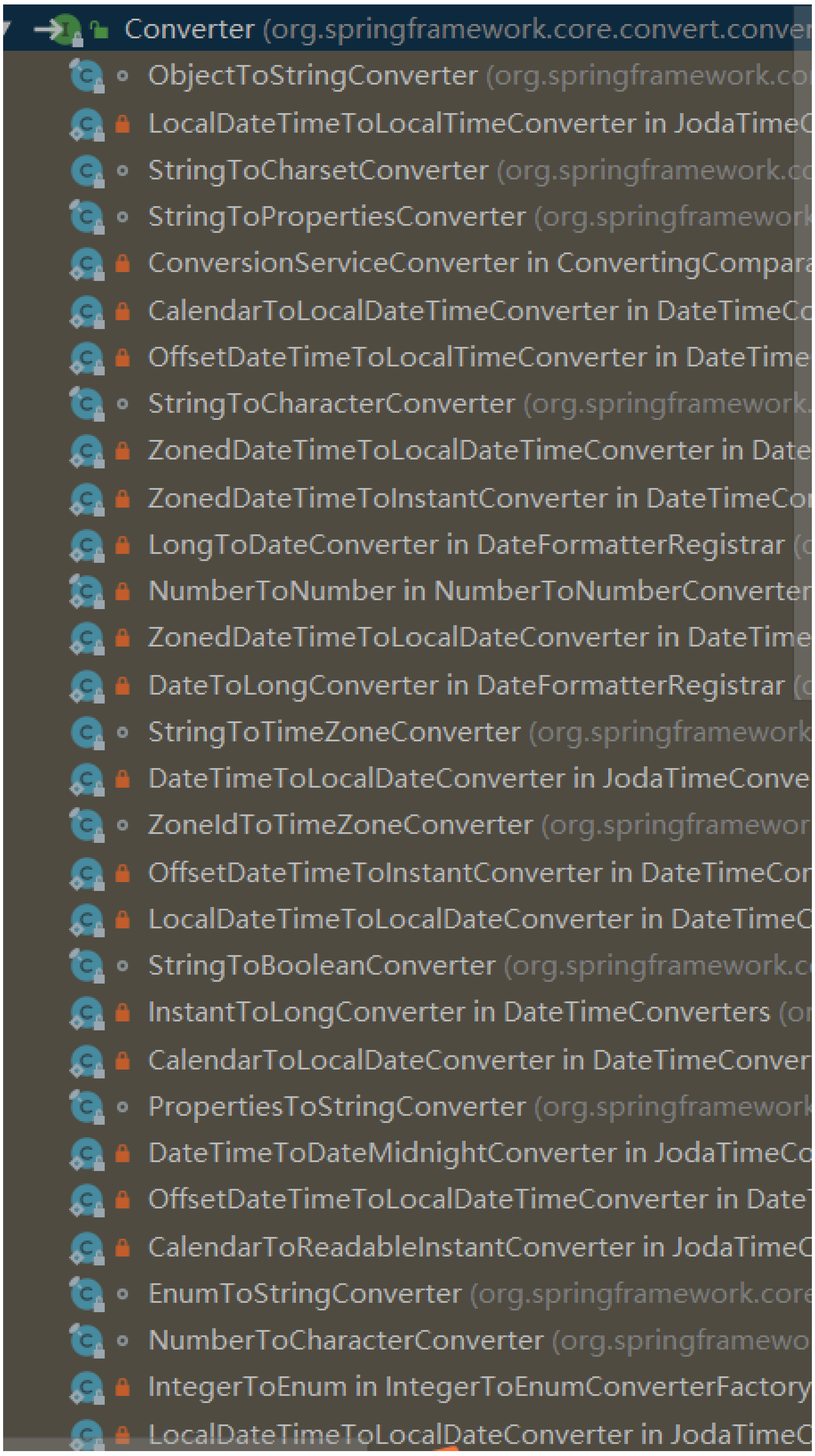
# 04-SpringMVC基于注解使用：类型转换&数据格式化&数据验证

## 1、类型转换器

在日常的企业开发需求中，我们输入文本框的内容全部都是字符串类型，但是在后端处理的时候我们可以用其他基本类型来接受数据，也可以使用实体类来接受参数，这个是怎么完成的呢？就是通过SpringMVC提供的类型转换器，SpringMVC内部提供了非常丰富的类型转换器的支持，但是有些情况下有可能难以满足我们的需求，因此需要我们自己实现，如下：


Converter (org.springframework.core.convert.conver
ObjectToStringConverter (org.springframework.co
LocalDateTimeToLocalTimeConverter in JodaTimeC
StringToCharsetConverter (org.springframework.co
StringToPropertiesConverter (org.springframework
ConversionServiceConverter in ConvertingCompara
CalendarToLocalDateTimeConverter in DateTimeCo
OffsetDateTimeToLocalTimeConverter in DateTime
StringToCharacterConverter (org.springframework.
ZonedDateTimeToLocalDateTimeConverter in Date
ZonedDateTimeToInstantConverter in DateTimeCo
LongToDateConverter in DateFormatterRegistrar (o
NumberToNumber in NumberToNumberConverter
ZonedDateTimeToLocalDateConverter in DateTime
DateToLongConverter in DateFormatterRegistrar (o
StringToTimeZoneConverter (org.springframework
DateTimeToLocalDateConverter in JodaTimeConve
ZoneIdToTimeZoneConverter (org.springframewor
OffsetDateTimeToInstantConverter in DateTimeCo
LocalDateTimeToLocalDateConverter in DateTimeC
StringToBooleanConverter (org.springframework.c
InstantToLongConverter in DateTimeConverters (or
CalendarToLocalDateConverter in DateTimeConver
PropertiesToStringConverter (org.springframework
DateTimeToDateMidnightConverter in JodaTimeCo
OffsetDateTimeToLocalDateTimeConverter in DateT
CalendarToReadableInstantConverter in JodaTimeC
EnumToStringConverter (org.springframework.core
NumberToCharacterConverter (org.springframewo
IntegerToEnum in IntegerToEnumConverterFactory
LocalDateTimeToLocalDateConverter in JodaTimeC

```
ConversionService converters =
    @org.springframework.format.annotation.DateTimeFormat java.lang.Long -> java.lang.Str
    @org.springframework.format.annotation.DateTimeFormat java.time.LocalDate -> java.lang
    @org.springframework.format.annotation.DateTimeFormat java.time.LocalDateTime -> java
    @org.springframework.format.annotation.DateTimeFormat java.time.LocalTime -> java.lang
    @org.springframework.format.annotation.DateTimeFormat java.time.OffsetDateTime -> java
    @org.springframework.format.annotation.DateTimeFormat java.time.OffsetTime -> java.la
```

```
@org.springframework.format.annotation.DateTimeFormat java.time.ZonedDateTime -> java
@org.springframework.format.annotation.DateTimeFormat java.util.Calendar -> java.lang
@org.springframework.format.annotation.DateTimeFormat java.util.Date -> java.lang.Str
@org.springframework.format.annotation.NumberFormat java.lang.Byte -> java.lang.Strin
@org.springframework.format.annotation.NumberFormat java.lang.Double -> java.lang.Str
@org.springframework.format.annotation.NumberFormat java.lang.Float -> java.lang.Stri
@org.springframework.format.annotation.NumberFormat java.lang.Integer -> java.lang.St
@org.springframework.format.annotation.NumberFormat java.lang.Short -> java.lang.Stri
@org.springframework.format.annotation.NumberFormat java.math.BigDecimal -> java.lang
@org.springframework.format.annotation.NumberFormat java.math.BigInteger -> java.lang
java.lang.Boolean -> java.lang.String : org.springframework.core.convert.support.Obje
java.lang.Character -> java.lang.Number : org.springframework.core.convert.support.Ch
java.lang.Character -> java.lang.String : org.springframework.core.convert.support.Ob
java.lang.Enum -> java.lang.Integer : org.springframework.core.convert.support.EnumTo
java.lang.Enum -> java.lang.String : org.springframework.core.convert.support.EnumToS
java.lang.Integer -> java.lang.Enum : org.springframework.core.convert.support.Intege
java.lang.Long -> java.time.Instant : org.springframework.format.datetime.standard.Da
java.lang.Long -> java.util.Calendar : org.springframework.format.datetime.DateFormat
java.lang.Long -> java.util.Date : org.springframework.format.datetime.DateFormatterR
java.lang.Number -> java.lang.Character : org.springframework.core.convert.support.Nu
java.lang.Number -> java.lang.Number : org.springframework.core.convert.support.Numbe
java.lang.Number -> java.lang.String : org.springframework.core.convert.support.Objec
java.lang.String -> @org.springframework.format.annotation.DateTimeFormat java.lang.L
java.lang.String -> @org.springframework.format.annotation.DateTimeFormat java.time.L
java.lang.String -> @org.springframework.format.annotation.DateTimeFormat java.time.L
java.lang.String -> @org.springframework.format.annotation.DateTimeFormat java.time.L
java.lang.String -> @org.springframework.format.annotation.DateTimeFormat java.time.O
java.lang.String -> @org.springframework.format.annotation.DateTimeFormat java.time.O
java.lang.String -> @org.springframework.format.annotation.DateTimeFormat java.time.Z
java.lang.String -> @org.springframework.format.annotation.DateTimeFormat java.util.C
java.lang.String -> @org.springframework.format.annotation.DateTimeFormat java.util.D
java.lang.String -> @org.springframework.format.annotation.NumberFormat java.lang.Byt
java.lang.String -> @org.springframework.format.annotation.NumberFormat java.lang.Dou
java.lang.String -> @org.springframework.format.annotation.NumberFormat java.lang.Flo
java.lang.String -> @org.springframework.format.annotation.NumberFormat java.lang.Int
java.lang.String -> @org.springframework.format.annotation.NumberFormat java.lang.Sho
java.lang.String -> @org.springframework.format.annotation.NumberFormat java.math.Big
java.lang.String -> @org.springframework.format.annotation.NumberFormat java.math.Big
java.lang.String -> java.lang.Boolean : org.springframework.core.convert.support.Stri
java.lang.String -> java.lang.Character : org.springframework.core.convert.support.St
java.lang.String -> java.lang.Enum : org.springframework.core.convert.support.StringTo
java.lang.String -> java.lang.Number : org.springframework.core.convert.support.String
java.lang.String -> java.nio.charset.Charset : org.springframework.core.convert.suppo
java.lang.String -> java.time.Duration: org.springframework.format.datetime.standard.
java.lang.String -> java.time.Instant: org.springframework.format.datetime.standard.I
java.lang.String -> java.time.Month: org.springframework.format.datetime.standard.Mon
java.lang.String -> java.time.MonthDay: org.springframework.format.datetime.standard.
java.lang.String -> java.time.Period: org.springframework.format.datetime.standard.Pe
java.lang.String -> java.time.Year: org.springframework.format.datetime.standard.Year
java.lang.String -> java.time.YearMonth: org.springframework.format.datetime.standard
java.lang.String -> java.util.Currency : org.springframework.core.convert.support.Str
java.lang.String -> java.util.Locale : org.springframework.core.convert.support.String
java.lang.String -> java.util.Properties : org.springframework.core.convert.support.S
java.lang.String -> java.util.TimeZone : org.springframework.core.convert.support.Str
java.lang.String -> java.util.UUID : org.springframework.core.convert.support.StringTo
java.nio.charset.Charset -> java.lang.String : org.springframework.core.convert.suppo
java.time.Duration -> java.lang.String : org.springframework.format.datetime.standard
java.time.Instant -> java.lang.Long : org.springframework.format.datetime.standard.Da
java.time.Instant -> java.lang.String : org.springframework.format.datetime.standard.
java.time.LocalDateTime -> java.time.LocalDate : org.springframework.format.datetime.
java.time.LocalDateTime -> java.time.LocalTime : org.springframework.format.datetime.
java.time.Month -> java.lang.String : org.springframework.format.datetime.standard.Mo
java.time.MonthDay -> java.lang.String : org.springframework.format.datetime.standard
java.time.OffsetDateTime -> java.time.Instant : org.springframework.format.datetime.s
java.time.OffsetDateTime -> java.time.LocalDate : org.springframework.format.datetime
java.time.OffsetDateTime -> java.time.LocalDateTime : org.springframework.format.date
java.time.OffsetDateTime -> java.time.LocalTime : org.springframework.format.datetime
java.time.OffsetDateTime -> java.time.ZonedDateTime : org.springframework.format.date
java.time.Period -> java.lang.String : org.springframework.format.datetime.standard.P
java.time.Year -> java.lang.String : org.springframework.format.datetime.standard.Yea
java.time.YearMonth -> java.lang.String : org.springframework.format.datetime.standar
java.time.ZoneId -> java.util.TimeZone : org.springframework.core.convert.support.Zon
java.time.ZonedDateTime -> java.time.Instant : org.springframework.format.datetime.st
java.time.ZonedDateTime -> java.time.LocalDate : org.springframework.format.datetime.
java.time.ZonedDateTime -> java.time.LocalDateTime : org.springframework.format.datet
java.time.ZonedDateTime -> java.time.LocalTime : org.springframework.format.datetime.
java.time.ZonedDateTime -> java.time.OffsetDateTime : org.springframework.format.date
java.time.ZonedDateTime -> java.util.Calendar : org.springframework.core.convert.supp
java.util.Calendar -> java.lang.Long : org.springframework.format.datetime.DateFormat
```

```
java.util.Calendar -> java.time.Instant : org.springframework.format.datetime.standar
java.util.Calendar -> java.time.LocalDate : org.springframework.format.datetime.standa
java.util.Calendar -> java.time.LocalDateTime : org.springframework.format.datetime.st
java.util.Calendar -> java.time.LocalTime : org.springframework.format.datetime.standa
java.util.Calendar -> java.time.OffsetDateTime : org.springframework.format.datetime.
java.util.Calendar -> java.time.ZonedDateTime : org.springframework.format.datetime.st
java.util.Calendar -> java.util.Date : org.springframework.format.datetime.DateFormatt
java.util.Currency -> java.lang.String : org.springframework.core.convert.support.Obj
java.util.Date -> java.lang.Long : org.springframework.format.datetime.DateFormatterR
java.util.Date -> java.util.Calendar : org.springframework.format.datetime.DateFormatt
java.util.Locale -> java.lang.String : org.springframework.core.convert.support.Objec
java.util.Properties -> java.lang.String : org.springframework.core.convert.support.P
java.util.UUID -> java.lang.String : org.springframework.core.convert.support.ObjectT
org.springframework.core.convert.support.ArrayToArrayConverter@2b19ffdb
org.springframework.core.convert.support.ArrayToCollectionConverter@4c019cc0
org.springframework.core.convert.support.ArrayToObjectConverter@343b143b
org.springframework.core.convert.support.ArrayToStringConverter@654a7e11
org.springframework.core.convert.support.ByteBufferConverter@209ae681
org.springframework.core.convert.support.ByteBufferConverter@209ae681
org.springframework.core.convert.support.ByteBufferConverter@209ae681
org.springframework.core.convert.support.ByteBufferConverter@209ae681
org.springframework.core.convert.support.CollectionToArrayConverter@66c9dab
org.springframework.core.convert.support.CollectionToCollectionConverter@470e5927
org.springframework.core.convert.support.CollectionToObjectConverter@6c6550c9
org.springframework.core.convert.support.CollectionToStringConverter@4c04c66a
org.springframework.core.convert.support.FallbackObjectToStringConverter@2b2ed015
org.springframework.core.convert.support.IdToEntityConverter@1661f15d,org.springframew
org.springframework.core.convert.support.MapToMapConverter@7292a9a
org.springframework.core.convert.support.ObjectToArrayConverter@7a8895dc
org.springframework.core.convert.support.ObjectToCollectionConverter@52fd0eec
org.springframework.core.convert.support.ObjectToOptionalConverter@f5a1070
org.springframework.core.convert.support.ObjectToOptionalConverter@f5a1070
org.springframework.core.convert.support.ObjectToOptionalConverter@f5a1070
org.springframework.core.convert.support.StreamConverter@3820b54f
org.springframework.core.convert.support.StreamConverter@3820b54f
org.springframework.core.convert.support.StreamConverter@3820b54f
org.springframework.core.convert.support.StreamConverter@3820b54f
org.springframework.core.convert.support.StringToArrayConverter@1c9b244
org.springframework.core.convert.support.StringToCollectionConverter@13163c

```

**自定义类型转换器**

User.java

```java
package cn.tulingxueyuan.bean;

public class User {

    private Integer id;
    private String name;
    private Integer age;
    private String gender;

    public User() {
   }

    public Integer getId() {
        return id;
   }

    public void setId(Integer id) {
        this.id = id;
   }

    public String getName() {
        return name;
   }

    public void setName(String name) {
        this.name = name;
   }

    public Integer getAge() {
        return age;
   }

    public void setAge(Integer age) {
        this.age = age;
   }

```

```java
    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    @Override
    public String toString() {
        return "User{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", age=" + age +
                ", gender='" + gender + '\'' +
                '}';
    }
}
```

MyConverter.java

```java
package cn.tulingxueyuan.converter;

import cn.tulingxueyuan.bean.User;
import org.springframework.core.convert.converter.Converter;
import org.springframework.stereotype.Component;

@Component
public class MyConverter implements Converter<String, User> {
    public User convert(String source) {
        User user = null;
        String[] split = source.split("-");
        if (source!=null && split.length==4){
            user = new User();
            user.setId(Integer.parseInt(split[0]));
            user.setName(split[1]);
            user.setAge(Integer.parseInt(split[2]));
            user.setGender(split[3]);
        }
        return user;
    }
}
```

UserController.java

```java
package cn.tulingxueyuan.controller;

import cn.tulingxueyuan.bean.User;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class UserController {

    @RequestMapping("/user")
    public String add(User user, Model model){
        System.out.println(user);
        model.addAttribute("user","user");
        return "success";
    }
}
```

success.jsp

```jsp
<%--
 Created by IntelliJ IDEA.
 User: root
 Date: 2020/3/12
 Time: 21:36
 To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
   <title>Title</title>
</head>
<body>
${requestScope.user}
</body>
```

```
</html>
```

springmvc.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:context="http://www.springframework.org/schema/context"
      xmlns:mvc="http://www.springframework.org/schema/mvc"

      xsi:schemaLocation="http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd
      http://www.springframework.org/schema/context
      https://www.springframework.org/schema/context/spring-context.xsd http://www.sprin

   <context:component-scan base-package="cn.tulingxueyuan"></context:component-scan>
   <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
       <property name="prefix" value="/WEB-INF/page/"></property>
       <property name="suffix" value=".jsp"></property>
   </bean>
   <bean class="cn.tulingxueyuan.view.MyViewResolver">
       <property name="order" value="1"></property>
   </bean>
   <mvc:annotation-driven conversion-service="conversionService"></mvc:annotation-driven
   <bean id="conversionService" class="org.springframework.context.support.ConversionSer
       <property name="converters">
           <set>
               <ref bean="myConverter"></ref>
           </set>
       </property>
   </bean>
</beans>

```

## 2、数据格式化

Spring 提供了两个可以用于格式化数字、日期和时间的注解@NumberFormat和@DateTimeFormat，这两个标签可以用于javabean的属性或方法参数上。@NumberFormat可以用

来格式化任何的数字的基本类型（如int，long）或java.lang.Number的实例（如 BigDecimal, Integer)。@DateTimeFormat可以用来格式化java.util.Date、java.util.Calendar和 java.util.Long类型.

要指定数字或日期/时间类型的属性，只需要在其上添加 @NumberFormat或@DateTimeFormat注解接可以了。例如下面的代码：

```java
import java.math.BigDecimal;
import java.util.Calendar;
import java.util.Date;
import org.joda.time.LocalTime;
import org.springframework.format.annotation.DateTimeFormat;
import org.springframework.format.annotation.NumberFormat;
import org.springframework.format.annotation.DateTimeFormat.ISO;
import org.springframework.format.annotation.NumberFormat.Style;
public class Employee {
private String name;
/**
* numeric fields using @NumberFormat annotation for formatting.
*/
@NumberFormat(style = Style.CURRENCY)
private double salary;
@NumberFormat(style = Style.PERCENT)
private double w4AdditionalWithdraw;
@NumberFormat
private int dependents;
@NumberFormat(pattern = "0.00")
private BigDecimal visualAcuity;
/**
* date and time fields using @DateTimeFormat annotation for formatting.
*/
@DateTimeFormat(style = "M-")
private Date birthDate;
@DateTimeFormat(pattern = "w:yyyy")
private Calendar hireDate;
@DateTimeFormat(style = "-S")
private LocalTime startTime;
@DateTimeFormat(iso = ISO.DATE_TIME)
private long lastTimeEntry;
/**
```

```java
* initialization block to provide sample data for display
*/
{
this.name = "John Doe";
this.salary = 30100.50;
this.w4AdditionalWithdraw = 0.02;
this.dependents = 5;
this.visualAcuity = new BigDecimal(".1");
Calendar dob = Calendar.getInstance();
dob.set(1964, Calendar.AUGUST, 30);
this.birthDate = dob.getTime();
this.hireDate = Calendar.getInstance();
this.startTime = new LocalTime(8, 0);
this.lastTimeEntry = (new Date()).getTime() - 10000;
}
public String getName() {
return name;
}
public void setName(String name) {
this.name = name;
}
public double getSalary() {
return salary;
}
public void setSalary(double salary) {
this.salary = salary;
}
public double getW4AdditionalWithdraw() {
return w4AdditionalWithdraw;
}
public void setW4AdditionalWithdraw(double w4AdditionalWithdraw) {
this.w4AdditionalWithdraw = w4AdditionalWithdraw;
}
public int getDependents() {
return dependents;
}
public void setDependents(int dependents) {
this.dependents = dependents;
}
public BigDecimal getVisualAcuity() {
```

```
return visualAcuity;
}
public void setVisualAcuity(BigDecimal visualAcuity) {
this.visualAcuity = visualAcuity;
}
public Date getBirthDate() {
return birthDate;
}
public void setBirthDate(Date birthDate) {
this.birthDate = birthDate;
}
public LocalTime getStartTime() {
return startTime;
}
public void setStartTime(LocalTime startTime) {
this.startTime = startTime;
}
public Calendar getHireDate() {
return hireDate;
}
public void setHireDate(Calendar hireDate) {
this.hireDate = hireDate;
}
public long getLastTimeEntry() {
return lastTimeEntry;
}
public void setLastTimeEntry(long lastTimeEntry) {
this.lastTimeEntry = lastTimeEntry;
}
}
```

## 注解类型

DateTimeFormat,互斥属性:

- **iso**。类型为DateTimeFormat.ISO
  - DateTimeFormat.ISO.DATE: 格式yyyy-MM-dd。
  - DateTimeFormat.ISO.DATE_TIME: 格式yyyy-MM-dd HH:mm:ss .SSSZ。
  - DateTimeFormat.ISO.TIME: 格式HH:mm:ss .SSSZ。
  - DateTimeFormat.ISO.NONE: 表示不使用ISO格式的时间。
- **pattern**。类型为String，使用自定义的时间格式化字符串。
- **style**。类型为String，通过样式指定日期时间的格式，由两位字符组成，第1位表示日期的样式，第2位表示时间的格式:

    - S: 短日期/时间的样式;
    - M: 中日期/时间的样式;
    - L: 长日期/时间的样式;
    - F: 完整日期/时间的样式;
    - -: 忽略日期/时间的样式;
- NumberFormat
  - **pattern**。类型为String，使用自定义的数字格式化字符串，"##,###.##"。
  - **style**。类型为NumberFormat.Style，常用值:
    - Style.NUMBER正常数字类型
    - Style.PERCENT百分数类型
    - Style.CURRENCY 货币类型

## 3、数据校验

一般情况下我们会在前端页面实现数据的校验，但是大家需要注意的是前端校验会存在数据的不安全问题，因此一般情况下我们都会使用前端校验+后端校验的方式，这样的话既能够满足用户的体验度，同时也能保证数据的安全，下面来说一下在springmvc中如何进行后端数据校验。

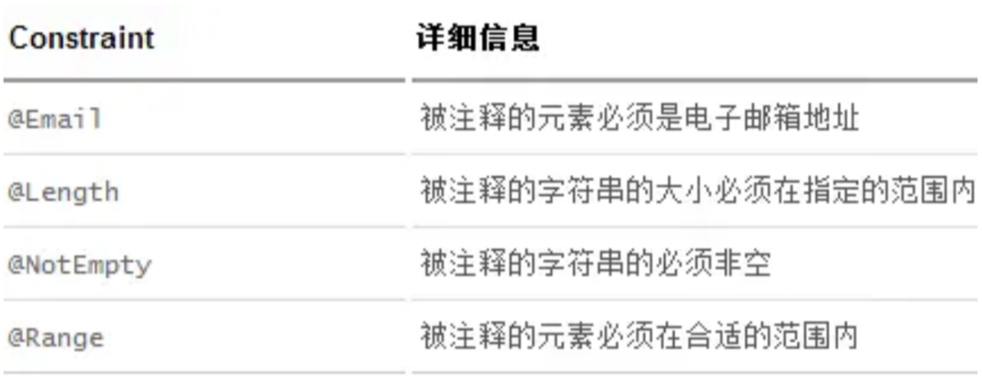
JSR303是 Java 为 Bean 数据合法性校验提供的标准，它已经包含在 JavaEE 6.0 中 。JSR 303 (Java Specification Requests意思是Java 规范提案)通过**在 Bean 属性上标注**类似于 @NotNull、@Max 等标准的注解指定校验规则，并通 j过标准的验证接口对 Bean 进行验证。

Hibernate Validator 实现了JSR349验证注解规范的技术

JSR303:

| Constraint | 详细信息 |
|---|---|
| @Null | 被注释的元素必须为 null |
| @NotNull | 被注释的元素必须不为 null |
| @AssertTrue | 被注释的元素必须为 true |
| @AssertFalse | 被注释的元素必须为 false |
| @Min(value) | 被注释的元素必须是一个数字，其值必须大于等于指定的最小值 |
| @Max(value) | 被注释的元素必须是一个数字，其值必须小于等于指定的最大值 |
| @DecimalMin(value) | 被注释的元素必须是一个数字，其值必须大于等于指定的最小值 |
| @DecimalMax(value) | 被注释的元素必须是一个数字，其值必须小于等于指定的最大值 |
| @Size(max, min) | 被注释的元素的大小必须在指定的范围内 |
| @Digits (integer, fraction) | 被注释的元素必须是一个数字，其值必须在可接受的范围内 |
| @Past | 被注释的元素必须是一个过去的日期 |
| @Future | 被注释的元素必须是一个将来的日期 |
| @Pattern(value) | 被注释的元素必须符合指定的正则表达式 |

Hibernate Validator 扩展注解:



| Constraint | 详细信息 |
|---|---|
| @Email | 被注释的元素必须是电子邮箱地址 |
| @Length | 被注释的字符串的大小必须在指定的范围内 |
| @NotEmpty | 被注释的字符串的必须非空 |
| @Range | 被注释的元素必须在合适的范围内 |

spring中拥有自己的数据校验框架，同时支持JSR303标准的校验框架，可以在通过添加注解的方式进行数据校验。在spring中本身没有提供JSR303的实现，需要导入依赖的包。
pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xs
    <modelVersion>4.0.0</modelVersion>

    <groupId>cn.tulingxueyuan</groupId>
    <artifactId>springmvc_viewResolver</artifactId>
    <version>1.0-SNAPSHOT</version>

    <dependencies>

        <!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>5.2.3.RELEASE</version>
        </dependency>

        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-validator</artifactId>
            <version>5.1.0.Final</version>
        </dependency>

    </dependencies>
</project>
```

index.jsp

```
<%--
 Created by IntelliJ IDEA.
 User: root
 Date: 2020/3/12
 Time: 15:23
 To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
 <head>
   <title>$Title$</title>
 </head>
 <body>
<form action="dataValidate" method="post">
编号：<input type="text" name="id"><br>
姓名：<input type="text" name="name"><br>
年龄：<input type="text" name="age"><br>
性别：<input type="text" name="gender"><br>
日期：<input type="text" name="birth"><br>
邮箱：<input type="text" name="email"><br>
 <input type="submit" value="提交">
</form>
 </body>
</html>
```

DataValidateController.java

```java
package cn.tulingxueyuan.controller;

import cn.tulingxueyuan.bean.User;
import org.springframework.stereotype.Controller;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.RequestMapping;

import javax.validation.Valid;


```

```java
@Controller
public class DataValidateController {
    @RequestMapping("/dataValidate")
    public String validate(@Valid User user, BindingResult bindingResult) {
        System.out.println(user);
        if (bindingResult.hasErrors()) {
            System.out.println("验证失败");
            return "redirect:/index.jsp";
        } else {
            System.out.println("验证成功");
            return "hello";
        }
    }
}
```

User.java

```java
package cn.tulingxueyuan.bean;

import org.hibernate.validator.constraints.Email;
import org.hibernate.validator.constraints.Length;
import org.springframework.format.annotation.DateTimeFormat;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Past;
import java.util.Date;

public class User {

    private Integer id;
    @NotNull
    @Length(min = 5,max = 10)
    private String name;
    private Integer age;
    private String gender;
    @Past
    @DateTimeFormat(pattern = "yyyy-MM-dd")
    private Date birth;
    @Email
    private String email;

    public User() {
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public Date getBirth() {
        return birth;
    }

    public void setBirth(Date birth) {
        this.birth = birth;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    @Override
    public String toString() {
        return "User{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", age=" + age +
                ", gender='" + gender + '\'' +
                ", birth=" + birth +
                ", email='" + email + '\'' +
                '}';
    }
}
```

此时大家发现在报错的地方无法出现错误提示:

### 3.1、原生的表单如何获取错误信息:

DataValidateController.java

```java
package cn.tulingxueyuan.controller;

import cn.tulingxueyuan.bean.User;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.validation.FieldError;
import org.springframework.web.bind.annotation.RequestMapping;

import javax.validation.Valid;
import java.util.HashMap;
import java.util.List;
import java.util.Map;


@Controller
public class DataValidateController {
    @RequestMapping("/dataValidate")
    public String validate(@Valid User user, BindingResult bindingResult, Model model) {
        System.out.println(user);
        Map<String,Object> errorsMap = new HashMap<String, Object>();
        if (bindingResult.hasErrors()) {
            System.out.println("验证失败");
            List<FieldError> fieldErrors = bindingResult.getFieldErrors();
            for (FieldError fieldError : fieldErrors) {
                System.out.println(fieldError.getDefaultMessage());
                System.out.println(fieldError.getField());
                errorsMap.put(fieldError.getField(),fieldError.getDefaultMessage());
            }
            model.addAttribute("errorInfo",errorsMap);
            return "add";
        } else {
            System.out.println("验证成功");
            return "hello";
        }
    }

    @RequestMapping("add")
    public String add(Model model){
        model.addAttribute("user",new User(1,"zhangsan",12,"女",null,"1234@qq.com"));
        return "add";
    }
}
```

add.jsp

```jsp
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
```

```
<html>
<head>
   <title>$Title$</title>
</head>
<body>
<form action="dataValidate" method="post">
编号：<input type="text" name="id">${errorInfo.id}<br>
姓名：<input type="text" name="name">${errorInfo.name}<br>
年龄：<input type="text" name="age">${errorInfo.age}<br/>
性别：<input type="text" name="gender">${errorInfo.gender}<br/>
日期：<input type="text" name="birth">${errorInfobirth}<br/>
邮箱：<input type="text" name="email">${errorInfo.email}<br/>
 <input type="submit" value="提交">
</form>
</body>
</html>
```

**3.2、Springmvc form**标签

自动绑定，自动回显数据， 如果是新增的情况下也需要保证有该标签所需的对象

Form标签

1. 支持全部http请求方法 比如method=”put”  put\delete 提交方式
2. 数据自动回显：需要使用modelAttribute指定数据的对象
3. 使用path来双向绑定属性
4. **动态数据绑定**：Select 、 checkboxes、 radiobottons、 都可以使用Items 制定数据源 可以是list （当List的泛型是javaBean的时候需要制定itemValue和itemLabel）、map(不需要制定itemValue和itemLabel)

add.jsp

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
   <title>$Title$</title>
</head>
<body>
<form:form action="dataValidate"  modelAttribute="user" method="post">
  id:<form:input path="id"></form:input><form:errors path="id"></form:errors> <br/>
```

```
  name:<form:input path="name"></form:input><form:errors path="name"></form:errors><br/>
  age:<form:input path="age"></form:input><form:errors path="age"></form:errors><br/>
  gender:<form:input path="gender"></form:input><form:errors path="gender"></form:errors
  birth:<form:input path="birth"></form:input><form:errors path="birth"></form:errors><b
  email:<form:input path="email"></form:input><form:errors path="email"></form:errors><b
   <input type="submit" value="submit">
</form:form>
</body>
</html>
```

index.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
 <head>
   <title>$Title$</title>
 <body>
<a href="add">添加用户</a>

 </body>
</html>
```

DataValidateController.java

```java
package cn.tulingxueyuan.controller;

import cn.tulingxueyuan.bean.User;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.RequestMapping;

import javax.validation.Valid;


@Controller
public class DataValidateController {
    @RequestMapping("/dataValidate")
    public String validate(@Valid User user, BindingResult bindingResult, Model model) {
        System.out.println(user);
        if (bindingResult.hasErrors()) {
            System.out.println("验证失败");
            return "add";
        } else {
            System.out.println("验证成功");
            return "hello";
        }
    }

    @RequestMapping("add")
    public String add(Model model){
        model.addAttribute("user",new User(1,"zhangsan",12,"女",null,"1234@qq.com"));
        return "add";
    }
}
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/
        version="4.0">
    <servlet>
        <servlet-name>springmvc</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:springmvc.xml</param-value>
        </init-param>

    </servlet>
    <servlet-mapping>
        <servlet-name>springmvc</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
    <filter>
        <filter-name>encoding</filter-name>
        <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-clas
        <init-param>
            <param-name>encoding</param-name>
```

```xml
            <param-value>UTF-8</param-value>
        </init-param>
        <init-param>
            <param-name>forceEncoding</param-name>
            <param-value>true</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>encoding</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
</web-app>
```