后端参数校验

简介：

`spring-boot-starter-validation` 是 Spring Boot 提供的一个启动器模块，它主要用于简化在 Spring Boot 应用程序中的验证功能。开发者可以在模型类字段上使用注解来定义验证规则。

使用

常用注解说明

- @NotNull：值不能为null；
- @NotEmpty：字符串、集合或数组的值不能为空，即长度大于0；
- @NotBlank：字符串的值不能为空白，即不能只包含空格；
- @Size：字符串、集合或数组的大小是否在指定范围内；
- @Min：数值的最小值；
- @Max：数值的最大值；
- @Pattern：字符串是否匹配指定的正则表达式；
- @Email：字符串是否为有效的电子邮件地址；
- @Future：日期是否为将来的日期；
- @Past：日期是否为过去的日期；

引入依赖：

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
```

@Validated：用于标记，某个类将触发验证

```
@GetMapping("/validation")
public String validation(@Validated ValidationDTO validationDTO) {
    return "参数校验测试";
```

```
}
```

参数类修改

```
import jakarta.validation.constraints.Email;
import jakarta.validation.constraints.Future;
import jakarta.validation.constraints.Max;
import jakarta.validation.constraints.Min;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.Past;
import jakarta.validation.constraints.Pattern;
import jakarta.validation.constraints.Size;
import lombok.Getter;
import lombok.Setter;

import java.time.LocalDate;

@Getter
@Setter
public class ValidationDTO {

    @NotBlank(message = "用户账号不能为空")
    private String userAccount;

    @NotBlank(message = "用户密码不能为空")
    @Size(min = 5, max = 10, message = "密码长度不能少于6位，不能大于10位")
    private String password;

    @Min(value = 0, message = "年龄不能小于0岁")
    @Max(value = 60, message = "年龄不能大于60岁")
    private int age;

    @Email(message = "必须符合邮箱格式")
    private String email;

    @Pattern(regexp = "^(13[0-9]|14[01456879]|15[0-35-9]|16[2567]|17[0-8]|18[0-9]|19[0-35-9])\\d{8}$", message = "手机号码格式不正确")
    private String phone;

    @Past(message = "开始日期必须是过去的日期")
    private LocalDate startDate;

    @Future(message = "结束日期必须是未来的日期")
    private LocalDate endDate;
}
```

增加参数异常捕获

```
 @ExceptionHandler(BindException.class)
public R<Void> handleBindException(BindException e) {
    log.error(e.getMessage());
    String message = join(e.getAllErrors(),
DefaultMessageSourceResolvable::getDefaultMessage, ", ");
    return R.fail(ResultCode.FAILED_PARAMS_VALIDATE.getCode(), message);
}

private <E> String join(Collection<E> collection, Function<E, String>
function, CharSequence delimiter) {
    if (CollUtil.isEmpty(collection)) {
        return StrUtil.EMPTY;
    }
    return
collection.stream().map(function).filter(Objects::nonNull).collect(Collectors.j
oining(delimiter));
}
```