开发用户信息修改和删除服务

常用的验证注解

Hibernate Validator

@NotNull	值不能为空
@Null	值必须为空
@Pattern(regex=)	字符串必须匹配正则表达式
@Size(min=, max=)	集合的元素数量必须在min和max之间
@CreditCardNumber(ignoreNonDigitCharacters=)	字符串必须是信用卡号(按美国的标准验的 -_-!)
@Email	字符串必须是Email地址
@Length(min=,max=)	检查字符串的长度
@NotBlank	字符串必须有字符
@NotEmpty	字符串不为null，集合有元素
@Range(min=,max=)	数字必须大于等于min,小于等于max
@SafeHtml	字符串是安全的html
@URL	字符串是合法的URL

Hibernate Validator

注解	描述
@AssertFalse	值必须是false
@AssertTrue	值必须是true
@DecimalMax(value=,inclusive=)	值必须 小于等于(inclusive=true)/小于(inclusive=false) value 属性指定的值。可以注解在字符串类型的属性上。
@DecimalMin(value=,inclusive=)	值必须 大于等于(inclusive=true)/大于(inclusive=false) value 属性指定的值。可以注解在字符串类型的属性上。
@Digits(integer=,fraction=)	数字格式检查。integer指定整数部分的最大长度，fraction指定小数部分的最大长度
@Future	值必须是未来的日期。
@Past	值必须是过去的日期。
@Max(value=)	值必须小于等于value指定的值。不能注解在字符串类型的属性上。
@Min(value=)	值必须大于等于value指定的值。不能注解在字符串类型的属性上。

```
1 package com.xiaoming.dto;
2
3 import java.util.Date;
4
5 import javax.validation.constraints.Past;
6
7 import org.hibernate.validator.constraints.NotBlank;
8
```

```java
import com.fasterxml.jackson.annotation.JsonView;

/**
 * @author 江小明
 *
 */
public class User {

 public interface UserSimpleView {};
 public interface UserDetailView extends UserSimpleView{};

 private String username;
 @NotBlank(message = "密码不能为空")
 private String password;
 private String id;

 @Past(message = "生日必须为过去的时间")
 private Date birth;

 @JsonView(UserSimpleView.class)
 public Date getBirth() {
 return birth;
 }
 public void setBirth(Date birth) {
 this.birth = birth;
 }
 @JsonView(UserSimpleView.class)
 public String getId() {
 return id;
 }
 public void setId(String id) {
 this.id = id;
 }
 @JsonView(UserSimpleView.class)
 public String getUsername() {
 return username;
 }
 public void setUsername(String username) {
 this.username = username;
 }
```

```
 @JsonView(UserDetailView.class)
 public String getPassword() {
 return password;
 }
 public void setPassword(String password) {
 this.password = password;
 }


}
```

```
@PutMapping("/{id:\\d+}")
 public User update(@Valid @RequestBody User user, BindingResult errors)
{
 //把错误的信息和对应的字段都打印出来
 if(errors.hasErrors()) {
 errors.getAllErrors().stream().forEach(error -> {
 /*
 * FieldError fieldError = (FieldError)error; String message =
 * fieldError.getField() +" "+ error.getDefaultMessage();
 * System.out.println(message);
 */
 System.out.println(error.getDefaultMessage());
 });
 }

 System.out.println(user.getUsername());
 System.out.println(user.getPassword());
 System.out.println(user.getId());
 System.out.println(user.getBirth());

 user.setId("1");
 return user;
 }
```

自定义消息

```
@MyConstraint(message = "这是一个测试")
 private String username;
 @NotBlank(message = "密码不能为空")
 private String password;
 private String id;
```

```
6
7   @Past(message = "生日必须为过去的时间")
8   private Date birth;
```

自定义校验注解

```
1  /**
2   * 自定义注解
3   * @author 江小明
4   *
5   */
6  @Target({ElementType.METHOD, ElementType.FIELD}) //表示这个注解可以标记在哪个元素上面
7  @Retention(RetentionPolicy.RUNTIME) //运行时注解
8  @Constraint(validatedBy = MyConstraintValidator.class) //当前注解使用某个类去注解
9  public @interface MyConstraint {
10  //下面三个属性都是必须
11  String message(); //校验错误信息
12
13  Class<?>[] groups() default { };
14
15  Class<? extends Payload>[] payload() default { };
16 }
```

```
1  /**
2   *
3   * @author 江小明
4   *<MyConstraint, Object> MyConstraint要验证的注解 Object标注的类型
5   */
6  public class MyConstraintValidator implements ConstraintValidator<MyConstraint, Object> {
7
8   //可以注入服务
9   @Autowired
10   private HelloService helloService;
11
12   //初始化
13   @Override
14   public void initialize(MyConstraint constraintAnnotation) {
15   System.out.println("my validator init");
16   }
17
```

```
18   //验证 进行业务逻辑处理
19   @Override
20   public boolean isValid(Object value, ConstraintValidatorContext
context) {
21   helloService.greeting("xiaoming");
22   System.out.println(value);
23   //true 代表校验成功
24   return false;
25   }
26
27  }
```