**Giới thiệu**

Bản thân Hibernate và Java đã cung cấp cho chúng ta rất nhiều các Annotation để validate dữ liệu của model.

Chẳng hạn như: @@NotBlank, @Size, @Email, v.v..

Tuy nhiên, trên thực tế, chúng ta có rất nhiều các điều kiện được đặt ra, tuỳ thuộc vào business và mô hình dự án.

Ví dụ như trong dự án của tôi, tôi muốn tất cả User đều phải có một thuộc tính là lodaId.

Một lodaId hợp lệ là chuỗi String có tiền tố: loda://xxxx

Lúc này, làm sao để tôi chắc chắn được rằng mọi User trước khi tạo đều phải có lodaId hợp lệ?

Rõ ràng tôi phải tự tạo ra một bộ kiểm tra cho riêng mình để kiểm soát tính hợp lệ của User.

Rất may, hibernate-validator sẽ giúp tôi làm điều đó.

Trong bài có sử dụng các kiến thức:

1. [Spring Boot](https://loda.me/spring-boot-0-series-lam-chu-spring-boot-zero-to-hero-loda1558963914472)
2. [jpa](https://loda.me/spring-boot-11-huong-dan-spring-boot-jpa-my-sql-loda1558687596060)
3. [lombok](https://loda.me/general-huong-dan-su-dung-lombok-giup-code-java-nhanh-hon-69-loda1552789752787)

**Cài đặt**

*pom.xml*

<?xml version="1.0" encoding="UTF-8"?>

<projectxmlns="http://maven.apache.org/POM/4.0.0"xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"xsi:schemaLocation="http://maven.apache.org/POM/4.0.0https://maven.apache.org/xsd/maven-4.0.0.xsd"><modelVersion>4.0.0</modelVersion><parent><groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-parent</artifactId><version>2.0.5.RELEASE</version><relativePath/> <!-- lookup parent from repository -->

</parent><groupId>me.loda.spring</groupId><artifactId>example-independent-maven-spring-project</artifactId><version>0.0.1-SNAPSHOT</version><name>example-independent-maven-spring-project</name><description>Demo project for Spring Boot</description><properties><java.version>1.8</java.version></properties><dependencies><dependency><groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-web</artifactId></dependency><dependency><groupId>org.springframework.boot</groupId><artifactId>spring-boot-devtools</artifactId><scope>runtime</scope><optional>true</optional></dependency><dependency><groupId>org.projectlombok</groupId><artifactId>lombok</artifactId><optional>true</optional></dependency><dependency><groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-test</artifactId><scope>test</scope></dependency><!--https://mvnrepository.com/artifact/org.hibernate/hibernate-validator -->

<dependency><groupId>org.hibernate</groupId><artifactId>hibernate-validator</artifactId><version>6.1.0.Final</version></dependency></dependencies><build><plugins><plugin><groupId>org.springframework.boot</groupId><artifactId>spring-boot-maven-plugin</artifactId></plugin></plugins></build></project>

**Prepare**

Trước khi tạo ra một bộ Validator cho riêng mình, chúng ta tạo ra các thành phần chính.

Tạo ra model User

*User.java*

@Data

public class User {

private String lodaId;

}

Tạo Controller giao tiếp với client.

*UserController.java*

@RestController

@RequestMapping("api/v1/users")

public class UserController {

/\*

Đánh dấu object với @Valid để validator tự động kiểm tra object đó có hợp lệ hay không

\*/

@PostMapping

public Object createUser(@Valid @RequestBody User user) {

return user;

}

}

**Tạo Annotation**

Để tạo ra một validator với Hibernate-Validator, chúng ta cần khai báo một annotation mới.

Ở đây, tôi tạo ra một Annotation của chính mình như sau:

/\*\*

\* Khai báo một custom annotation

\*/

@Documented

@Constraint(validatedBy = LodaIdValidator.class)

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

@Retention(RetentionPolicy.RUNTIME)

public @interface LodaId {

// trường message là bắt buộc, khai báo nội dung sẽ trả về khi field k hợp lệ

String message() default "LodaId must start with loda://";

// Cái này là bắt buộc phải có để Hibernate Validator có thể hoạt động

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

// Cái này là bắt buộc phải có để Hibernate Validator có thể hoạt động

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

}

Vậy là xong, giờ chỉ cần gắn @LodaId lên trường nào cần kiểm tra tính hợp lệ.

@Data

public class User {

// Đánh dấu field lodaId sẽ cần validate bởi @LodaId

@LodaId

private String lodaId;

}

**Tạo Validator**

Sau khi đã tạo ra @LodaId, chúng ta cần tạo ra bộ kiểm tra cho Annotation này.

Rất đơn giản, bạn chỉ cần kế thừa lớp ConstraintValidator của Hibernate Validator

import javax.validation.ConstraintValidator;

import javax.validation.ConstraintValidatorContext;

public class LodaIdValidator implements ConstraintValidator<LodaId, String> {

private static final String LODA\_PREFIX = "loda://";

/\*\*

\* Kiểm tra tính hợp lệ của trường được đánh dấu bởi @LodaId

\* @param s

\* @param constraintValidatorContext

\* @return

\*/

@Override

public boolean isValid(String s, ConstraintValidatorContext constraintValidatorContext) {

if (s == null || s.isEmpty()) return false;

return s.startsWith(LODA\_PREFIX);

}

}

**Chạy thử**

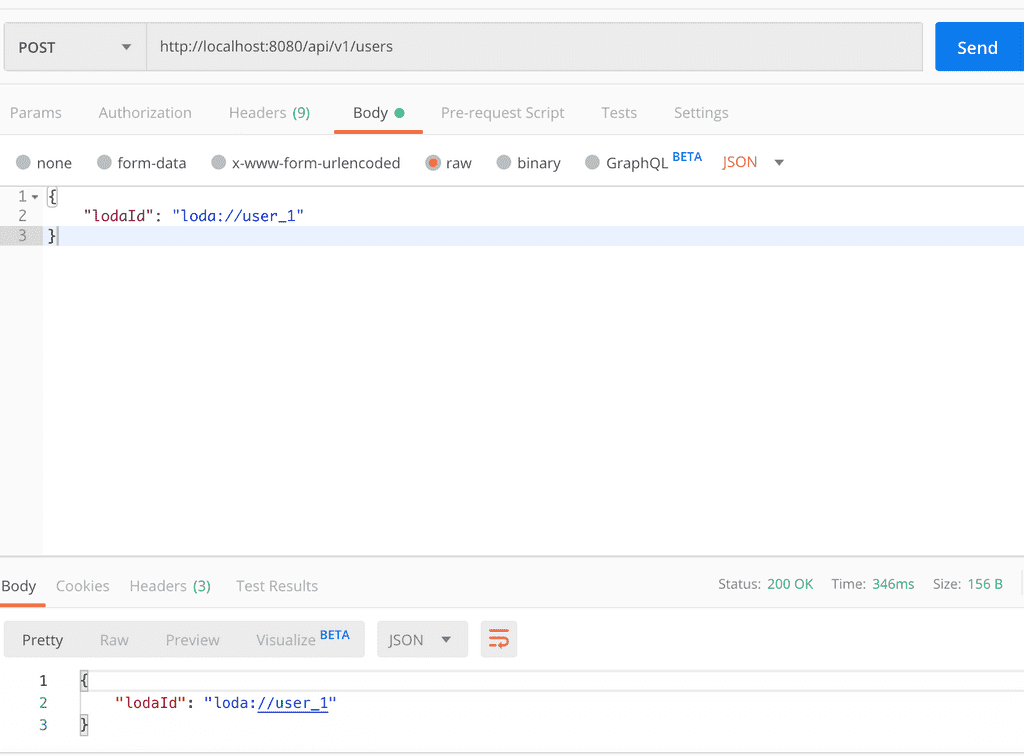
Request một User hợp lệ:

{

"lodaId": "loda://user\_1"

}

trả về thành công 200.



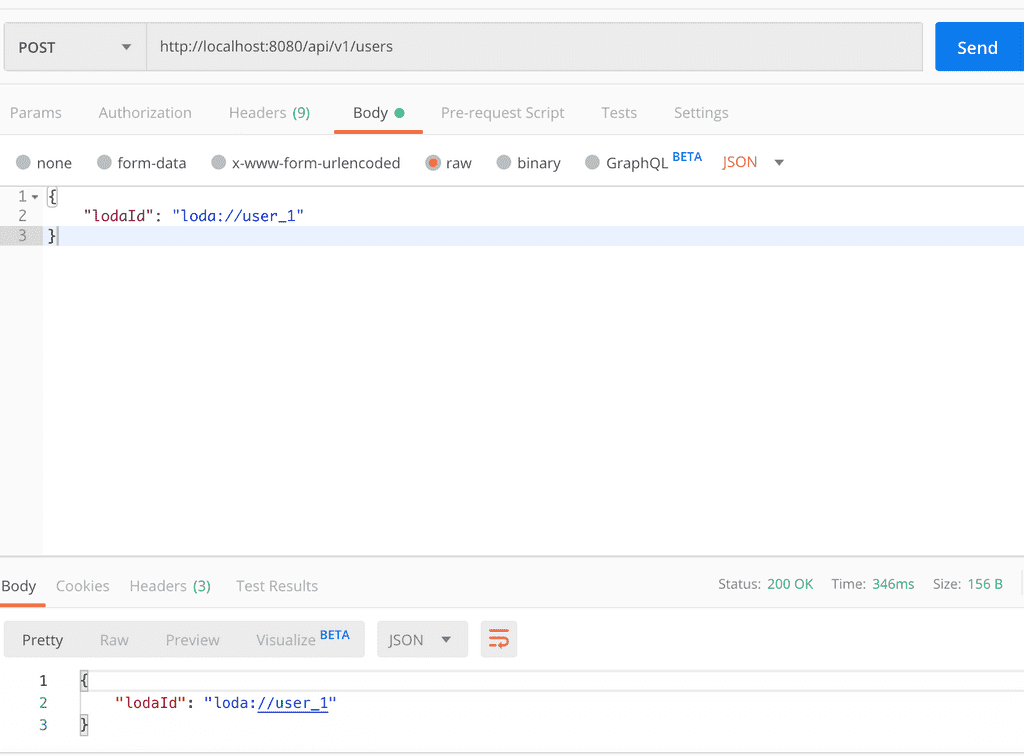
Request một User không hợp lệ:

{

"lodaId": "Laula://user\_1"

}

Trả về thất bại, Bad Request 400.



{

"timestamp": "2019-12-19T10:26:14.554+0000",

"status": 400,

"error": "Bad Request",

"errors": [

{

"codes": [

"LodaId.user.lodaId",

"LodaId.lodaId",

"LodaId.java.lang.String",

"LodaId"

],

"arguments": [

{

"codes": [

"user.lodaId",

"lodaId"

],

"arguments": null,

"defaultMessage": "lodaId",

"code": "lodaId"

}

],

"defaultMessage": "LodaId must start with loda://",

"objectName": "user",

"field": "lodaId",

"rejectedValue": "Laula://user\_1",

"bindingFailure": false,

"code": "LodaId"

}

],

"message": "Validation failed for object='user'. Error count: 1",

"path": "/api/v1/users"

}

**Kết**

Rất đơn giản để có thể tự tạo ra cho mình các Validator với Hibernate-Validator. Nó sẽ giúp bạn tiết kiệm rất nhiều thời gian.