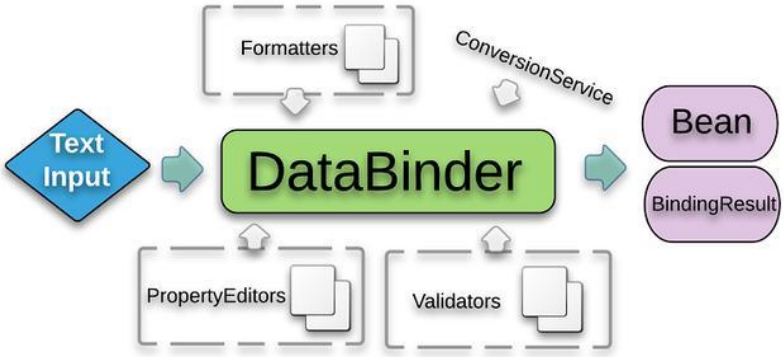
总结

Spring 中数据绑定的两种方式（BeanWrapper 或者 DataBinder），上面两个示例可以看出：

1. 当使用 BeanWrapper，属性转换异常的时候会抛出 org.springframework.beans.TypeMismatchException；
2. 当使用 DataBinder 时则将异常信息放入到 BindingResult，但不会抛出异常。

另外，DataBInder是一个可以在目标对象上设置属性值的绑定器，包括对验证和绑定结果分析的支持。

可以通过指定允许的字段来定制绑定过程，必需的字段，定制编辑器，等等。DataBInder 集成了 PropertyEditor，Validator，Formatter 或者 ConversionService 来完成上述功能。



DataBinder主要提供了两个功能：

1. 利用 BeanWrapper，给对象的属性设值；
2. 在设值的同时做 Validation。

使用校验 Validator 示例：

```
import java.util.Date;

import org.springframework.beans.MutablePropertyValues;
import org.springframework.validation.DataBinder;

public class DataBinderExample {
    public static void main (String[] args) {
        MutablePropertyValues mpv = new MutablePropertyValues();
        mpv.add("name", "davidwang456");
        mpv.add("age", "10");
        mpv.add("birthday", new Date());
        DataBinder db = new DataBinder(new POJOBean());

        db.bind(mpv);


        db.addValidators(new OrderValidator());
        db.validate();

        db.getBindingResult()
                    .getAllErrors()
                    .stream()
                    .forEach(System.out::println);
    }
}
```

其中：

```
package com.davidwang456.test;

import org.springframework.validation.Errors;
import org.springframework.validation.ValidationUtils;
import org.springframework.validation.Validator;

public class OrderValidator implements Validator {

    @Override
    public boolean supports (Class<?> clazz) {
        return POJOBean.class == clazz;
    }

    @Override
    public void validate (Object target, Errors errors) {
        ValidationUtils.rejectIfEmpty(errors, "birthday", "date empty");
        ValidationUtils.rejectIfEmpty(errors, "name", "name empty");
        ValidationUtils.rejectIfEmpty(errors, "age", "age empty");
        POJOBean order = (POJOBean) target;
        if (order.getAge() <= 0) {
            errors.rejectValue("age", "age invalid");
        }
    }
}
```

}

25 Spring Core 数据绑定之 BeanWrapper实现示例及背后原理探究

27 Spring MVC数据绑定 InitBinder揭秘