Contents

1.	Validation Annotation	2
	Cấu hình file pom.xml	
	Cấu hình file application.properties	
	Entity	
5.	Repository	6
6.	Customized Exception	6
а	. RecordNotFoundException	6
b	. InvalidUseridException	6
С	. CustomExeptionHandler	6
7.	ValidatingService	7
8.	Controller	7
9.	Demo	8

Link github: https://github.com/ngatdo3003/validation-exceptionHandler-example

1. Validation Annotation

ANNOTATION	USAGE
@AssertFalse	The annotated element must be false.
@AssertTrue	The annotated element must be true.
@DecimalMax	The annotated element must be a number whose value must be lower or equal to the specified maximum.
@DecimalMin	The annotated element must be a number whose value must be higher or equal to the specified minimum.
@Future	The annotated element must be an instant, date or time in the future.
@Max	The annotated element must be a number whose value must be lower or equal to the specified maximum.
@Min	The annotated element must be a number whose value must be higher or equal to the specified minimum.
@Negative	The annotated element must be a strictly negative number.
@NotBlank	The annotated element must not be null and must contain at least one non-whitespace character.
@NotEmpty	The annotated element must not be null nor empty.
@NotNull	The annotated element must not be null.
@Null	The annotated element must be null.
@Pattern	The annotated CharSequence must match the specified regular expression.
@Positive	The annotated element must be a strictly positive number.
@Size	The annotated element size must be between the specified boundaries (included).

<dependencies>

```
2. Cấu hình file pom.xml
cproperties>
  <java.version>1.8</java.version>
  <lombok.version>1.18.12</lombok.version>
</properties>
<dependencies>
  <dependency>
     <groupId>org.springframework.boot
     <artifactId>spring-boot-starter</artifactId>
  </dependency>
  <dependency>
     <groupId>org.springframework.boot
     <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
     <groupId>org.springframework.boot
     <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
     <groupId>mysql</groupId>
     <artifactId>mysql-connector-java</artifactId>
     <scope>runtime</scope>
  </dependency>
  <dependency>
     <groupId>org.springframework.boot
     <artifactId>spring-boot-devtools</artifactId>
     <scope>runtime</scope>
     <optional>true
  </dependency>
  <!-- https://mvnrepository.com/artifact/javax.validation/validation-api -->
  <dependency>
     <groupId>javax.validation
     <artifactId>validation-api</artifactId>
     <version>2.0.1.Final
  </dependency>
  <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-
databind -->
  <dependency>
     <groupId>com.fasterxml.jackson.core
     <artifactId>jackson-databind</artifactId>
     <version>2.10.2
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
  <dependency>
     <groupId>org.projectlombok</groupId>
     <artifactId>lombok</artifactId>
     <version>1.18.12
     <scope>provided</scope>
  </dependency>
</dependencies>
<dependencyManagement>
```

```
<dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-dependencies</artifactId>
        <version>${spring-cloud.version}</version>
        <type>pom</type>
        <scope>import</scope>
     </dependency>
  </dependencies>
</dependencyManagement>
<build>
  <plugins>
     <plugin>
        <groupId>org.apache.maven.plugins
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1
        <configuration>
          <!-- put your configurations here -->
          <source>1.8</source>
          <target>1.8</target>
          <annotationProcessorPaths>
             <path>
                <groupId>org.projectlombok</groupId>
                <artifactId>lombok</artifactId>
                <version>${lombok.version}
             </path>
          </annotationProcessorPaths>
        </configuration>
     </plugin>
  </plugins>
</build>
        3. Cấu hình file application.properties
# DATABASE
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/test2
spring.datasource.username=root
spring.datasource.password=123456
# JPA / HIBERNATE
#spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
```

4. Entity

Tạo class UserEnt cùng với các annotation của JPA và Validation

```
@Entity
@Data // lombok annotation for auto-generate getter/setter/constructor
@Table(name = "users", schema = "test2", catalog = "")
public class UserEnt implements Serializable {
    @Id
    @Column(name = "id", updatable = false, nullable = false)
    @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
    @Basic
    @Column (name = "userid")
    @NotBlank
   private String userid;
    @Basic
    @Column (name = "password")
    @Size(min = 6, max = 20)
   private String password;
    @Basic
    @Column(name = "email")
    @Email (message = "email should be a valid email")
   private String email;
    @Basic
    @Column (name = "phone")
    private String phone;
    @Basic
    @Column (name = "firstName")
    @NotBlank(message = "first name must not be blank")
   private String firstName;
    @Basic
    @Column(name = "lastName")
    @NotBlank(message = "last name must not be blank")
   private String lastName;
    @Basic
    @Column(name = "expired")
   private int expired;
    @Basic
    @Column (name = "disabled")
   private int disabled;
    @Basic
    @Column (name = "age")
    @Min(13)
    @Max (99)
   private int age;
}
```

5. Repository

```
Kế thừa lớp JpaRepository gồm các thao tác với csdl cơ bản:
```

```
public interface UserRepository extends JpaRepository<UserEnt, Long> {
}
```

6. Customized Exception

a. RecordNotFoundException

```
@ResponseStatus(HttpStatus.NOT_FOUND)
public class RecordNotFoundException extends RuntimeException {
    public RecordNotFoundException(String exception) {
        super(exception);
    }
}
```

b. InvalidUseridException

```
@ResponseStatus(HttpStatus.NOT_ACCEPTABLE)
public class InvalidUserIDException extends RuntimeException {
    public InvalidUserIDException(String exception) {
        super(exception);
    }
}
```

c. CustomExeptionHandler

```
@SuppressWarnings({"unchecked", "rawtypes"})
@ControllerAdvice
public class CustomExceptionHandler extends ResponseEntityExceptionHandler
    @ExceptionHandler(Exception.class)
    public final ResponseEntity<Object> handleAllExceptions(Exception ex,
WebRequest request) {
        List<String> details = new ArrayList<>();
        details.add(ex.getLocalizedMessage());
        ErrorResponse error = new ErrorResponse("Server Error", details);
        return new ResponseEntity(error, HttpStatus. INTERNAL SERVER ERROR);
    }
    @ExceptionHandler(RecordNotFoundException.class)
   public final ResponseEntity<Object>
handleUserNotFoundException (RecordNotFoundException ex, WebRequest request) {
        List<String> details = new ArrayList<>();
        details.add(ex.getLocalizedMessage());
        ErrorResponse error = new ErrorResponse("Record Not Found", details);
        return new ResponseEntity(error, HttpStatus.NOT FOUND);
    }
    @ExceptionHandler(InvalidUserIDException.class)
    public final ResponseEntity<Object>
handleInvalidUserIDException(InvalidUserIDException ex, WebRequest request) {
        List<String> details = new ArrayList<>();
        details.add(ex.getLocalizedMessage());
```

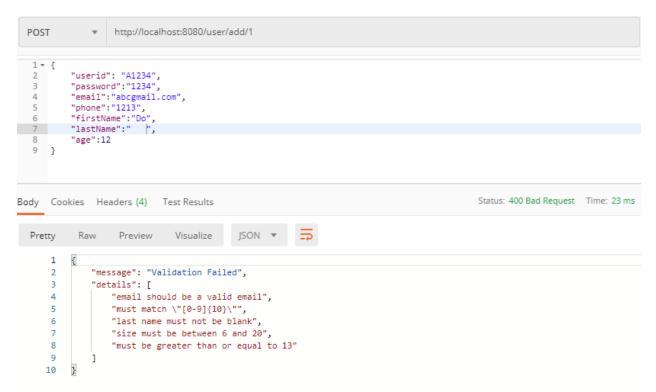
```
ErrorResponse error = new ErrorResponse("Invalid userID ", details);
        return new ResponseEntity(error, HttpStatus.NOT ACCEPTABLE);
    }
    @Override
    protected ResponseEntity<Object>
handleMethodArgumentNotValid(MethodArgumentNotValidException ex, HttpHeaders
headers, HttpStatus status, WebRequest request) {
        List<String> details = new ArrayList<>();
        for(ObjectError error : ex.getBindingResult().getAllErrors()) {
            details.add(error.getDefaultMessage());
        ErrorResponse error = new ErrorResponse("Validation Failed",
details);
        return new ResponseEntity(error, HttpStatus.BAD REQUEST);
}
         7. ValidatingService
 * Service validate dữ liệu dựa trên annotation @NotNull @NotBlank @Size @Min
@Max...
 */
@Service("ValidatingService")
@Validated
public class ValidatingService {
     * Hàm validate theo các Annotation
     * @param input
    public void validate(@Valid Object input) {
    /**
    * Hàm validate theo Annotation và thêm điều kiện là userID phải bắt đầu
bằng ABC
     * @param user
     * @return
    public void customizedValidate(@Valid UserEnt user) throws
InvalidUserIDException {
        if(!(user.getUserid()).startsWith("ABC")) throw new
InvalidUserIDException("UserID: " + user.getUserid() + " is invalid");
    }
}
         8. Controller
@RestController
public class UserController {
    UserRepository userRepository;
    @Autowired
```

ValidatingService validatingService;

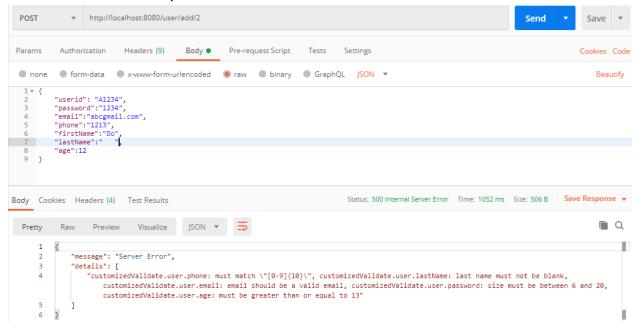
```
/**
     * Service thêm mới user
     * validate dữ liệu (chỉ validate dựa trên các annotation trong Entity)
ngay từ khi convert từ JSON sang Ent
     * @param user
     * @return
    @PostMapping(value = "/user/add/1")
   public ResponseEntity<UserEnt> addUser1(@RequestBody @Valid UserEnt user)
       user.setExpired(1);
       user.setDisabled(0);
       userRepository.save(user);
        return new ResponseEntity<UserEnt>(user, HttpStatus.OK);
    }
    /**
     * Service thêm mới user
     * Không validate dữ liệu khi convert từ JSON sang Ent
    * Validate dữ liệu sau đó dùng validatingService(validate dựa trên
annotation trong Entity và customized validation
     * @param user
     * @return
     */
    @PostMapping(value = "/user/add/2")
   public ResponseEntity<UserEnt> addUser2(@RequestBody UserEnt user)
throws InvalidUserIDException {
        user.setExpired(1);
        user.setDisabled(0);
        validatingService.customizedValidate(user);
        userRepository.save(user);
        return new ResponseEntity<UserEnt>(user, HttpStatus.OK);
    }
    /**
     * Service lấy thông tin User theo id
     * @param id
     * @return
    @GetMapping(value = "/user/{id}")
    public ResponseEntity<UserEnt> findById (@PathVariable("id") Long id)
        Optional < UserEnt > user = userRepository.findById(id);
        if(!user.isPresent()) {
            throw new RecordNotFoundException("Invalid user id : " + id);
        return new ResponseEntity<UserEnt>(user.get(), HttpStatus.OK);
    }
}
```

9. Demo

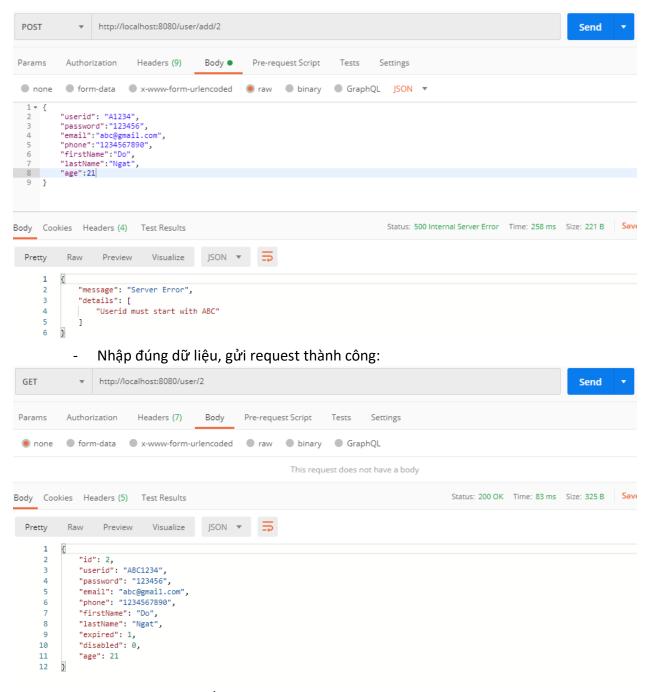
- Gọi đến web service add1 với các field không hợp lệ, ta được kết quả:



 Gọi đến web service add2 với các filed không hợp lệ, ta sẽ được báo lỗi của các filed được validate theo annotation trước



- Sau khi nhâp dữ liệu hợp lệ, chỉ trừ field userid, ta sẽ thấy thông báo lỗi:



Gọi web service lấy thông tin một user không có trong DB

