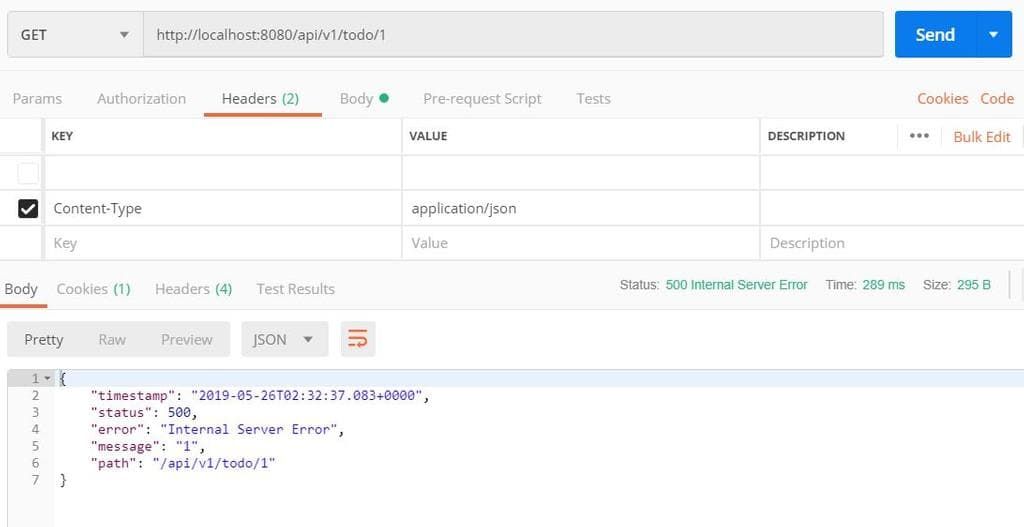
**Giới thiệu**

[Trong bài trước](https://loda.me/spring-boot-14-restful-api-rest-controller-path-variable-request-body-loda1558775921707) chúng ta đã biết cách làm một Rest Api Server với **Spring Boot**.

[🖊️](https://loda.me/articles/sb14-restful-api-restcontroller-pathvariable-requestbody)

[[SB14] Restful API + @RestController + @PathVariable + @RequestBody](https://loda.me/articles/sb14-restful-api-restcontroller-pathvariable-requestbody)

Tuy nhiên có một số vấn đề mà chúng ta kiểm soát. Ví dụ như nếu người dùng request lên một id không hề tồn tại thì sao?



Trong bài viết này, chúng ta sẽ tìm hiểu cách xử lý Exception trong **Spring Boot**

**@RestControllerAdvice & @ControllerAdvice + @ExceptionHandler**

@RestControllerAdvice là một Annotation gắn trên Class. Có tác dụng xen vào quá trình xử lý của các @RestController. Tương tự với @ControllerAdvice

@RestControllerAdvice thường được kết hợp với @ExceptionHandler để cắt ngang quá trình xử lý của Controller, và xử lý các ngoại lệ xảy ra.

@RestControllerAdvice

public class ApiExceptionHandler {

@ExceptionHandler(IndexOutOfBoundsException.class)

@ResponseStatus(value = HttpStatus.BAD\_REQUEST)

public ErrorMessage TodoException(Exception ex, WebRequest request) {

return new ErrorMessage(10100, "Đối tượng không tồn tại");

}

}

Hiểu đơn giản là Controller đang hoạt động bình thường, chẳng may có một Exception được ném ra, thì thay vì báo lỗi hệ thống, thì nó sẽ được thằng @RestControllerAdvice và @ExceptionHandler đón lấy và xử lý. Sau đó trả về cho người dùng thông tin hữu ích hơn.

**@ResponseStatus**

@ResponseStatus là một cách định nghĩa Http Status trả về cho người dùng.

Nếu bạn không muốn sử dụng ResponseEntity thì có thể dùng @ResponseStatus đánh dấu trên Object trả về.

**Demo**

**Cài đặt**

*pom.xml*

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

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<packaging>pom</packaging>

<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>spring-boot-learning</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-boot-learning</name>

<description>Everything about Spring Boot</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<!--spring mvc, rest-->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

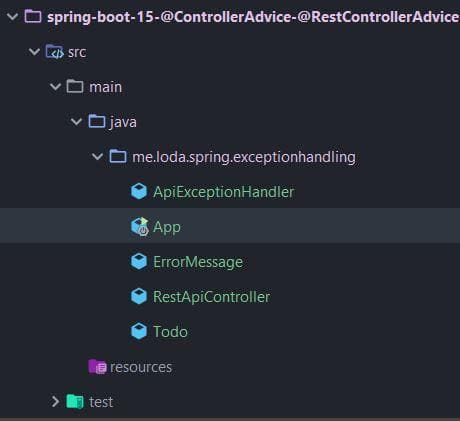
</plugin>

</plugins>

</build>

</project>

Cấu trúc thư mục:



**Tạo model**

Sử dụng [Lombok](https://loda.me/general-huong-dan-su-dung-lombok-giup-code-java-nhanh-hon-69-loda1552789752787) cho tiện nha các bạn.

*Todo.java*

@Data

@AllArgsConstructor

public class Todo {

private String title;

private String detail;

}

Tạo ra class ErrorMessage để chứa thông tin trả về cho Client.

*ErrorMessage.java*

@Data

@AllArgsConstructor

public class ErrorMessage {

private int statusCode;

private String message;

}

**Tạo Controller**

*RestApiController.java*

/\*\*

\* Lưu ý, @RequestMapping ở class, sẽ tác động tới

\* tất cả các RequestMapping ở bên trong nó.

\* <p>

\* Mọi Request ở trong method sẽ được gắn thêm prefix /api/v1

\*/

@RestController

@RequestMapping("/api/v1")

public class RestApiController {

private List<Todo> todoList;

// bạn còn nhớ @PostConstruct dùng để làm gì chứ?

// nếu không nhớ, hãy coi lại bài viết Spring Boot #3 nhé

@PostConstruct

public void init() {

todoList = IntStream.range(0, 10)

.mapToObj(i -> new Todo("title-" + i, "detail-" + i))

.collect(Collectors.toList());

}

/\*

phần path URL bạn muốn lấy thông tin sẽ để trong ngoặc kép {}

\*/

@GetMapping("/todo/{todoId}")

public Todo getTodo(@PathVariable(name = "todoId") Integer todoId) {

// @PathVariable lấy ra thông tin trong URL

// dựa vào tên của thuộc tính đã định nghĩa trong ngoặc kép /todo/{todoId}

return todoList.get(todoId);

}

}

**Tạo Exception Handler**

*ApiExceptionHandler.java*

@RestControllerAdvice

public class ApiExceptionHandler {

/\*\*

\* Tất cả các Exception không được khai báo sẽ được xử lý tại đây

\*/

@ExceptionHandler(Exception.class)

@ResponseStatus(value = HttpStatus.INTERNAL\_SERVER\_ERROR)

public ErrorMessage handleAllException(Exception ex, WebRequest request) {

// quá trình kiểm soat lỗi diễn ra ở đây

return new ErrorMessage(10000, ex.getLocalizedMessage());

}

/\*\*

\* IndexOutOfBoundsException sẽ được xử lý riêng tại đây

\*/

@ExceptionHandler(IndexOutOfBoundsException.class)

@ResponseStatus(value = HttpStatus.BAD\_REQUEST)

public ErrorMessage TodoException(Exception ex, WebRequest request) {

return new ErrorMessage(10100, "Đối tượng không tồn tại");

}

}

**Chạy thử**

*App.java*

@SpringBootApplication

public class App {

public static void main(String[] args) {

SpringApplication.run(App.class, args);

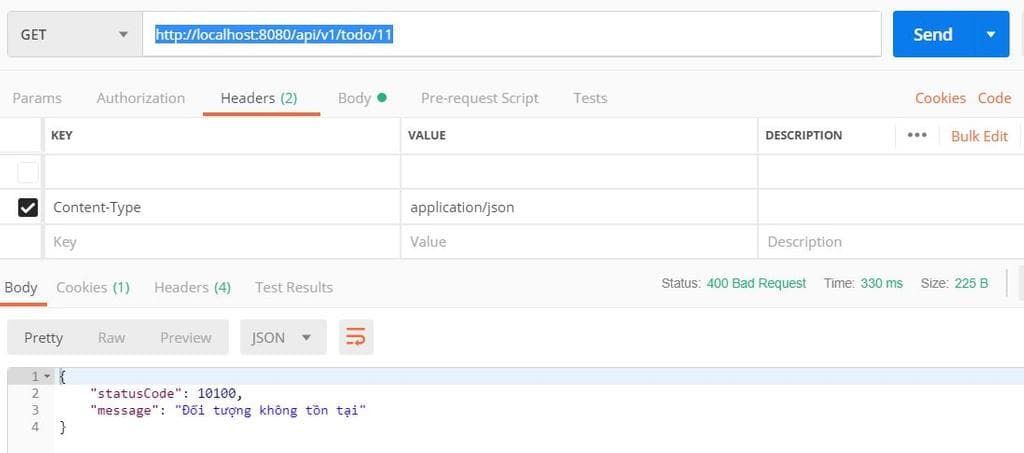
}

}

Gửi request tới địa chỉ:

GEThttp://localhost:8080/api/v1/todo/11

Ở đây, đối tượng 11 không tồn tại trong danh sách, chúng ta sẽ trả về lỗi cho phía Client.



**Kết**