**中软国际考题-笔试**

1. 单选题
2. Spring的校验框架在（D）包中。

A．org.springframework.core.serializer

B. org.springframework.cache.jcache.config

C. org.springframework.cache.jcache

D. org.springframework.validation

2、下面通过正则表达式来验证数据的是（C）

A. @NumberFormat(pattern = "#,###.##")

B. @Length(min = 2, max = 100)

C. @Pattern(regexp = "^\\w{4,30}$", message="字符长度应该是4-30.")

D. @DecimalMax(value = "100000.00")

3、下面表示验证字符长度的是（B）

A. @NumberFormat(pattern = "#,###.##")

B. @Length(min = 2, max = 100)

C. @Pattern(regexp = "^\\w{4,30}$", message="字符长度应该是4-30.")

D. @DecimalMax(value = "100000.00")

4、下面表示验证数字格式的是（A）

A. @NumberFormat(pattern = "#,###.##")

B. @Length(min = 2, max = 100)

C. @Pattern(regexp = "^\\w{4,30}$", message="字符长度应该是4-30.")

D. @DecimalMax(value = "100000.00")

5、下列jar包那个不是spring校验需要的（D）

A、classmate-1.3.1.jar

B、hibernate-validator-5.4.1.Final.jar

C、hibernate-validator-annotation-processor-5.4.1.Final.jar

D、spring-tx.jar

6、数据校验工作，最有效的层是哪？（E）

1. UI层 B. 服务层 C. 数据访问层
2. 控制层 E. 实体层

7、 在javaEE7规范中，要使用的数据校验规范的版本是( B )

A. Bean Validation 1.0

B. Bean Validation 1.1

C. Bean Validation 1.2

D. Bean Validation 12.0

8、内置的验证约束注解，NotNull可以校验什么类型？（B）

A.Boolean B.任意类型 C.java.util.Date D.BigDecimal

8、为了创建一个Spring校验器，需要实现的接口为（ C ）

A. java.validation.Validator

B. javax.validation.Validator

C.org.springframework.validation.Validator

D. javax.validation.Validation

9、下列哪个可以实现数据校验是（A）

AValidator BData C.jdbc D.都不可能

10、 MD5加密是否可逆(A)

A.不可逆 B.可逆 C.明文

11、Spring中自带的数据校验，需实现以下哪个接口（ ）

A． Validator

B． Validation

C． Object

D． Compare

1. 多选
2. Spring校验框架的重要接口和类包括（ABCD）

A．Errors

B．ValidationUtils

C．LocalValidatorFactoryBean

D．Validator

2、下面是数据校验所需要的jar包文件有（A B）【选两项】

A. validation-api-1.0.0.GA.jar

B. hibernate-validator-4.3.0.Final.jar

C. jboss-validate-log4j-3.1.0.jar

D. spring-validation-core-3.1.2.jar

3、 下面哪些是sping 自带的校验注解？（A、B、C、D）

A. NotNull B. Size C. DecimalMin

D. Pattern

4、数据校验工作应该在哪个层做？（A、B、C、D）

A. UI层 B. 服务层 C. 数据访问层 D.控制层

5. 如下WELD注解中，哪个scope是有效的 （A、C、D）

A. @SessionScoped B. @RequestScoped

C. @ApplicationScoped D. @Singleton

6、下列注解那些是判断为空的？

A. @NotNull B. @NotEmpty C. @NotBlank D. @Null

7、Spring校验器接口包括下面哪些方法？（ B、C ）

A. init

B. supports

C. validate

D. doValidate

E. execute

8、下面常用的校验标准有（A、B）

A. JSR-303 B. JSR-349 C. JSR-380

D. 以上都不是

9、MD5的好处是（AB）

A.加密

B.不可逆

C.明文

10、 以下哪些为hibernate validator中的常见注解?（ ）

A． @Null

B． @Min(value)

C． @DecimalMin(value)

D． @Size(max=, min=)

1. 填空题

1、springMVC提供了强大的数据校验功能，其中有两种方法可以验证输入：一种是利用Spring自带的\_\_\_\_\_\_Validation\_\_\_\_\_校验框架，另一种是利用\_\_\_\_\_\_JSR 303\_\_\_\_\_(Java校验规范)实现校验功能。

1. Spring 3.0拥有自己独立的数据校验框架，同时支持JSR 303标准的校验框架。Spring的DataBinder在进行数据绑定时，可同时调用校验框架完成数据校验工作
2. spring自定义实现数据校验要实现的接口是 。(Validator)
3. 是**Java为Bean数据合法性校验**提供的一个标准框架。（JSR303）
4. 在java数据校验规范中，需要使用到CDI，这里的CDI是指\_\_\_\_\_\_\_\_\_\_ and\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_for java。(context、dependency、injection)
5. 把一个java bean注入到系统中，应该使用的注解是：\_\_\_\_\_\_\_\_\_\_\_（@Inject ）
6. 校验java bean对象是否有效，应该使用的注解是：\_\_\_\_\_\_\_\_\_\_\_（@Valid ）
7. 可以校验为空的注解（@NotNull），（@NotEmpty），（@NotBlank）
8. 校验邮箱的注解（@Email）
9. 被注解的元素必须在制定的范围(数据类型:String, Collection, Map and arrays)用的注解是（@Size(min=, max=)）
10. 被注解的元素必须是日期，检查给定的日期是否比现在晚的注解（@Future）
11. 被注释的元素必须在合适的范围内 (数据：BigDecimal, BigInteger, String, byte, short, int, long and 原始类型的包装类 )的注解是（@Range(min=, max=)）
12. 被注解的元素必须为null的注解是（@Null）
13. 被注解的对象必须是字符串的大小必须在制定的范围内的注解是（@Length(min=, max=)）
14. Spring自行开发的数据校验功能由3部分组成：校验器、\_\_\_\_\_\_\_\_\_ 、\_\_\_\_\_\_\_\_\_。(校验对象、校验结果)
15. 校验器是、\_\_\_\_\_\_\_\_\_\_\_\_\_(Validator)
16. 数据校验的核心是数据的 \_\_\_\_\_\_\_ (完整性)
17. Spring MVC 提供了两种方法来对用户的输入数据进行校验，一种是Spring 自带的 校验框架，另一种是利用 验证框架进行验证。 （Validation、JRS-303）
18. 判断题
19. Spring在进行数据绑定时，可同时调用校验框架来完成数据校验工作（T）
20. 由于Validation框架通过硬编码完成数据校验，在实际开发中会显得比较麻烦，因此现在更加推荐使用Java校验规范（T）
21. Spring本身提供了Java校验规范接口的实现（F）（暂时没有提供）
22. 使用<mvc:annotation-driven>启动验证框架。（ √ ）
23. 应用程序在执行业务逻辑前，数据校验保证接收到的输入数据是正确合法的。（ T ）
24. java bean对象的生命期由Spring容器管理（ F ）
25. 所有的java bean对象都是有状态的 （ F ）
26. 被注解的元素必须过去的日期，检查标注对象中的值表示的日期比当前早的注解是@Past(java.util.Date/Calendar)（对）
27. org.springframework.validation.ValidationUtils 是Spring数据校验的工具类。（ T ）
28. Spring核心部分已经提供Bean Validation相关的实现类，可以直接使用。（ F ）
29. 使用JSR Bean Validation时，需要我们自己编写校验器。（ F ）
30. 使用JSR Bean Validation时，可以利用JSR 303注解类型嵌入约束。（ T ）
31. 在某些项目可能会出现在Spring Validation的基础上新增Bean Validation功能，可以使用SpringValidatorAdapter适配器来解决。（ T ）
32. Validator实现不了数据校验（ F ）
33. 数据校验不需要保证数据的完整性（F ）
34. Spring中不支持服务器端验证。（ F ）
35. Spring中数据校验不支持注解方式。（ F ）

五、简答题

# 1、简单介绍一下SpringMVC中使用spring存放错误信息的接口?

答案：Errors是Spring用来存放错误信息的接口，SpringMVC框架在将请求数据绑定到入参对象后，就会调用校验框架实施校验，而校验结果保存在处理方法的入参对象之后的参数对象中。这个保存校验结果的参数对象必须是Errors或者BindingResult类型。一个Errors对象中包含了一系列的FieldError和ObjectError对象。FieldError表示与被校验的对象中的某个属性相关的一个错误。BindingResult扩展了Errors接口，同时可以获取数据绑定结果对象的信息

2、前端JavaScript验证和Spring数据校验的区别？

答案：其根本在于安全性方面。前端JavaScript验证代码在客户端执行，可被下载后篡改，导致客户端绕过验证。而Spring MVC属于服务器验证，不能被篡改，所以具有安全性。

3、描述数据校验的流程：

答案：

1）.校验

a.使用JSR303验证标准

b.加入 hibernate validate验证框架

c.在springmvc中配置annotation

d.在需要的bean的属性上添加对应的注解

e .在目标方法bean 类型的前面添加@Valid 注解

2）.验证出错转向页面

3）.<form:errors/>标签显示错误信息及国际化标准

4、简述在javaEE7的数据校验规范中，包含了哪些JSR文档？ 常用的数据校验的注解有哪些？（15分）

答案：

规范：**[Contexts and Dependency Injection for Java 1.1](http://jcp.org/en/jsr/detail?id=346" \t "_blank)**

[**Dependency Injection for Java 1.0**](http://jcp.org/en/jsr/detail?id=330)

[**Bean Validation 1.1**](http://jcp.org/en/jsr/detail?id=349)

常用注解：

@Inject @Valid @NotNull @NotBlank

@MySize @Email @Pattern @Past

5、使用Spring3开始支持JSR-303验证框架，实现导入的jar文件是什么？

答案：validation-api-1.0.0.GA.jar JSR-303规范API包  
hibernate-validator-4.3.0.Final.jar Hibernat

6、使用Spring校验器的@Validated和JSR-303规范的@Valid对数据进行验证的区别是什么？

答案：

两者在分组、注解地方、嵌套验证等功能方面有所不同：

（1）分组

@Validated：提供了一个分组功能，可以在入参验证时，根据不同的分组采用不同的验证机制，这个网上也有资料，不详述。

@Valid：作为标准JSR-303规范，还没有吸收分组的功能。

（2） 注解地方

@Validated：可以用在类型、方法和方法参数上。但是不能用在成员属性（字段）上

@Valid：可以用在方法、构造函数、方法参数和成员属性（字段）上。

（3）嵌套验证

@Validated：用在方法入参上无法单独提供嵌套验证功能。不能用在成员属性（字段）上，也无法提示框架进行嵌套验证。能配合嵌套验证注解@Valid进行嵌套验证。

@Valid：用在方法入参上无法单独提供嵌套验证功能。能够用在成员属性（字段）上，提示验证框架进行嵌套验证。能配合嵌套验证注解@Valid进行嵌套验证。

7、 为什么要使用Spring的数据校验框架？

答案：

Spring为我们提供了基于注解的数据检验框架，框架会将所有的错误提示保存在集合里（BindingResult），我们可以将这个集合将里面的错误信息取出来提示给前台。使得我们不需要再自己去写一堆的数据合法性检验的代码，让数据检验层代码变得优雅。

8、 对于Spring提供的校验器和JSR Bean Validataion，我们在项目中应该如何选择？

答案：

JSR Bean Validataion是JSR-303的规范，建议在项目中优先选择使用。

9、 JSR Bean Validataion目前有哪些实现？

答案：

JSR Bean Validataion目前有2个实现。

第1个是Hibernate Validator，JSR-303和JSR-349它都实现了。

第2个是Apache BVal。

10、 org.springframework.validation.ValidationUtils提供了哪些功能？

答案：

（1）执行数据校验；

（2）提供错误信息绑定的功能，例如ValidationUtils.rejectIfEmpty会将对应的信息写入到Errors中。

11、spring如何实现数据校验

答案：实现Validator接口，实现校验的业务逻辑

注入校验的功能

12、校验是什么

校验,是为保护数据的完整性，用一种指定的算法对原始数据计算出的一个校验值。当接收方用同样的算法再算一次校验值，如果两次校验值一样，表示数据完整。

13. 最简单的校验

实现方法：将原始数据和待比较的数据直接进行比较，看是否完全一样。

特点：最安全准确效率最低

适用范围：简单的数据量极小的通讯

14. 奇偶校验

实现方法：在数据存储和传输中，字节中额外增加一个比特位，用来传送奇/偶校验位来检验错误。

15、CRC码集选择的原则公式:

答： 若设码字长度为N，信息字段为K位，校验字段为R位(N=K+R)，则对于CRC码集中的任一码字，存在且仅存在一个R次多项式g(x)，使得

V(x)=A(x)g(x)=xRm(x)+r(x);

其中: m(x)为K次信息多项式， r(x)为R-1次校验多项式，

g(x)称为生成多项式：

g(x)=g0+g1x+g2x2+...+g(R-1)x(R-1)+gRxR

发送方通过指定的g(x)产生CRC码字，接收方则通过该g(x)来验证收到的CRC

15、什么是数据校验？

答案：

用来验证客户输入的数据是否合法，比如客户登录时，用户名不能为空，或者不能超出指定长度等要求，这就叫做数据校验。

数据校验分为客户端校验和服务端校验

16、简述Spring Validation数据校验由哪些部分构成？

1) 校验器——Validator，他会运行校验代码。

2) 校验对象，实际上就是一个JavaBean，Validator会对其进行校验。

3) 校验结果——Errors，一次校验的结果都存放在Errors实例中。

17 简述JSR303/JSR-349，hibernate validation，spring validation之间的关系

JSR303是一项标准,JSR-349是其的升级版本，添加了一些新特性，他们规定一些校验规范即校验注解，如@Null，@NotNull，@Pattern，他们位于javax.validation.constraints包下，只提供规范不提供实现。

hibernate validation是对这个规范的实践，他提供了相应的实现，并增加了一些其他校验注解，如@Email，@Length，@Range等等，他们位于org.hibernate.validator.constraints包下。而spring为了给开发者提供便捷，对hibernate validation进行了二次封装，显示校验validated bean时，你可以使用spring validation或者hibernate validation，而spring validation另一个特性，便是其在springmvc模块中添加了自动校验，并将校验信息封装进了特定的类中。

18 列出hibernate validator中五个常见注解，并说明作用

@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

六、程序实现题

1、前端有请求参数登录名密码：loginname,password；现有User类，属性为私有的userName,password，提供set/get方法；使用Spring的Validation校验，编写一个方法对目标targer进行校验，并将校验错误记录在errors当中；校验登录名和登录密码不能为空,用户名不能超过20个自己，密码不能少于6个字节

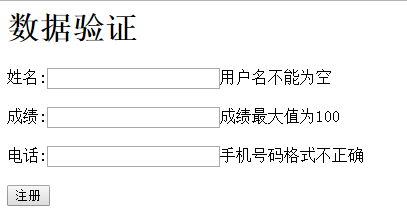
|  |
| --- |
| package org.study.validation.validator; import org.springframework.validation.Errors; import org.springframework.validation.ValidationUtils; import org.springframework.validation.Validator; import org.study.domain.User; public class UserValidator implements Validator { @Override public boolean supports(Class<?> clazz) { return clazz.equals(User.class); } @Override public void validate(Object target, Errors errors) { ValidationUtils.rejectIfEmpty(errors, "userName", "user.userName.required", "用户名不能为空"); ValidationUtils.rejectIfEmpty(errors, "password", "user.password.required", "密码不能为空");  User user = (User)target; int length = user.getUserName().length(); if(length>20){ errors.rejectValue("userName", "user.userName.too\_long", "用户名不能超过{20}个字符"); } length = user.getPassword().length(); if(length <6){ errors.rejectValue("password", "user.password.too\_short", "密码太短，不能少于{6}个字符"); }  } } |

2、使用数据校验的第一步是在xml编写什么代码来启用数据校验？

|  |
| --- |
| xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.1.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.1.xsd*  *http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.1.xsd">*  <beans >  <mvc:annotation-driven/>  <context:component-scan base-package="com.pxy.mvc" />  <bean id="validator" class="org.springframework.validation.beanvalidation.LocalValidatorFactoryBean">  </bean>  </beans> |

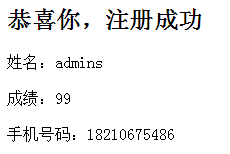
3、使用spring 数据校验实现如图所示：

图一若不填写数据：



若输入合法：





|  |
| --- |
| 1、applicationContext.xml中配置验证器：  https://images2015.cnblogs.com/blog/903715/201701/903715-20170108122712487-550199468.png  2、定义实体类，打注解标记  public class UserInfo {      @NotEmpty(message="用户名不能为空")      @Size(min=3,max=6,message="姓名长度应在{min}-{max}")      private String username;        @NotNull(message="成绩最大值为100")      @Min(value=0,message="成绩不能小于{value}")      @Max(value=100,message="成绩不能大于{value}")      private Integer score;        @NotEmpty(message="手机号码不允许为空")      @Pattern(regexp="^1[34578]\\d{9}$",message="手机号码格式不正确")      private String phone;  }  3、书写Controller方法  https://images2015.cnblogs.com/blog/903715/201701/903715-20170108123335675-2129712366.png  4、搭建jsp页面  <form action="${pageContext.request.contextPath }/first.do" method="post">  <h1>数据验证</h1>  姓名:<input name="username"/>${namemsg }<br/><br/>  成绩:<input name="score" />${scoremsg}<br/><br/>  电话:<input name="phone" />${phonemsg }<br/><br/>  <input type="submit" value="注册"/>  </form> |

3、做登陆页面，完成数据校验。

（10分）

|  |
| --- |
| **在Spring配置总添加对JSR-303验证框架的支持**  <!-- 以下 validator ConversionService 在使用 mvc:annotation-driven 会 自动注册-->  <bean id="validator"  class="org.springframework.validation.beanvalidation.LocalValidatorFactoryBean">  <property name="providerClass" value="org.hibernate.validator.HibernateValidator"/>  <!-- 如果不加默认到 使用classpath下的 ValidationMessages.properties -->  <property name="validationMessageSource" ref="messageSource"/>  </bean>  **使用JSR-303验证框架注解为模型对象指定验证信息**  **import javax.validation.constraints.NotNull;**  **public class UserModel {**  **@NotNull(message="{username.not.empty}")**  **private String username;**  **}**  **验证失败后需要展示的页面（/WEB-INF/jsp/validate/error.jsp）**  <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <%@taglib prefix="form" uri="http://www.springframework.org/tags/form" %>  <form:form commandName="user">  <form:errors path="\*" cssStyle="color:red"></form:errors><br/>  </form:form>  **控制器：**  //省略import  @Controller  public class HelloWorldController {  @RequestMapping("/validate/hello")  public String validate(@Valid @ModelAttribute("user") UserModel user, Errors errors) {  if(errors.hasErrors()) {  return "validate/error";  }  return "redirect:/success";  }  } |

2、@NotNull @NotEmpty @NotBlank 3个注解的区别：

@NotNull 任何对象的value不能为null   
@NotEmpty 集合对象的元素不为0，即集合不为空，也可以用于字符串不为null   
@NotBlank 只能用于字符串不为null，并且字符串trim()以后length要大于0

3、校验邮箱的注解？

@Email

4、校验密码长度为6-20之间

@Length（min=6，max=20,message="密码长度6-20之间"）

5、@Valid含义是什么？

被注释的元素是一个对象，需要检查所有字段值

4、使用Spring校验器对产品的名称（非空）和生产日期（不能早于当前时间）进行校验。

|  |
| --- |
| 1. 实体定义  public class Product implements Serializable  {  private static final long serialVersionUID = -164834188493691288L;  private String name;  private Date productionDate;  public String getName()  {  return name;  }  public void setName(String name)  {  this.name = name;  }  public Date getProductionDate()  {  return productionDate;  }  public void setProductionDate(Date productionDate)  {  this.productionDate = productionDate;  }  }  2. 定义校验器  public class ProductValidator implements Validator  {  @Override  public boolean supports(Class<?> clazz)  {  return Product.class.isAssignableFrom(clazz);  }  @Override  public void validate(Object target, Errors errors)  {  Product product = (Product)target;  ValidationUtils.rejectIfEmpty(errors, "name", "productname.required");  Date productionDate = product.getProductionDate();  if (productionDate != null)  {  if (productionDate.after(new Date()))  {  errors.rejectValue("productionDate", "productiondate.invalid");  }  }  }  }  3. 使用Spring校验器  @Controller  public class ProductController  {  private static final Log logger = LogFactory.getLog(ProductController.class);  @org.springframework.web.bind.annotation.InitBinder  public void InitBinder(WebDataBinder binder)  {  binder.setValidator(new ProductValidator());  binder.validate();  }  @RequestMapping(value = "/product\_save")  public String saveProduct(@ModelAttribute Product product, BindingResult bindingResult, Model model)  {  // ProductValidator productValidator = new ProductValidator();  // productValidator.validate(product, bindingResult);  if (bindingResult.hasErrors())  {  FieldError fieldError = bindingResult.getFieldError();  logger.info("Code:" + fieldError.getCode() + ",field:" + fieldError.getField());  return "ProductForm";  }  // save product here  model.addAttribute("product", product);  return "ProductDetails";  }  } |

5、使用JSR Bean Validation对产品的名称（非空）和生产日期（不能早于当前时间）进行校验。

1. 定义实体类

public class Product implements Serializable

{

private static final long serialVersionUID = -164834188493691288L;

@Size(min = 1, max = 10)

private String name;

@Past

private Date productionDate;

public String getName()

{

return name;

}

public void setName(String name)

{

this.name = name;

}

public Date getProductionDate()

{

return productionDate;

}

public void setProductionDate(Date productionDate)

{

this.productionDate = productionDate;

}

}

2. 数据校验

@Controller

public class ProductController

{

private static final Log logger = LogFactory.getLog(ProductController.class);

@RequestMapping(value = "/product\_save")

public String saveProduct(@Valid @ModelAttribute Product product, BindingResult bindingResult,

Model model)

{

if (bindingResult.hasErrors())

{

FieldError fieldError = bindingResult.getFieldError();

logger.info("Code:" + fieldError.getCode() + ",field:" + fieldError.getField());

return "ProductForm";

}

// save product here

model.addAttribute("product", product);

return "ProductDetails";

}

}

6、实现一段能够校验的功能代码

（10分）

|  |
| --- |
| public class VehicleValidator implements Validator {  private List<String> \_TYPE = Arrays.asList(new String[] { "CAR", "SUV", "MPV" });  public boolean supports(Class<?> clazz) {  //将验证器和实体类进行绑定，如果这里返回false在验证过程中会抛出类型不匹配的异常  return Vehicle.class.isAssignableFrom(clazz);  }  public void validate(Object target, Errors errors) { //验证数据  Vehicle vehicle = Vehicle.class.cast(target);  if (null == vehicle.getName()) {  //使用验证工具绑定结果  ValidationUtils.rejectIfEmpty(errors, "name", "name.empty", "车辆名称为空");  }  if (!\_TYPE.contains(vehicle.getType())) {  //向Error添加验证错误信息  <2> errors.rejectValue("type", "type.error", "汽车类型必须是" + \_TYPE);  }  //More validate ......  }  } |

7、使用Annotaion JSR-303标准的验证方式验证User类中的各个属性，username不能为空、password不能为空，并且长度必须在6~10位之间，age必须在18到60岁之间，email格式必须合法。只需列出User类中配置即可。

|  |
| --- |
| import javax.validation.constraints.Pattern;  import org.hibernate.validator.constraints.Email;  import org.hibernate.validator.constraints.Length;  import org.hibernate.validator.constraints.NotBlank;  import org.hibernate.validator.constraints.Range;  public class User {  @NotBlank(message = "用户名不能为空")  private String username;  @NotBlank(message = "密码不能为空")  @Length(min = 6, max = 10, message = "密码的长度必须在6~10位之间")  private String password;  @Range(min = 18, max = 60, message = "年龄必须在18岁到60岁之间")  private Integer age;  @Email(message = "请输入合法的邮箱地址")  private String email;  public String getUsername() {  return username;  }  public void setUsername(String username) {  this.username = username;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  public Integer getAge() {  return age;  }  public void setAge(Integer age) {  this.age = age;  }  public String getEmail() {  return email;  }  public void setEmail(String email) {  this.email = email;  }  } |