Leader.us

2016年版教程

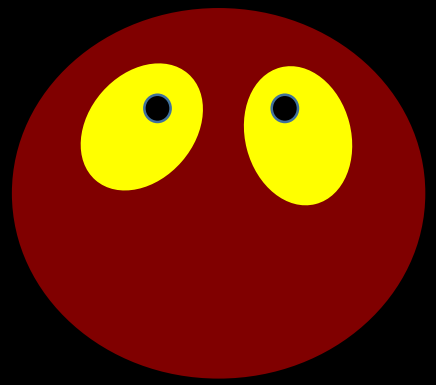
SPRING+架构篇

SpringBoot篇

Expert Java Leader.us

MVC第六篇

类型转换+数据校验



基于注解的类型转换和格式化功能

Expert Java Leader.us

提供类型转换器注册支持，运行时类型转换API支持。

Printer接口：格式化显示接口，将T类型的对象根据Locale信息以某种格式进行打印显示

解析接口，根据Locale信息解析字符串到T类型的对象

ConverterRegistry

Printer<T>

Parser<T>

①注册格式化转换器

FormatterRegistry

Formatter<T>

格式化SPI接口，继承Printer和Parser接口，完成T类型对象的格式化和解析功能；

②使用格式化转换器进行格式化

FormattingConversionService

继承自ConversionService，运行时类型转换和格式化服务接口，提供运行期类型转换和格式化的支持。

AnnotationFormatterFactory<...>

getFieldTypes() Set<Class<?>>

getPrinter(A, Class<?>) Printer<?>

getParser(A, Class<?>) Parser<?>

注解驱动的字段格式化工厂，用于创建带注解的对象字段的Printer和Parser，即用于格式化和解析带注解的对象字段。

DefaultFormattingConversionService

此处可以可以看出之前的Converter SPI完成任意Object与Object之间的类型转换，而Formatter SPI完成任意Object与String之间的类型转换（即格式化和解析，与PropertyEditor类似）。

Spring内建的Formater

Expert Java Leader.us

类名	说明
DateFormatter	java.util.Date<----->String 实现日期的格式化/解析
NumberFormatter	java.lang.Number<----->String 实现通用样式的格式化/解析
CurrencyFormatter	java.lang.BigDecimal<----->String 实现货币样式的格式化/解析
PercentFormatter	java.lang.Number<----->String 实现百分数样式的格式化/解析
NumberFormatAnnotationFormatterFactory	@NumberFormat注解类型的数字字段类型<----->String ①通过@NumberFormat指定格式化/解析格式 ②可以格式化/解析的数字类型：Short、Integer、Long、Float、Double、BigDecimal、BigInteger
JodaDateTimeFormatAnnotationFormatterFactory	@DateTimeFormat注解类型的日期字段类型<----->String ①通过@DateTimeFormat指定格式化/解析格式 ②可以格式化/解析的日期类型： joda中的日期类型（org.joda.time包中的）：LocalDate、LocalDateTime、LocalTime、ReadableInstant java内置的日期类型：Date、Calendar、Long classpath中必须有Joda-Time类库，否则无法格式化日期类型

```java
@Configuration
public class SpringConfig extends WebMvcConfigurerAdapter{

    @Override
    public void addFormatters(FormatterRegistry registry) {
        registry.addFormatterForFieldAnnotation(new TimestampFormatAnnotationFormatterFactory());
        super.addFormatters(registry);
    }

}
```

注册自定义的Formatter

基础的重要性_兰彻_新浪博客
2012年12月29日 - 万丈高楼平地起,再怎么强调基础的重要性都不为过,不仅对于学习知识领域,在很多...所以,每个扎扎实实努力的人要对社会有信心,相信自己踏实做事,干好每...
blog.sina.com.cn/s/blo... - 百度快照 - 4502条评价

形容基础重要的名言都有什么?_百度知道
2个回答 - 提问时间: 2008年01月04日
出处:《老子·德经》六十四章 原文:合抱之木,生于毫末;九层之台,起于累土;千里之行,始于足下。
更多关于基础很重要的问题>>

日期转换问题

Expert Java Leader.us

```java
@RequestMapping(value = "/showdate", method = RequestMethod.GET)
@ResponseBody
public String showdate(@RequestParam("date") Date theDate) {
    System.out.println("theDate " + theDate);
    return "Success ";
}
```

ObjectToObjectConverter

```java
public Object convert(Object source, TypeDescriptor sourceType, TypeDescriptor targetType) {
    if (source == null) {
        return null;
    }
    Class<?> sourceClass = sourceType.getType();
    Class<?> targetClass = targetType.getType();
    Member member = getValidatedMember(targetClass, sourceClass);

    try {
        if (member instanceof Method) {
            Method method = (Method) member;
            ReflectionUtils.makeAccessible(method);
            if (!Modifier.isStatic(method.getModifiers())) {
                return method.invoke(source);
            }
            else {
                return method.invoke(null, source);
            }
        }
        else if (member instanceof Constructor) {
            Constructor<?> ctor = (Constructor<?>) member;
            ReflectionUtils.makeAccessible(ctor);
            return ctor.newInstance(source);
        }
    }
    catch (InvocationTargetException ex) {
        throw new ConversionFailedException(sourceType, targetType, source, ex.getTargetException());
    }
```

http://localhost:8080/showdate?date=2017-4-4

Failed to convert from type [java.lang.String] to type [@org.springframework.web.bind.annotation.RequestParam java.util.Date] for value '2017-4-4'; nested exception is java.lang.IllegalArgumentException

- spring.mvc.date-format: 设定日期的格式，比如dd/MM/yyyy
- spring.mvc.locale: 指定使用的Locale

日期转换问题2

Expert Java Leader.us

@DateTimeFormat可以用来格式化java.util.Date、java.util.Calendar和 java.util.Long类型，也可以用于Joda Time类型的字段或参数。（Joda Time是一个开源的包，提供了对date和time类的一些替代类）In Spring 4.0, **@DateTimeFormat annotation can be used with Java 8 Date-Time API (java.time) out-of-the-box, without extra effort.**

```
@DateTimeFormat(style = "M-")
@DateTimeFormat(pattern = "w:yyyy")
@DateTimeFormat(style = "-S")
```

```
@RequestMapping(value = "/showdate", method = RequestMethod.GET)
@ResponseBody
public String showdate(@RequestParam("date") @DateTimeFormat(pattern = "yyyy-MM-dd") Date theDate) {
    System.out.println("theDate " + theDate);
    return "Success ";
}
```

设置全局默认日期格式

spring.mvc.locale

```
application.hellowmsg:  Leader Spring Boot
logging.level.=DEBUG
spring.mvc.date-format=yyyy/MM/dd
#logging.level.org.springframework.web.servlet.DispatcherSe
spring.datasource.url=jdbc:mysql://localhost:3306/leaderspr
spring.datasource.username=root
spring.datasource.password=123456
```

http://localhost:8080/showdate2?date=2017/4/4

Spring 4.0 brings Jsr310DateTimeFormatAnnotationFormatterFactory that formats Java 8 Date-Time fields annotated with the @DateTimeFormat. Supported field types are as follows:

- java.time.LocalDate
- java.time.LocalTime
- java.time.LocalDateTime
- java.time.ZonedDateTime
- java.time.OffsetDateTime
- java.time.OffsetTime

日期转换问题2

Expert Java Leader.us

```java
@Configuration
public class MyWebMvcContext extends WebMvcConfigurerAdapter {

    @Override
    public void addFormatters(FormatterRegistry registry) {
        registry.addConverter(new MyDateConverter("yyyy-MM-dd"));
    }

    final class MyDateConverter implements Converter<String, Date> {

        private final SimpleDateFormat formatter;

        public MyDateConverter(String dateFormat) {
            this.formatter = new SimpleDateFormat(dateFormat);
        }

        @Override
        public Date convert(String source) {
            if (source == null || source.isEmpty()) {
                return null;
            }

            try {
                return formatter.parse(source);
            } catch (ParseException e) {
                throw new RuntimeException(e);
            }
        }
    }
}
```

数字转换

Expert Java Leader.us

Spring 3 提供了@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)。

@NumberFormat

注解有两个可选的属性：style和pattern。style属性是一个NumberFormat.Style枚举值，可以是以下的三个值之一：

- **NUMBER** 缺省值
- **CURRENCY**
- **PERCENT**

具体的style的表现形式是与区域相关的。
例如，一个double类型的字段，如果style是CURRENCY，那么在en-us的区域显示的时候前面会加上$，在zh-cn的区域显示的时候前面会加上￥。如果以上的3中方式无法满足需求，我们可以使用pattern属性来指定特殊的输出格式。Pattern的值要遵循Java标准的numeric formatting pattern。

Expert Java Leader.us

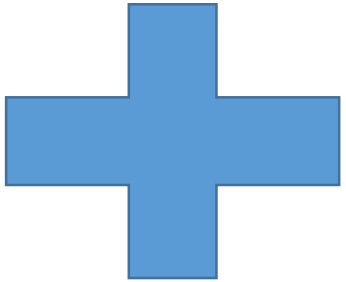
Validation

Spring支持的2种数据验证方式

Expert Java Leader.us

JSR-303 Validation

Bean Validation



Spring Validator

Bean Validation

Expert Java Leader.us

JSR 303 – Bean Validation 是一个数据验证的规范，2009 年 11 月确定最终方案。2009 年 12 月 Java EE 6 发布， Bean Validation 作为一个重要特性被包含其中。 **Hibernate Validator 是 Bean Validation 的参考实现， Hibernate Validator 5.x is the reference implementation Bean Validation 1.1**（ JSR 349）**!**. Hibernate Validator 提供了 JSR 303 规范中所有内置 constraint 的实现，除此之外还有一些附加的 constraint。 While we envision supporting and leveraging Java 8 as the "main theme" of **Bean Validation 2.0,Lambda expressions might be a useful** vehicle to express small ad-hoc validation routines.

Bean Validation 1.1 Specification

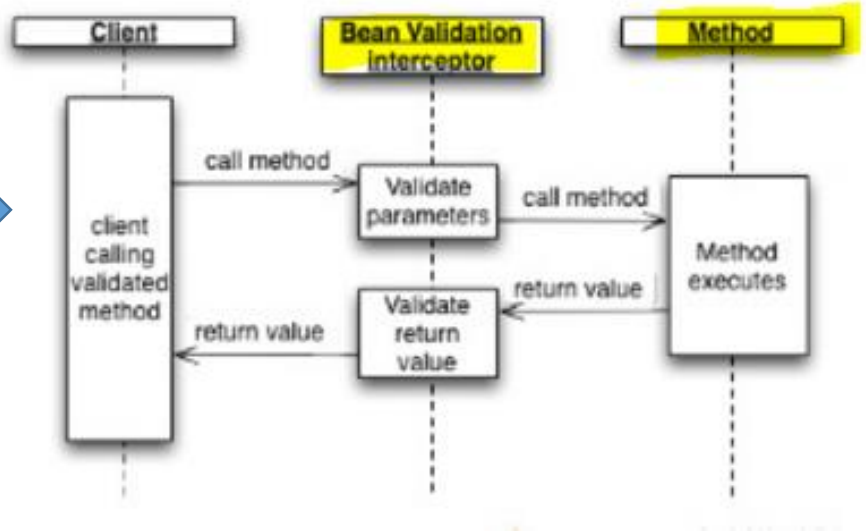
You can read the full Bean Validation 1.1 specification or browse its API JavaDocs.

Bean Validation - Bean Validation 2.0 - A new JSR is born!
beanvalidation.org/news/2016/07/15/bean-validation-2-0-is-coming/ ▾ 翻译此页
2016年7月15日 - Bean Validation 1.0 and 1.1 (JSRs 303/349) saw a huge adoption by the Java community and are integrated with a wide range of technologies, ...

Changes between Bean Validation 1.0 and 1.1

Bean Validation 1.1 focused on the following topics:

- openness of the specification and its process
- method-level validation (validation of parameters or return values)
- dependency injection for Bean Validation components
- integration with Context and Dependency Injection (CDI)
- group conversion
- error message interpolation using EL expressions



Bean Validation 2

Expert Java Leader.us

在Bean的属性、方法（参数、返回值）上定义校验规则（Constraint），其他的事情就交给框架处理了

In the following example, a constraint is placed on a field using the built-in `@NotNull` constraint:

```
public class Name {
    @NotNull
    private String firstname;

    @NotNull
    private String lastname;
}
```

You can also place more than one constraint on a single JavaBeans component object. For example, you can place an additional constraint for size of field on the `firstname` and the `lastname` fields:

```
public class Name {
    @NotNull
    @Size(min=1, max=16)
    private String firstname;

    @NotNull
    @Size(min=1, max=16)
    private String lastname;
}
```

The following example shows a method with a user-defined constraint that checks for a predefined email address pattern such as a corporate email account:

```
@ValidEmail
public String getEmailAddress() {
    return emailAddress;
}
```

Bean Validation 3

Expert Java Leader.us

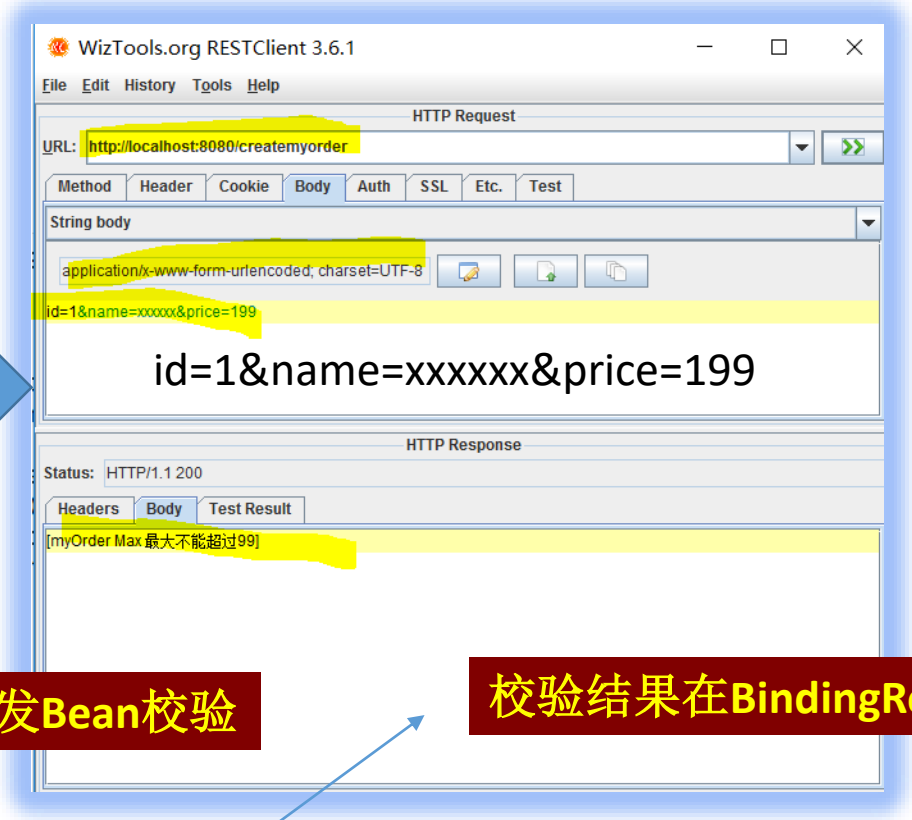
Constraints can be built in or user defined. User-defined constraints are called custom constraints. Several built-in constraints are available in **the javax.validation.constraints** package.

Constraint	Description	Example
@AssertFalse	The value of the field or property must be false.	@AssertFalse boolean isUnsupported;
@AssertTrue	The value of the field or property must be true.	@AssertTrue boolean isActive;
@DecimalMax	The value of the field or property must be a decimal value lower than or equal to the number in the value element.	@DecimalMax("30.00") BigDecimal discount;
@DecimalMin	The value of the field or property must be a decimal value greater than or equal to the number in the value element.	@DecimalMin("5.00") BigDecimal discount;
@Digits	The value of the field or property must be a number within a specified range. The integer element specifies the maximum integral digits for the number, and the fraction element specifies the maximum fractional digits for the number.	@Digits(integer=6, fraction=2) BigDecimal price;
@Future	The value of the field or property must be a date in the future.	@Future Date eventDate;
@Max	The value of the field or property must be an integer value lower than or equal to the number in the value element.	@Max(10) int quantity;
@Min	The value of the field or property must be an integer value greater than or equal to the number in the value element.	@Min(5) int quantity;
@NotNull	The value of the field or property must not be null.	@NotNull String username;
@Null	The value of the field or property must be null.	@Null String unusedString;
@Past	The value of the field or property must be a date in the past.	@Past Date birthday;
@Pattern	The value of the field or property must match the regular expression defined in the regexp element.	@Pattern(regexp="\\(\\d{3}\\)\\d{3}-\\d{4}") String phoneNumber;
@Size	The size of the field or property is evaluated and must match the specified boundaries. If the field or property is a String, the size of the string is evaluated. If the field or property is a Collection, the size of the Collection is evaluated. If the field or property is a Map, the size of the Map is evaluated. If the field or property is an array, the size of the array is evaluated. Use one of the optional max or min elements to specify the boundaries.	@Size(min=2, max=240) String briefMes

Bean Validation例子

Expert Java Leader.us

```java
public class MyOrder {

    private Integer id;
    @NotNull
    private String name;
    @Min(9)
    @Max(99)
    private int price;
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
```



@Valid触发Bean校验

校验结果在BindingResult里

```java
@RequestMapping(value = "/createmyorder", method = RequestMethod.POST, consumes = MediaType.APPLICATION_FORM_URI
public String createMyOrder(@Valid MyOrder requestForm, BindingResult result) {
    System.out.println("received body " + requestForm);
    if (result.hasErrors()) {
        Object[] errmsg = result.getAllErrors().stream()
                .map(o -> o.getObjectName() + " " + o.getCode() + " " + o.getDefaultMessage()).toArray();
        return Arrays.toString(errmsg);
    }
    return "Success ";
}
```

得到Error信息并处理

Validation依赖的类库

Expert Java Leader.us

- classmate-1.3.3.jar
- fastjson-1.2.22.jar
- hibernate-validator-5.2.4.Final.jar
- jackson-annotations-2.8.4.jar
- jackson-core-2.8.4.jar
- jackson-databind-2.8.4.jar
- jboss-logging-3.3.0.Final.jar
- jcl-over-slf4j-1.7.21.jar
- jul-to-slf4j-1.7.21.jar
- log4j-over-slf4j-1.7.21.jar
- logback-classic-1.1.7.jar
- logback-core-1.1.7.jar
- spring-core-4.3.4.RELEASE.jar
- spring-expression-4.3.4.RELEASE.jar
- spring-jdbc-4.3.4.RELEASE.jar
- spring-tx-4.3.4.RELEASE.jar
- spring-web-4.3.4.RELEASE.jar
- spring-webmvc-4.3.4.RELEASE.jar
- tomcat-embed-core-8.5.6.jar
- tomcat-embed-el-8.5.6.jar
- tomcat-embed-websocket-8.5.6.jar
- tomcat-jdbc-8.5.6.jar
- tomcat-juli-8.5.6.jar
- validation-api-1.1.0.Final.jar

Validation错误提示信息

hibernate-validator-5.2.4.Final.jar

自定义错误提示信息，并且国际化

Expert Java Leader.us

- HibernateValidator.class
- HibernateValidatorConfiguration.class
- HibernateValidatorContext.class
- HibernateValidatorFactory.class
- ValidationMessages.properties
- ValidationMessages_cs.properties
- ValidationMessages_de.properties
- ValidationMessages_en.properties
- ValidationMessages_es.properties
- ValidationMessages_fr.properties
- ValidationMessages_hu.properties
- ValidationMessages_ko.properties
- ValidationMessages_mn_MN.properties
- ValidationMessages_pt_BR.properties
- ValidationMessages_tr.properties
- ValidationMessages_zh_CN.properties

```
javax.validation.constraints.AssertFalse.message = \u53EA\u80FD\u4E3Afalse
javax.validation.constraints.AssertTrue.message  = \u53EA\u80FD\u4E3Atrue
javax.validation.constraints.DecimalMax.message  = \u5FC5\u987B\u5C0F\u4E8E\u6216\u7B49\u4E8E{value}
javax.validation.constraints.DecimalMin.message  = \u5FC5\u987B\u5927\u4E8E\u6216\u7B49\u4E8E{value}
javax.validation.constraints.Digits.message      = \u6570\u5B57\u7684\u503C\u8D85\u51FA\u4E86\u5141\u8BB8\u8303\u56F
javax.validation.constraints.Future.message      = \u9700\u8981\u662F\u4E00\u4E2A\u5C06\u6765\u7684\u65F6\u95F4
javax.validation.constraints.Max.message         = \u6700\u5927\u4E0D\u80FD\u8D85\u8FC7{value}
javax.validation.constraints.Min.message         = \u6700\u5C0F\u4E0D\u80FD\u5C0F\u4E8E{value}
javax.validation.constraints.NotNull.message     = \u4E0D\u80FD\u4E3Anull
javax.validation.constraints.Null.message        = \u5FC5\u987B\u4E3Anull
javax.validation.constraints.Past.message        = \u9700\u8981\u662f\u4e00\u4e2a\u8fc7\u53bb\u7684\u65F6\u95F4
javax.validation.constraints.Pattern.message     = \u9700\u8981\u5339\u914D\u6B63\u5219\u8868\u8FBE\u5F0F"{regexp}"
javax.validation.constraints.Size.message        = \u4E2A\u6570\u5FC5\u987B\u5728{min}\u548C{max}\u4E4B\u95F4
```

```
public class MyOrder {

    private Integer id;
    @Size(min=3,max=6,message="{myorder.name.invalid} ")
    private String name;
    @Min(9)
    @Max(99)
```

- src/main/resources
  - static
  - application.properties
  - ValidationMessages.properties

```
1 myorder.name.invalid=\u65E0\u6548\u7684\u7528\u6237\u540D
```

EL表达式更好的展示错误信息

Expert Java Leader.us

```
@Size(min = 5, message = "\"${validatedValue}\" is too short.")
private String bidder;


@Future(message = "The value \"${formatter.format('%1$tY-%1$tm-%1$td', validatedValue)}\" is not in future!")
private Date expiresAt;
```

Bean Validation+Rest接口的建议

Expert Java Leader.us

OperateResult

String errorCode;
String errorMsg;
Object **value**

→ **Errors**

public OperateResult doxxxx(....)

Group Validation

Expert Java Leader.us

```java
public class MyOrder {

    public interface ForUnpdate {
    };

    public interface ForGrounding {
    };

    private Integer id;
    @Size(min = 3, max = 6, message = "{myorder.name.invalid}
    private String name;
    @Min(9)
    @Max(99)
    private int price;

    @NotNull(groups = { MyOrder.ForGrounding.class })
    private Integer shopId;

    public Integer getId() {
        return id;
    }
```

ForGrounding业务操作的时候，
只校验shopId不为空

```java
}
@RequestMapping(value = "/createmyorder2", method = RequestMethod.POST, consumes = MediaType.A
public String createMyOrder2(@Validated({MyOrder.ForGrounding.class}) MyOrder requestForm, Bir
    System.out.println("received body " + requestForm);
    if (result.hasErrors()) {
        Object[] errmsg = result.getAllErrors().stream()
                .map(o -> o.getObjectName() + " " + o.getCode() + " " + o.getDefaultMessage())
        return Arrays.toString(errmsg);
    }
    return "Success ";
}
```

可以多个group来校验， groups = {xx.class,yyyy.class}
。多个group的时候，可以用@GroupSequence指定分组验证顺序：

```java
@GroupSequence({First.class, Second.class, User.class})
public class User implements Serializable {
```

Group Validation 2

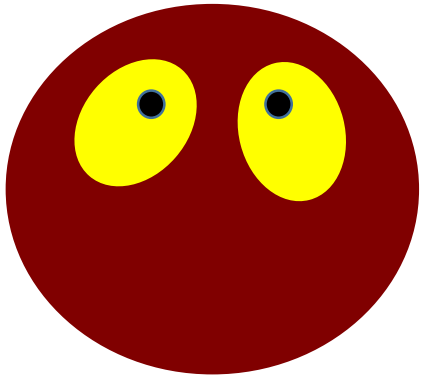
Expert Java Leader.us

What if you wanted to validate the car related checks together with the driver checks? Of course you could pass the required groups to the validate call explicitly, but what if you wanted to make these validations occur as part of the Default group validation? Here @ConvertGroup comes into play which allows you during cascaded validation to use a different group than the originally requested one.

```java
package org.hibernate.validator.referenceguide.chapter05.groupconversion;

@GroupSequence({ CarChecks.class, Car.class })
public class Car {

    @NotNull
    private String manufacturer;

    @NotNull
    @Size(min = 2, max = 14)
    private String licensePlate;

    @Min(2)
    private int seatCount;

    @AssertTrue(
            message = "The car has to pass the vehicle inspection first",
            groups = CarChecks.class
    )
    private boolean passedVehicleInspection;

    @Valid
    @ConvertGroup(from = Default.class, to = DriverChecks.class)
    private Driver driver;

    public Car(String manufacturer, String licencePlate, int seatCount) {
        this.manufacturer = manufacturer;
        this.licensePlate = licencePlate;
        this.seatCount = seatCount;
    }

    // getters and setters ...
}
```

需要联合@Valid注解使用

Spring self Validation



Spring Validator

org.springframework.validation.Validator

Expert Java Leader.us

A validator for application-specific objects. This interface is totally divorced from any infrastructure or context; that is to **say it is not coupled to validating only objects in the web tier, the data-access tier, or the whatever-tier. As such it is amenable to being used in any layer of an application,** and supports the encapsulation of validation logic as a first-class citizen in its own right.

Validator

- supports(Class<?>) : boolean
- validate(Object, Errors) : void

Using → **Errors**

Stores and exposes information about data-binding and validation errors for a specific object

SmartValidator

```
void validate(Object target, Errors errors, Object... validationHints);
```

This variant of validate() supports **validation hints**, such as validation groups against a JSR-303 provider (in which case, the provided hint objects need to be annotation arguments of type Class).

ModelAttributeMethodProcessor

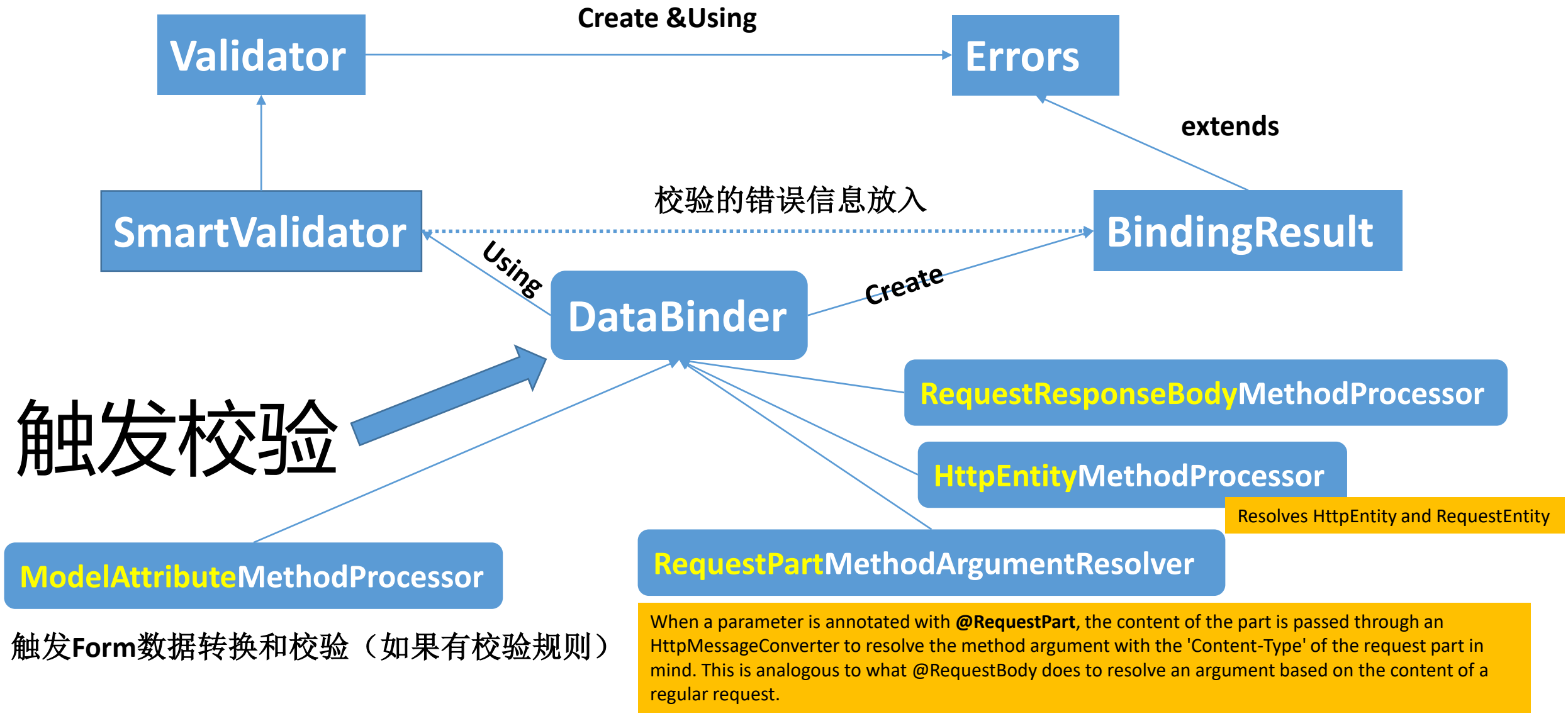
```
protected void validateIfApplicable(WebDataBinder binder, MethodParameter methodParam) {
    Annotation[] annotations = methodParam.getParameterAnnotations();
    for (Annotation ann : annotations) {
        Validated validatedAnn = AnnotationUtils.getAnnotation(ann, Validated.class);
        if (validatedAnn != null || ann.annotationType().getSimpleName().startsWith("Valid")) {
            Object hints = (validatedAnn != null ? validatedAnn.value() : AnnotationUtils.getValue(ann));
            Object[] validationHints = (hints instanceof Object[] ? (Object[]) hints : new Object[] {hints});
            binder.validate(validationHints);
            break;
        }
    }
}
```

DataBinder

```
public void validate(Object... validationHints) {
    for (Validator validator : getValidators()) {
        if (!ObjectUtils.isEmpty(validationHints) && validator instanceof SmartValidator) {
            ((SmartValidator) validator).validate(getTarget(), getBindingResult(), validationHints);
        }
        else if (validator != null) {
            validator.validate(getTarget(), getBindingResult());
        }
    }
}
```

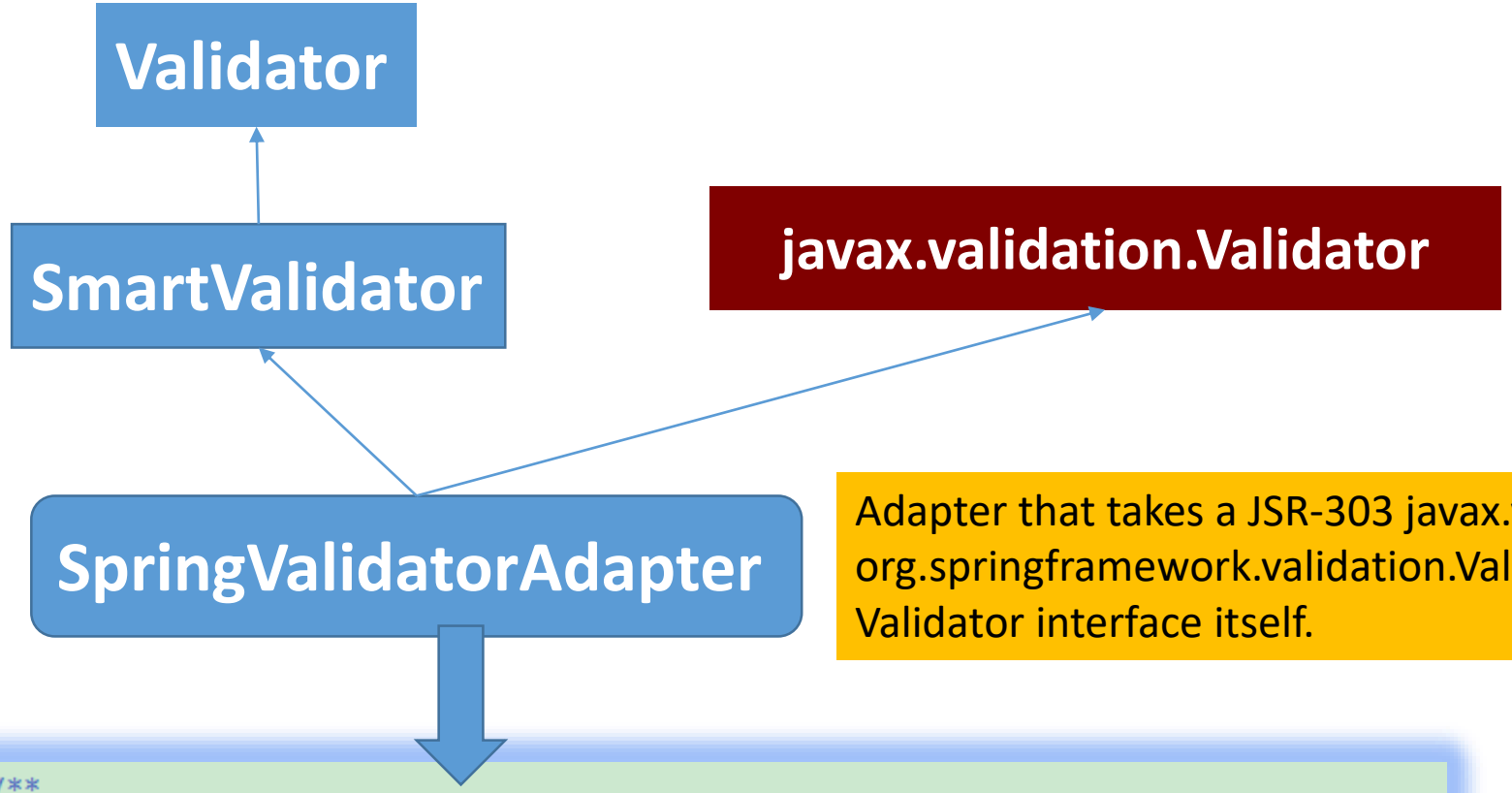
Spring Validator2

Expert Java Leader.us



Spring Validator3

Expert Java Leader.us



Adapter that takes a JSR-303 javax.validator.Validator and exposes it as a Spring org.springframework.validation.Validator while also exposing the original JSR-303 Validator interface itself.

```
/**
 * Create a new SpringValidatorAdapter for the given JSR-303 Validator.
 * @param targetValidator the JSR-303 Validator to wrap
 */
public SpringValidatorAdapter(javax.validation.Validator targetValidator) {
    Assert.notNull(targetValidator, "Target Validator must not be null");
    this.targetValidator = targetValidator;
}
```

Spring Validator 4

Expert Java Leader.us

SpringValidatorAdapter

ValidatorFactory

This is the central class for javax.validation (JSR-303) setup in a Spring application context: It bootstraps a javax.validation.ValidationFactory and exposes it through the Spring org.springframework.validation.Validator interface as well as through the JSR-303 javax.validation.Validator interface and the javax.validation.ValidatorFactory interface itself.

LocalValidatorFactoryBean

```
public static GenericBootstrap byDefaultProvider() {
    return new GenericBootstrapImpl();
}
```

```
public void afterPropertiesSet() {
    Configuration<?> configuration;
    if (this.providerClass != null) {
        ProviderSpecificBootstrap bootstrap = Validation.byProvider(this.providerClass);
        if (this.validationProviderResolver != null) {
            bootstrap = bootstrap.providerResolver(this.validationProviderResolver);
        }
        configuration = bootstrap.configure();
    }
    else {
        GenericBootstrap bootstrap = Validation.byDefaultProvider();
        if (this.validationProviderResolver != null) {
            bootstrap = bootstrap.providerResolver(this.validationProviderResolver);
        }
        configuration = bootstrap.configure();
    }

    // Try Hibernate Validator 5.2's externalClassLoader(ClassLoader) method
    if (this.applicationContext != null) {
        try {
            Method eclMethod = configuration.getClass().getMethod("externalClassLoader", ClassLoader.class);
            ReflectionUtils.invokeMethod(eclMethod, configuration, this.applicationContext.getClassLoader());
        }
        catch (NoSuchMethodException ex) {
            // Ignore - no Hibernate Validator 5.2+ or similar provider
        }
    }
```

JSR Validation

OptionalValidatorFactoryBean

LocalValidatorFactoryBean subclass that simply turns org.springframework.validation.Validator calls into no-ops in case of no Bean Validation provider being available.

Spring Validator 5

Return a global Validator instance for example for validating @ModelAttribute and @RequestBody method arguments. Delegates to getValidator() first and if that returns null checks the classpath for the presence of a JSR-303 implementations before creating a OptionalValidatorFactoryBean.If a JSR-303 implementation is not available, a no-op Validator is returned.

Expert Java Leader.us

MVC默认全局的Validator

WebMvcConfigurationSupport

```
@Bean
public Validator mvcValidator() {
    Validator validator = getValidator();
    if (validator == null) {
        if (ClassUtils.isPresent("javax.validation.Validator", getClass().getClassLoader())) {
            Class<?> clazz;
            try {
                String className = "org.springframework.validation.beanvalidation.OptionalValidatorFactoryBean";
                clazz = ClassUtils.forName(className, WebMvcConfigurationSupport.class.getClassLoader());
            }
            catch (ClassNotFoundException ex) {
                throw new BeanInitializationException("Could not find default validator class", ex);
            }
            catch (LinkageError ex) {
                throw new BeanInitializationExcepti
            }
            validator = (Validator) BeanUtils.insta
        }
        else {
            validator = new NoOpValidator();
        }
    }
    return validator;
}
```

```
    return validator;
}

/**
 * Override
 */
protected V
    return
}
```

```
validator= OptionalValidatorFactoryBean (id=60)
    applicationContext= null
    constraintValidatorFactory= null
    mappingLocations= null
    messageInterpolator= null
    parameterNameDiscoverer= DefaultParameterNameDiscoverer (id=75)
    providerClass= null
org.springframework.validation.beanvalidation.OptionalValidatorFactory
```

To Be Continued