异常处理

@Target({ElementType.METHOD, ElementType.FIELD})

@Retention(RetentionPolicy.RUNTIME)

@Constraint(validatedBy = **MyConstraintValidator**.class)

public @interface **MyConstraint** {

String message();

Class<?>[] groups() default { };

Class<? extends Payload>[] payload() default { };

}

public class **MyConstraintValidator** implements **ConstraintValidator**<**MyConstraint**, Object> {

@Autowired

private HelloService helloService;

@Override

public void initialize(MyConstraint constraintAnnotation) {

System.out.println("my validator init");

}

@Override

public boolean isValid(Object value, ConstraintValidatorContext context) {

helloService.greeting("tom");

System.out.println(value);

return false;

}

}

public class User {

public interface **UserSimpleView** {};

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

private String id;

@**MyConstraint**(message = "这是一个测试")

private String username;

@NotBlank(message = "密码不能为空")

private String password;

@Past(message = "生日必须是过去的时间")

private Date birthday;

@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;

}

@JsonView(UserSimpleView.class)

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

@JsonView(UserSimpleView.class)

public Date getBirthday() {

return birthday;

}

public void setBirthday(Date birthday) {

this.birthday = birthday;

}

}

public class **UserNotExistException** extends **RuntimeException** {

private static final long serialVersionUID = -6112780192479692859L;

private String id;

public UserNotExistException(String id) {

super("user not exist");

this.id = id;

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

}

@**ControllerAdvice**

public class ControllerExceptionHandler {

@ExceptionHandler(UserNotExistException.class)

@ResponseBody

@ResponseStatus(HttpStatus.INTERNAL\_SERVER\_ERROR)

public Map<String, Object> handleUserNotExistException(UserNotExistException ex) {

Map<String, Object> result = new HashMap<>();

result.put("id", ex.getId());

result.put("message", ex.getMessage());

return result;

}

}

@PutMapping("/{id:\\d+}")

public User update(**@Valid** @RequestBody User user, BindingResult errors去掉这个就进不了方法了) {

if (errors.hasErrors()) {

errors.getAllErrors().stream().forEach(error -> {

// FieldError fieldError = (FieldError)error;

// String message = fieldError.getField() +" "+

// error.getDefaultMessage();

System.out.println(error.getDefaultMessage());

});

}

System.out.println(user.getId());

System.out.println(user.getUsername());

System.out.println(user.getPassword());

System.out.println(user.getBirthday());

user.setId("1");

return user;

}

@GetMapping("/{id:\\d+}")

@JsonView(User.UserDetailView.class)

public User getInfo(@PathVariable String id) {

throw new UserNotExistException(id);

}