

LẬP TRÌNH JAVA NÂNG CAO

CHƯƠNG 7: RestFull API



Giảng Viên Giảng Dạy:
ThS. Nguyễn Minh Tân
ThS. Đặng Đình Hòa
ThS. Trần Công Thanh
HỌC KỲ III – NĂM HỌC 2023-2024



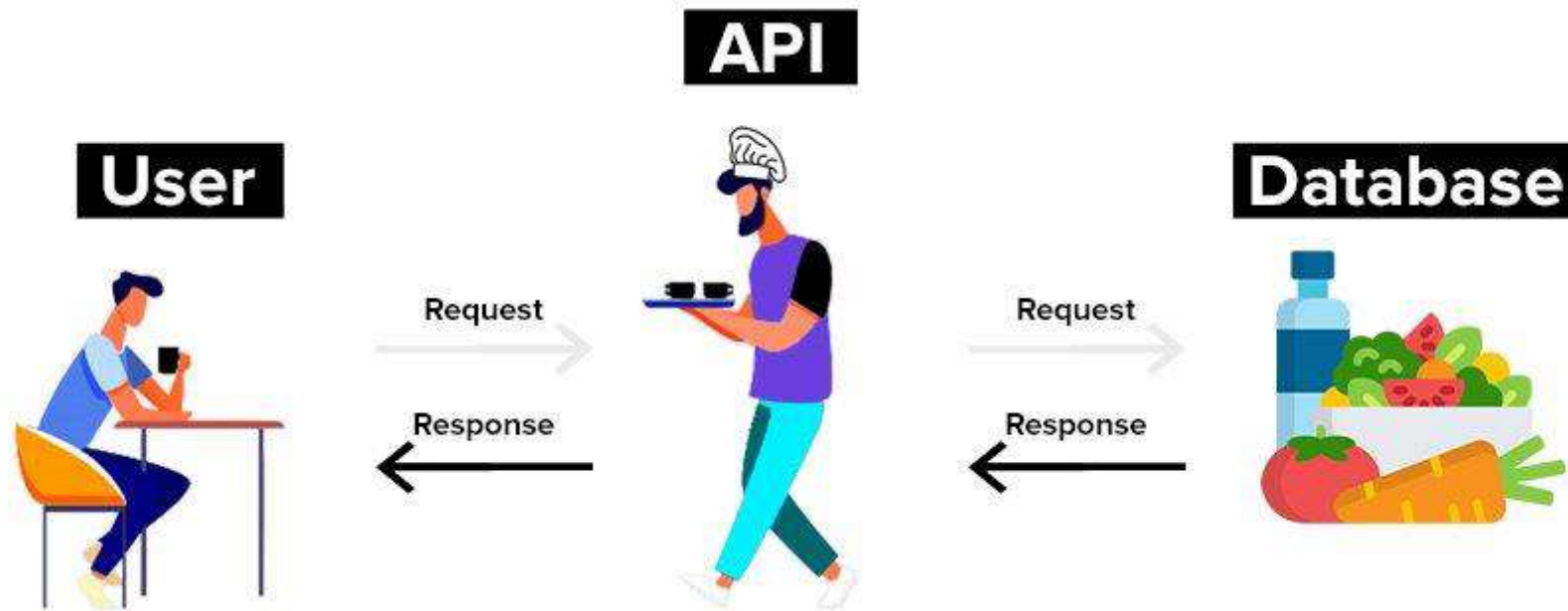
KHÓA 27T-IT



Nội dung

- 1 RestAPI
- 2 JSON
- 3 POJO
- 4 Exception
- 5 Q & A

REST API Representational State Transfer





API – Application Programming Interface

<https://www.leadmine.net/glossary/application-programming-interface/>

- API (Application Programming Interface) là một tập hợp các quy tắc, giao thức và công cụ cho phép hai phần mềm tương tác và trao đổi dữ liệu với nhau.
- Nó định nghĩa các phương thức và định dạng dữ liệu mà các ứng dụng khác nhau có thể sử dụng để giao tiếp và tương tác với nhau.



REST

- Là một kiến trúc phần mềm được sử dụng trong lập trình web để thiết kế các dịch vụ web dựa trên giao thức HTTP
- REST được phát triển bởi Roy Fielding trong luận văn tiến sĩ của ông vào năm 2000



Các nguyên tắc cơ bản của REST:

- Tài nguyên (Resources): Tài nguyên là các đối tượng có thể được truy cập hoặc được thao tác trong dịch vụ web. Mỗi tài nguyên được đại diện bằng một URI (Uniform Resource Identifier) duy nhất.
- Ví dụ, trong một ứng dụng quản lý sách, tài nguyên có thể là "sách" và URI có thể là "/books".
- Phân biệt URL và URI



Các nguyên tắc cơ bản của REST:

- Giao thức HTTP : REST sử dụng các phương thức HTTP như GET, POST, PUT và DELETE để thực hiện các hoạt động tương ứng với các tài nguyên (resources) trên máy chủ.
- Ví dụ, để lấy thông tin về một cuốn sách từ dịch vụ web, ta sử dụng phương thức GET trên URI `"/books/{id}"`



Các nguyên tắc cơ bản của REST:

- Định dạng dữ liệu (Data Formats) : Dữ liệu truyền tải giữa máy khách và máy chủ thông qua RESTful webservice thường được sử dụng các định dạng dữ liệu như JSON (JavaScriptObject Notation) hoặc XML (eXtensible Markup Language).

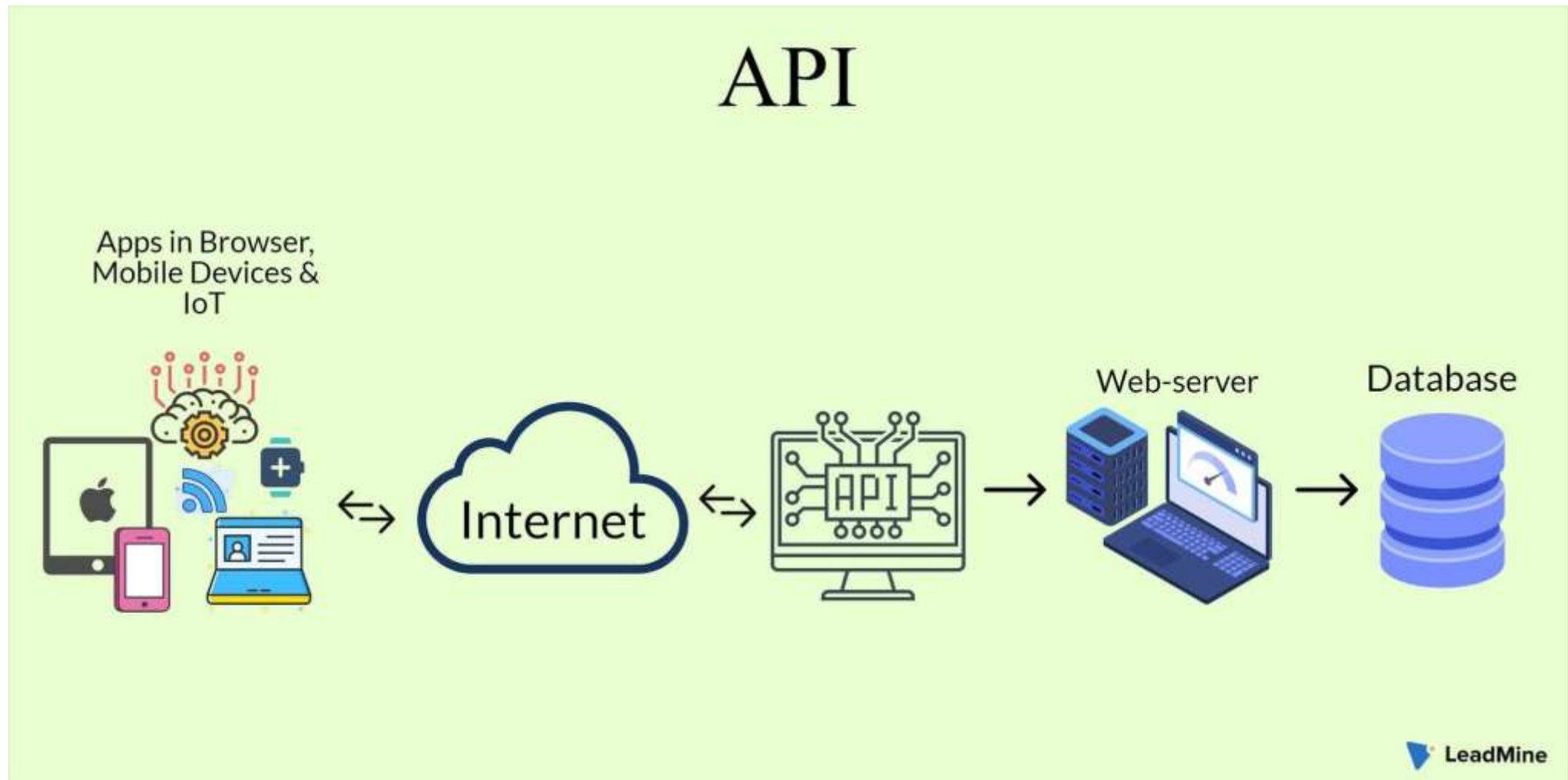


Các nguyên tắc cơ bản của REST:

- Định dạng dữ liệu (Data Formats) : Dữ liệu truyền tải giữa máy khách và máy chủ
- Trạng thái (Statelessness) : RESTful webservice tuân thủ nguyên tắc của giao tiếp không trạng thái, nghĩa là mỗi yêu cầu từ phía máy khách phải chứa đủ thông tin để máy chủ hiểu và xử lý yêu cầu, không cần lưu trạng thái trước đó.

Có phụ thuộc vào ngôn ngữ lập trình không?

<https://www.leadmine.net/glossary/application-programming-interface/>





Data Format

- Định dạng dữ liệu (Data Formats) : Dữ liệu truyền tải giữa máy khách và máy chủ
- Có thể sử dụng bất kỳ định dạng nào
- Phổ biến : XML và JSON
- JSON (JavaScript Object Notation) được sử dụng nhiều nhất



Ví dụ lấy thông tin IP

<https://ipinfo.io/161.185.160.93/geo>

<https://ipinfo.io/27.64.26.21/geo>

```
▼ {  
  "ip": "161.185.160.93",  
  "city": "New York City",  
  "region": "New York",  
  "country": "US",  
  "loc": "40.7143,-74.0060",  
  "org": "AS22252 The City of New York",  
  "postal": "10001",  
  "timezone": "America/New_York",  
  "readme": "https://ipinfo.io/missingauth"  
}
```

```
▼ {  
  "ip": "27.64.26.21",  
  "city": "Ho Chi Minh City",  
  "region": "Ho Chi Minh",  
  "country": "VN",  
  "loc": "10.8230,106.6296",  
  "org": "AS7552 Viettel Group",  
  "postal": "71606",  
  "timezone": "Asia/Ho_Chi_Minh",  
  "readme": "https://ipinfo.io/missingauth"  
}
```

Ví dụ lấy thông tin giá Bitcoin

<https://api.coindesk.com/v1/bpi/currentprice.json>

```
{
  "time": {
    "updated": "Jul 5, 2024 11:50:16 UTC",
    "updatedISO": "2024-07-05T11:50:16+00:00",
    "updateduk": "Jul 5, 2024 at 12:50 BST"
  },
  "disclaimer": "This data was produced from the CoinDesk Bitcoin Price Index (USD). Non-USD currency data converted using hourly conversion rate from openexchangerates.org",
  "chartName": "Bitcoin",
  "bpi": {
    "USD": {
      "code": "USD",
      "symbol": "$",
      "rate": "55,585.326",
      "description": "United States Dollar",
      "rate_float": 55585.3257
    },
    "GBP": {
      "code": "GBP",
      "symbol": "pound;",
      "rate": "43,465.779",
      "description": "British Pound Sterling",
      "rate_float": 43465.7792
    },
    "EUR": {
      "code": "EUR",
      "symbol": "euro;",
      "rate": "51,351.614",
      "description": "Euro",
      "rate_float": 51351.6138
    }
  }
}
```

Ví dụ lấy thông tin thời tiết

<https://www.weatherbit.io/api/weather-current>



The image shows the Weatherbit website interface. At the top, there is a navigation bar with links: Features, Pricing, Weather API's, Resources, Contact Us, Log in, and Sign up. The main heading is "Weather API for Traders" with the subtitle "The High Performance Weather API for All of Your Data Needs." Below this, there is a sign-up form on the left and a weather forecast widget on the right.

Sign-up Form:

Enhance your business with our global weather data. Start with a free trial in just a few clicks!

Your email:

Choose a username:

Choose a password:

Create Account

Weather Forecast (Ivano-Frankivsk, UA):

16°C

Wind: 3 m/s
Precip: 0 mm/hr
Pressure: 1020 mb

Scattered Clouds

SUN	MON	TUE	WED	THU	FRI
21°C / 18°C	23°C / 17°C	25°C / 18°C	23°C / 11°C	23°C / 9°C	24°C / 11°C



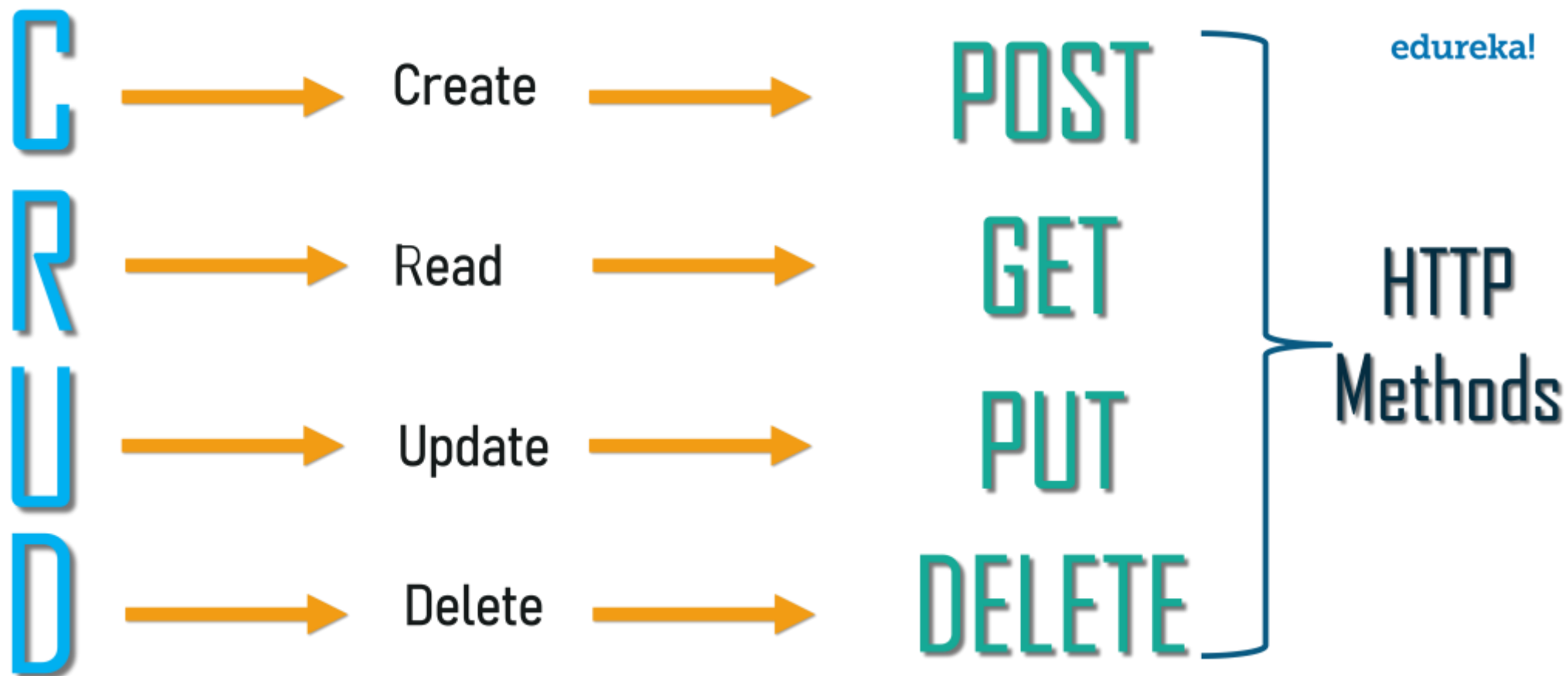
Các loại dịch vụ

- REST API
- REST WEB
- SERVICESREST SERVICES
- RESTful API
- RESTful WEB SERVICES
- RESTful SERVICES



Method API

<https://www.edureka.co/blog/what-is-rest-api/>





Trạng thái status

Mã Trạng thái	Nhóm	Ý nghĩa
1xx	Informational	Thông tin
100-199	-	Thông tin
2xx	Success	Thành công
200-299	+	Yêu cầu thành công
3xx	Redirection	Chuyển hướng
300-399	-	Chuyển hướng
4xx	Client Errors	Lỗi từ phía client
400-499	-	Lỗi từ phía client
5xx	Server Errors	Lỗi từ phía server
500-599	-	Lỗi từ phía server

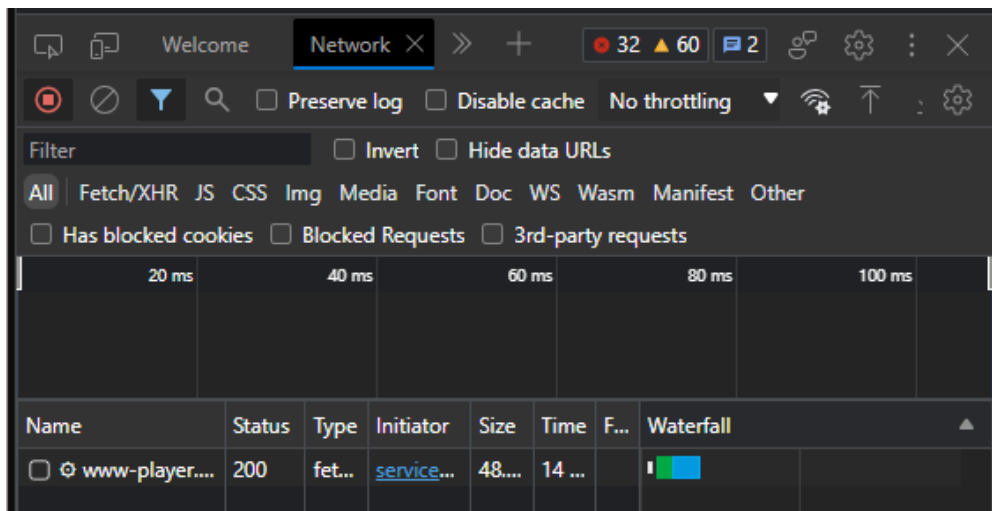


Trạng thái status

Mã Trạng thái	Ý nghĩa
200 OK	+ Yêu cầu thành công
201 Created	+ Tài nguyên đã được tạo mới
204 No Content	+ Yêu cầu thành công, không có nội dung trả về
400 Bad Request	- Yêu cầu không hợp lệ
401 Unauthorized	- Yêu cầu cần xác thực người dùng
403 Forbidden	- Yêu cầu bị từ chối truy cập
404 Not Found	- Tài nguyên không tồn tại
405 Method Not Allowed	- Phương thức không được hỗ trợ
500 Internal Server Error	- Lỗi nội bộ của máy chủ



Test Postman



404



JSON - JavaScript Object Notation

- JSON (JavaScript Object Notation) là một định dạng dữ liệu dựa trên văn bản (text-based) dễ đọc và dễ viết cho việc truyền tải dữ liệu giữa các máy tính.
- Nó đã trở thành một tiêu chuẩn cho truyền tải dữ liệu giữa các ứng dụng web.



Key-value

JSON được sử dụng để đại diện dữ liệu dưới dạng các cặp key-value.

Một đối tượng có thể gồm nhiều cặp giá trị.

Dữ liệu trong JSON được biểu diễn bằng cú pháp đơn giản và nhìn gần giống cú pháp của JavaScript, do đó nó phù hợp với việc sử dụng trong ứng dụng web.

```
json

{
  "name": "John",
  "age": 30,
  "city": "New York"
}
```



Đối tượng JSON lồng nhau

```
{  
  "name": "John",  
  "age": 30,  
  "address": {  
    "street": "123 Main Street",  
    "city": "New York",  
    "country": "USA"  
  }  
}
```

Mảng JSON

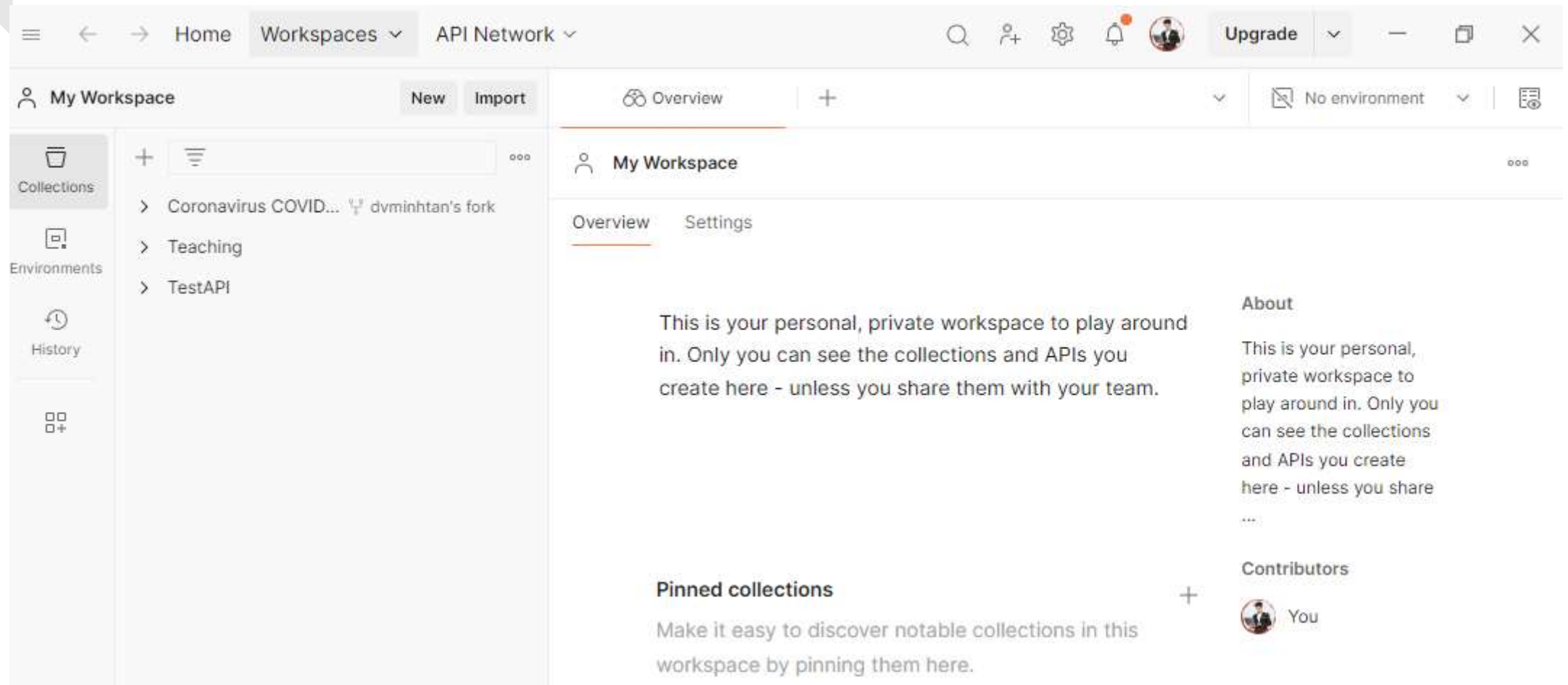
```
{
  "users": [
    {
      "name": "John",
      "age": 30,
      "city": "New York"
    },
    {
      "name": "Alice",
      "age": 25,
      "city": "London"
    },
    {
      "name": "Bob",
      "age": 35,
      "city": "Paris"
    }
  ]
}
```

```
{
  "employees": [
    {
      "name": "John",
      "position": "Manager"
    },
    {
      "name": "Alice",
      "position": "Developer"
    },
    {
      "name": "Bob",
      "position": "Designer"
    }
  ]
}
```

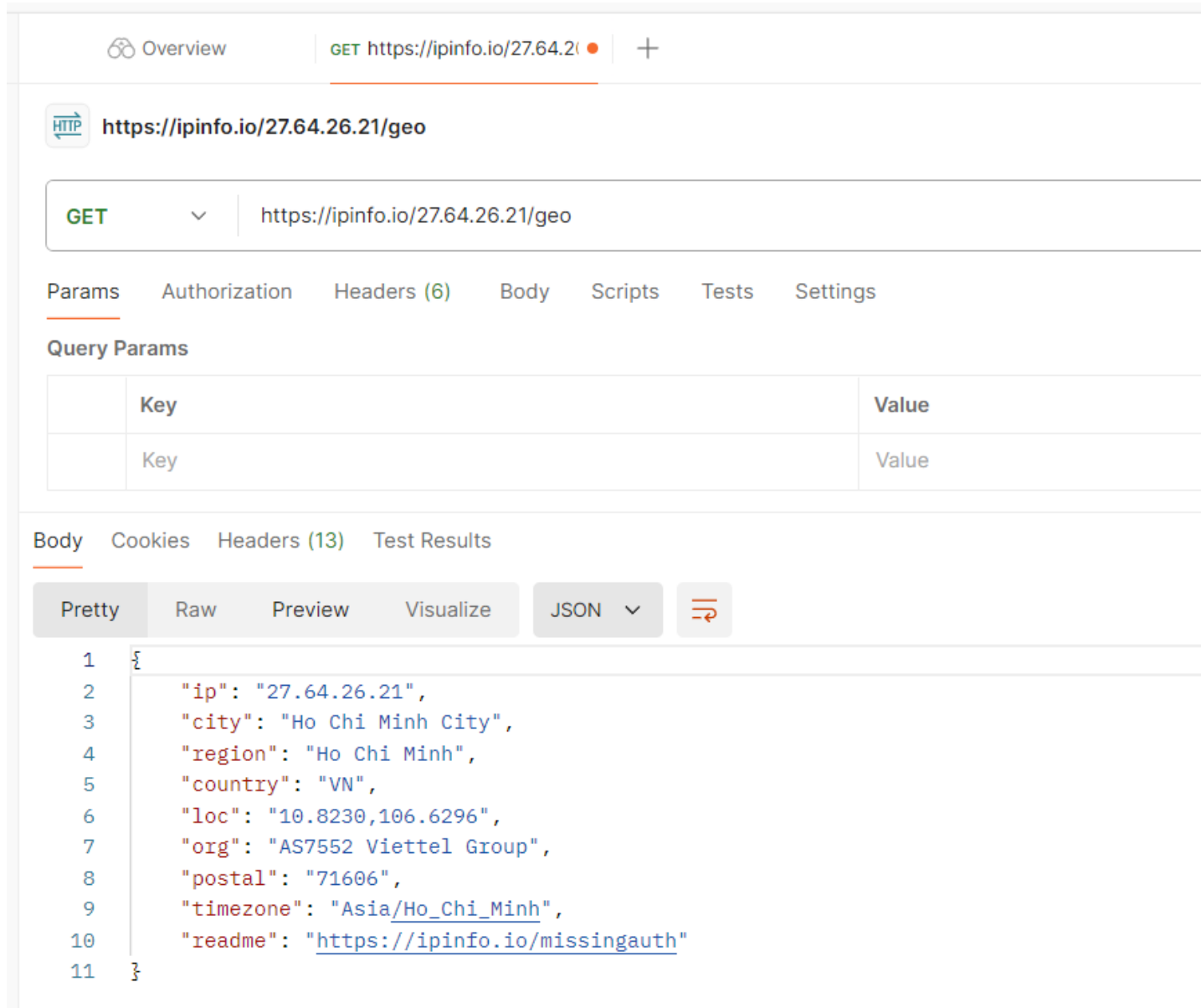
```
{
  "numbers": [1, 2, 3, 4, 5]
}
```

```
{
  "places": [
    {
      "name": "Paris",
      "country": "France",
      "description": "The City of Love"
    },
    {
      "name": "Tokyo",
      "country": "Japan",
      "description": "The Capital of Japan"
    },
    {
      "name": "Rio de Janeiro",
      "country": "Brazil",
      "description": "The Marvelous City"
    }
  ]
}
```

Cài đặt và sử dụng Postman



Cài đặt và sử dụng Postman



The image shows the Postman application interface. At the top, there's a tab for 'Overview' and a request summary: 'GET https://ipinfo.io/27.64.21/geo'. Below this, the request details are shown: 'https://ipinfo.io/27.64.26.21/geo' with a 'GET' method. The 'Params' tab is selected, showing a table for 'Query Params' with two rows, each with 'Key' and 'Value' columns. Below the table, the 'Body' tab is selected, showing a JSON response. The JSON is formatted in 'Pretty' view and contains location and network information for the IP address 27.64.26.21.

Overview | GET https://ipinfo.io/27.64.21/geo | +

HTTP | https://ipinfo.io/27.64.26.21/geo

GET | https://ipinfo.io/27.64.26.21/geo

Params | Authorization | Headers (6) | Body | Scripts | Tests | Settings

Query Params

Key	Value
Key	Value

Body | Cookies | Headers (13) | Test Results

Pretty | Raw | Preview | Visualize | JSON |

```
1 {
2   "ip": "27.64.26.21",
3   "city": "Ho Chi Minh City",
4   "region": "Ho Chi Minh",
5   "country": "VN",
6   "loc": "10.8230,106.6296",
7   "org": "AS7552 Viettel Group",
8   "postal": "71606",
9   "timezone": "Asia/Ho_Chi_Minh",
10  "readme": "https://ipinfo.io/missingauth"
11 }
```



RestController

@RestController

@RestController là một annotation trong Spring Framework được sử dụng để đánh dấu một lớp (class) trong ứng dụng là một RESTful Controller.

Nó kết hợp hai annotation khác là @Controller và @ResponseBody. Khi một lớp được đánh dấu bằng @RestController, nó cho phép Spring hiểu rằng lớp này chịu trách nhiệm xử lý các yêu cầu RESTful từ phía client và trả về dữ liệu dưới dạng JSON hoặc XML (thông qua @ResponseBody).



RestController

@RequestMapping

Là một annotation trong Spring Framework được sử dụng để ánh xạ các yêu cầu HTTP tới các phương thức trong một Controller.

- Nó xác định đường dẫn URL và phương thức HTTP mà một phương thức cụ thể trong Controller sẽ xử lý.
- Annotation @RequestMapping có thể được đặt cả trên cấp lớp (class-level) và phương thức (method-level) trong Controller.
- Khi đặt @RequestMapping trên class, nó xác định tiền tố của đường dẫn URL cho tất cả các phương thức trong Controller.



Hello world

```
@RestController
@RequestMapping("/api")
public class HelloWorldController {

    @GetMapping("/hello")
    public String helloWorld() {
        return "Hello, world!";
    }

    @GetMapping("/hello/spanish")
    public String helloWorldInSpanish() {
        return "¡Hola, mundo!";
    }
}
```

<http://localhost/api/hello>

<http://localhost/api/hello/spanish>



Hello world

```
@RestController
public class HelloWorldController {

    @GetMapping("/hello")
    public String helloWorld() {
        return "Hello, world!";
    }
}
```

<http://localhost/hello>



Hello world

Annotation	Phương thức HTTP	Mô tả
@RequestMapping	GET, POST, PUT, DELETE, và các phương thức HTTP khác	Xác định một đường dẫn URL và phương thức HTTP cho phương thức xử lý
@GetMapping	GET	Xác định một đường dẫn URL và phương thức GET cho phương thức xử lý
@PostMapping	POST	Xác định một đường dẫn URL và phương thức POST cho phương thức xử lý
@PutMapping	PUT	Xác định một đường dẫn URL và phương thức PUT cho phương thức xử lý
@DeleteMapping	DELETE	Xác định một đường dẫn URL và phương thức DELETE cho phương thức xử lý
@PatchMapping	PATCH	Xác định một đường dẫn URL và phương thức PATCH cho phương thức xử lý



Hello world

@PathVariable	-	Xác định một biến đường dẫn (path variable) trong URL để truyền vào phương thức xử lý
@RequestParam	-	Xác định một tham số truy vấn (query parameter) trong URL để truyền vào phương thức xử lý
@RequestBody	-	Xác định một đối tượng được gửi kèm trong phần thân (body) của yêu cầu HTTP
@ResponseBody	-	Xác định dữ liệu trả về từ phương thức xử lý sẽ được gửi về phần thân (body) của phản hồi HTTP
@RestControllerAdvice	-	Xác định một class chứa các hàm xử lý tương tự như Controller Advice cho các RestController
@ExceptionHandler	-	Xác định một phương thức xử lý cho xử lý các ngoại lệ (exception) trong RestController



Thực hành

Xây dựng ứng dụng RestController HelloWorld có sử dụng:

`@RestController @RequestMapping("/api")`

`@GetMapping("/hello")`

`@GetMapping("/hello/vn")`

Kết hợp kiểm tra các Get Request với Postman

Tạo mới dự án spring boot



Project

☐ Gradle - Groovy ☐ Gradle - Kotlin ☒ Java ☐ Kotlin ☐ Groovy
☒ Maven

Spring Boot

☐ 3.3.2 (SNAPSHOT) ☒ 3.3.1 ☐ 3.2.8 (SNAPSHOT) ☐ 3.2.7

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☒ 22 ☐ 21 ☐ 17

Dependencies

[ADD DEPENDENCIES...](#) CTRL + B

Spring Boot DevTools DEVELOPER TOOLS

Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

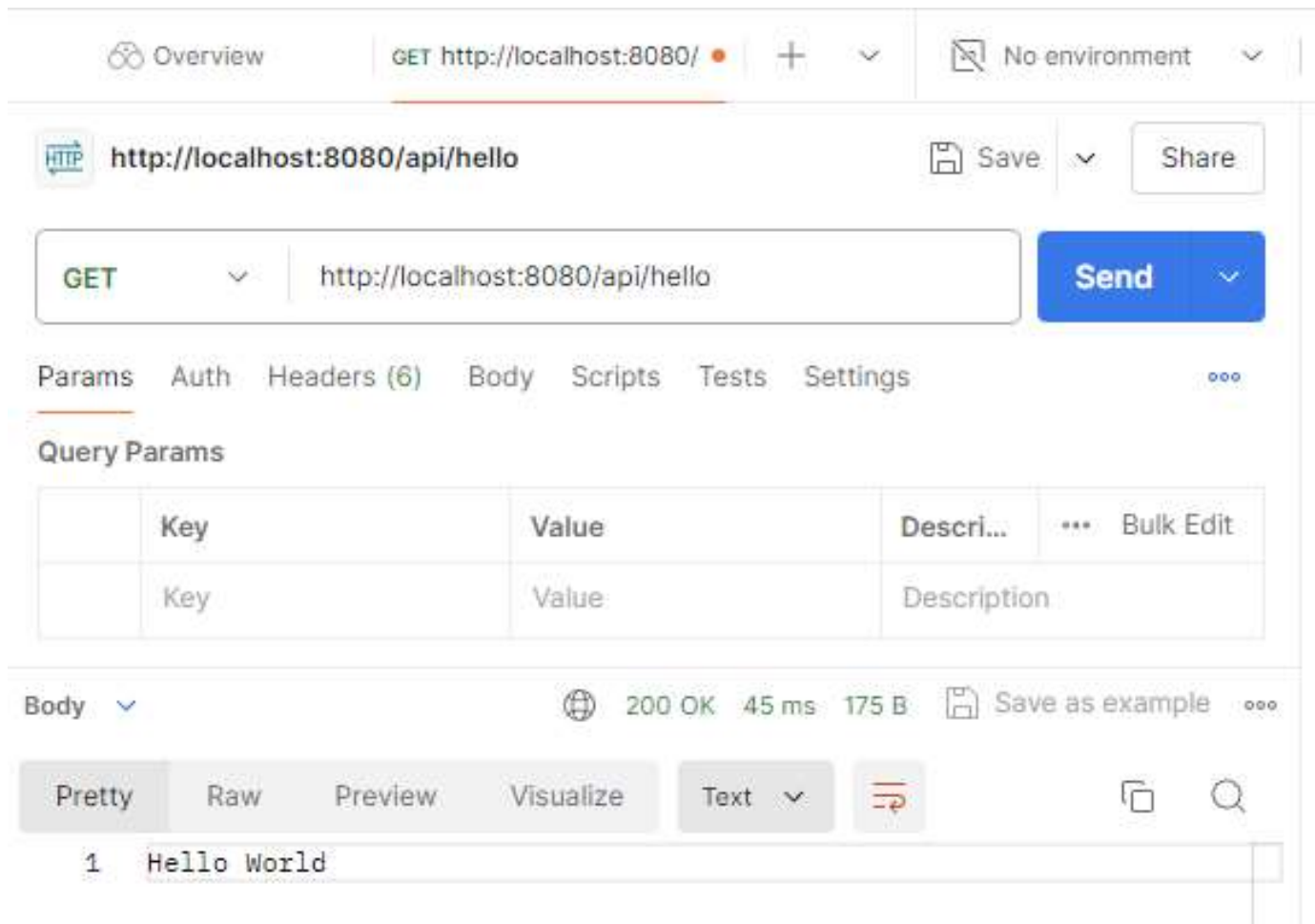
Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Tạo mới dự án spring boot

```
© HelloController.java x
1  package com.vanlang.api;
2
3  import org.springframework.web.bind.annotation.GetMapping;
4  import org.springframework.web.bind.annotation.RequestMapping;
5  import org.springframework.web.bind.annotation.RestController;
6
7  @RestController
8  @RequestMapping("/api")
9  public class HelloController {
10     @GetMapping("/hello")
11     public String hello() {
12         return "Hello World";
13     }
14     @GetMapping("/hello/vn")
15     public String hello2() {
16         return "Xin chao";
17     }
18 }
```

Tạo mới dự án spring boot



The screenshot displays a web client interface with the following components:

- Overview Tab:** Shows the request details for `GET http://localhost:8080/`.
- Request URL:** `http://localhost:8080/api/hello`
- Method:** `GET`
- Send Button:** A blue button labeled "Send".
- Params Tab:** Shows the response details.
- Query Params Table:**

Key	Value	Descri...	...	Bulk Edit
Key	Value	Description		

Body Tab: Shows the response body.

Response Status: 200 OK, 45 ms, 175 B

Response Body:

```
1 Hello World
```



Chuyển đổi JSON thành Java POJO



JSON

```
{  
  "id": 1,  
  "ten": "Nguyễn Văn A",  
  "tuoi": 20,  
  "nganhHoc": "Khoa học máy tính",  
  "diemTB": 3.8  
}
```



Class

```
public class Student {  
    public int id;  
    public String ten;  
    public int tuoi;  
    public String nganhHoc;  
    public double diemTB;  
}
```



Java POJO

- POJO là viết tắt của "Plain Old Java Object", có nghĩa là một lớp đơn giản trong Java không phụ thuộc vào bất kỳ framework hoặc thư viện nào khác.
- Nó là một lớp Java thuần túy, không có các yêu cầu đặc biệt về cú pháp hoặc quy ước.



Đặc điểm của một POJO gồm

POJO là viết tắt của "Plain Old Java Object", có nghĩa Chỉ chứa các thuộc tính (fields) và phương thức (methods) để truy cập và thay đổi các giá trị của thuộc tính (getter–setter).

Không có các ràng buộc về việc kế thừa các lớp hoặc triển khai các interface cụ thể.

Không phụ thuộc vào các thư viện, framework hoặc annotation đặc biệt.

JSON <> POJO gồm

Trong Spring Framework, chuyển đổi giữa JSON và Java POJO được thực hiện thông qua các thành phần của Jackson - một thư viện mạnh mẽ hỗ trợ xử lý JSON trong Java

```
{  
  "id": 1,  
  "ten": "Nguyễn Văn A",  
  "tuoi": 20,  
  "nganhHoc": "Khoa học máy tính",  
  "diemTB": 3.8  
}
```



```
public class Student {  
    public int id;  
    public String ten;  
    public int tuoi;  
    public String nganhHoc;  
    public double diemTB;  
}
```

Thực hành

Xây dựng RestController trả về danh sách (List) các sinh viên

```
{  
  "id": 1,  
  "ten": "Nguyễn Văn A",  
  "tuoi": 20,  
  "nganhHoc": "Khoa học máy tính",  
  "diemTB": 3.8  
}
```



```
public class Student {  
    public int id;  
    public String ten;  
    public int tuoi;  
    public String nganhHoc;  
    public double diemTB;  
}
```

Thực hành

SinhVien.java x SinhVienController.java

```
1  package com.vanlang.api.entity;
2
3  public class SinhVien { 9 usages
4      private int id; 3 usages
5      private String hoVaTen; 3 usages
6      private int tuoi; 3 usages
7      private String nganhHoc; 3 usages
8      private double diemTB; 3 usages
9
10     public SinhVien() { no usages
11     }
12
13     public SinhVien(int id, String hoVaTen, int tuoi, String nganhHoc, double diemTB) { 5 usages
14         this.id = id;
15         this.hoVaTen = hoVaTen;
16         this.tuoi = tuoi;
17         this.nganhHoc = nganhHoc;
18         this.diemTB = diemTB;
19     }
```

Thực hành

```
12 @RestController
13 @RequestMapping("/sinhvien")
14 public class SinhVienController {
15     public List<SinhVien> danhSach; 7 usages
16
17     @PostConstruct
18     public void createDanhSach(){
19         danhSach = new ArrayList<SinhVien>();
20         danhSach.add(new SinhVien(id: 1, hoVaTen: "Nguyen Minh Tan", tuoi: 18, ngànhHoc: "CNTT", diemTB: 6.5));
21         danhSach.add(new SinhVien(id: 2, hoVaTen: "Nguyen Thi Lan", tuoi: 19, ngànhHoc: "QTKD", diemTB: 7.5));
22         danhSach.add(new SinhVien(id: 3, hoVaTen: "Nguyen Anh Sang", tuoi: 17, ngànhHoc: "LT", diemTB: 9.5));
23         danhSach.add(new SinhVien(id: 4, hoVaTen: "Le Van Luyen", tuoi: 20, ngànhHoc: "Marketing", diemTB: 8.5));
24         danhSach.add(new SinhVien(id: 5, hoVaTen: "Tran Thi Huyen", tuoi: 21, ngànhHoc: "Luat", diemTB: 5.5));
25     }
26     @GetMapping("/")
27     public List<SinhVien> getDanhSach(){
28         return danhSach;
29     }
30 }
```

Thực hành

HTTP <http://localhost:8080/sinhvien/> Save Share </>

GET <http://localhost:8080/sinhvien/> Send ↕

Params Auth Headers (6) Body Scripts Tests Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body ⌵

200 OK 15 ms 553 B Save as example ...

Pretty Raw Preview Visualize JSON ⌵ 🔍

```
1 [
2   {
3     "id": 1,
4     "hoVaTen": "Nguyen Minh Tan",
5     "tuoi": 18,
6     "nganhHoc": "CNTT",
7     "diemTB": 6.5
8   },
9   {
10    "id": 2,
11    "hoVaTen": "Nguyen Thi Lan",
12    "tuoi": 19,
13    "nganhHoc": "QTKD",
14    "diemTB": 7.5
15  },
16  {
17    "id": 3,
18    "hoVaTen": "Nguyen Anh Sang",
19    "tuoi": 17,
20    "nganhHoc": "LT",
21    "diemTB": 8.5
```



Path Variables

- Path Variables trong Spring là một cách để trích xuất thông tin từ URL và sử dụng nó như một tham số trong phương thức xử lý trong một RestController.
- Thông tin này được chứa trong các đoạn đường dẫn cụ thể được định nghĩa trong `@RequestMapping` hoặc `@GetMapping`, `@PostMapping`, vv.



Path Variables

```
@RestController
@RequestMapping("/api/students")
public class StudentController {

    @GetMapping("/{id}")
    public ResponseEntity<Student> getStudentById(@PathVariable int id) {
        // Xử lý yêu cầu để lấy thông tin sinh viên với ID tương ứng
        // Sử dụng ID được truyền vào phương thức như một Path Variable
    }
}
```



Path Variables

Xây dựng RestControlle có sử dụng PathVariables để lấy ra 1 sinh viên.
Thông qua các PathVariables sau đây:


`{studentId}`

`index/{index}`

Path Variables

```
31      @GetMapping("/{id}")
32      public SinhVien getSinhVien(@PathVariable int id){
33          for(SinhVien sv : danhSach){
34              if(sv.getId() == id){
35                  return sv;
36              }
37          }
38          return null;
39      }
40      @GetMapping("/index/{index}")
41      public SinhVien getSinhVien2(@PathVariable int index){
42          return danhSach.get(index);
43      }
44  }
```

Path Variables



HTTP <http://localhost:8080/sinhvien/4> Save Share

GET <http://localhost:8080/sinhvien/4> Send

Params Auth Headers (6) Body Scripts Tests Settings Cookies

Query Params

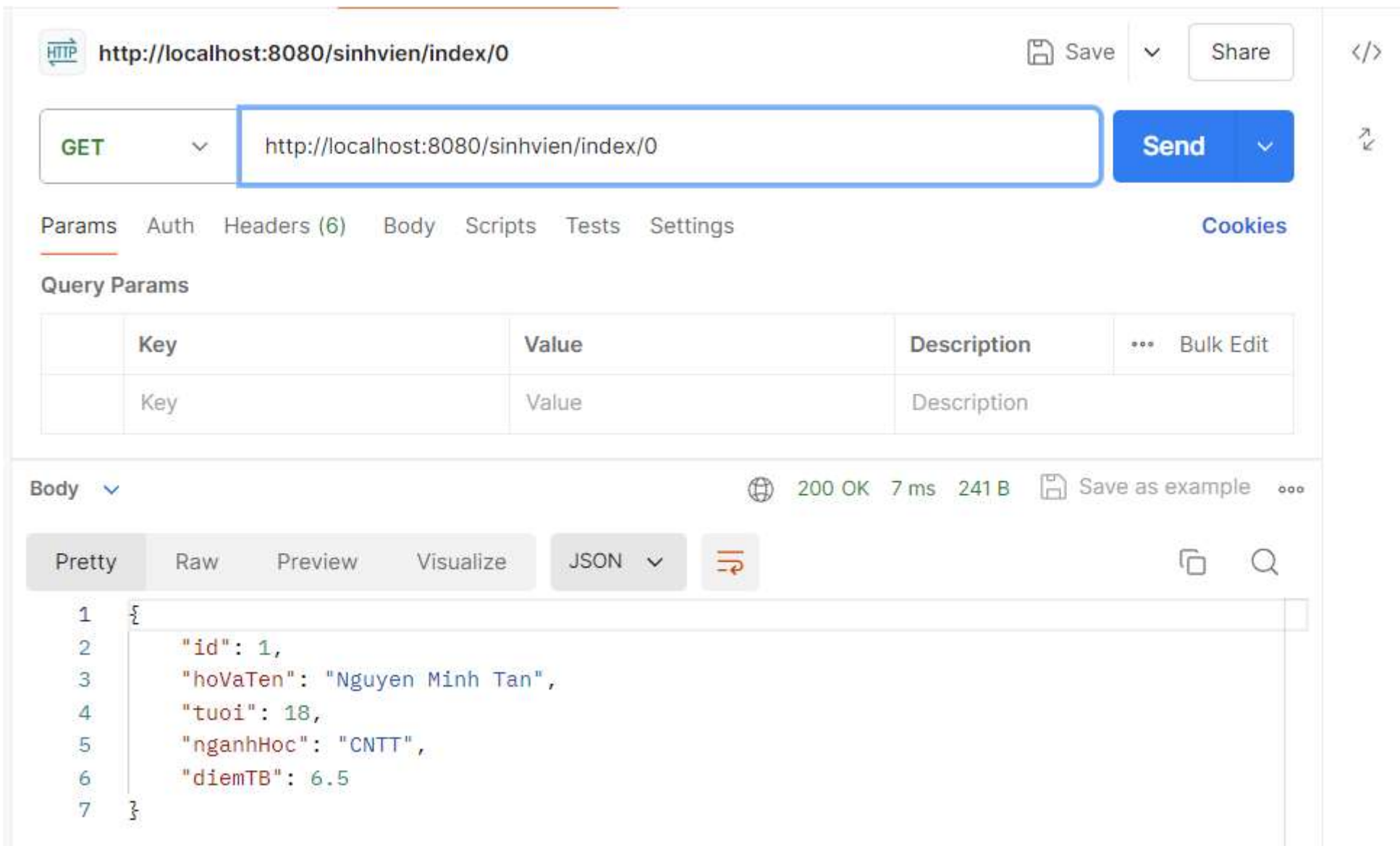
	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body 200 OK 5 ms 243 B Save as example

Pretty Raw Preview Visualize JSON Copy Search

```
1 {  
2   "id": 4,  
3   "hoVaTen": "Le Van Luyen",  
4   "tuoi": 20,  
5   "nganhHoc": "Marketing",  
6   "diemTB": 8.5  
7 }
```

Path Variables



The screenshot shows a web browser's developer tools interface. At the top, the address bar displays the URL `http://localhost:8080/sinhvien/index/0`. Below the address bar, the **GET** method is selected, and the same URL is entered in the request field. The **Send** button is visible. The **Params** tab is active, showing a table for query parameters. The **Body** tab is also visible, showing the response body in JSON format.

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body

200 OK 7 ms 241 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {  
2   "id": 1,  
3   "hoVaTen": "Nguyen Minh Tan",  
4   "tuoi": 18,  
5   "nganhHoc": "CNTT",  
6   "diemTB": 6.5  
7 }
```



Xử lý các ngoại lệ Exception Handling

@ExceptionHandler

Được sử dụng để xử lý các ngoại lệ (exception) trong một RestController hoặc một Controller.

Khi một ngoại lệ xảy ra trong phương thức xử lý yêu cầu của một RestController, Spring sẽ tìm kiếm phương thức được chú thích bằng @ExceptionHandler để xử lý ngoại lệ đó. Phương thức xử lý ngoại lệ này được gọi tự động và có thể thực hiện các xử lý tùy chỉnh như gửi phản hồi lỗi hoặc xử lý lỗi trong cách mà ứng dụng mong muốn.



Xử lý các ngoại lệ Exception Handling

- Tạo các lớp response error
- Tạo các lớp ngoại lệ
- Quăng ra các ngoại lệ nếu gặp lỗi



Xử lý các ngoại lệ Exception Handling

```
public class ErrorResponse {  
    private int status;  
    private String message;  
    private long timestamp;  
  
    public ErrorResponse(int status, String message) {  
        this.status = status;  
        this.message = message;  
        this.timestamp = System.currentTimeMillis();  
    }  
  
    public int getStatus() {  
        return status;  
    }  
  
    public void setStatus(int status) {  
        this.status = status;  
    }  
}
```



Xử lý các ngoại lệ Exception Handling


```
public class StudentNotFoundException extends RuntimeException {  
    public StudentNotFoundException(String message) {  
        super(message);  
    }  
}
```



Xử lý các ngoại lệ Exception Handling

```
@GetMapping("/{id}")  
public ResponseEntity<Student> getStudentById(@PathVariable int id) {  
    Student student = studentService.getStudentById(id);  
    if (student == null) {  
        throw new StudentNotFoundException("Student not found with ID: " + id);  
    }  
    return ResponseEntity.ok().body(student);  
}
```


Xử lý các ngoại lệ Exception Handling



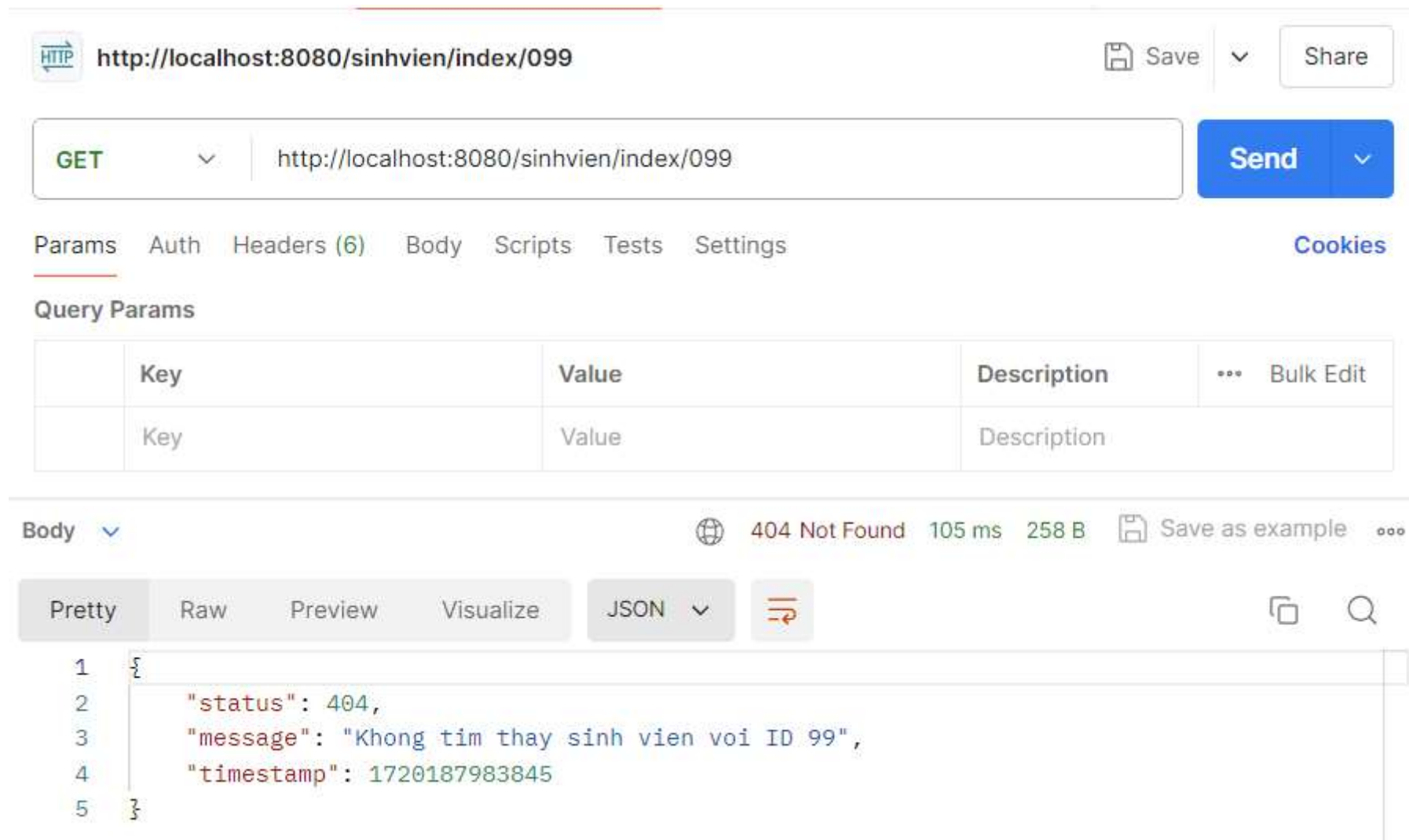
```
© SinhVien.java    © SinhVienController.java    © ErrorResponse.java x    ⚡ SinhVienException.java

1  package com.vanlang.api.entity;
2
3  public class ErrorResponse { no usages
4      private int status; 3 usages
5      private String message; 3 usages
6      private long timestamp; 3 usages
7      public ErrorResponse() {} no usages
8      public ErrorResponse(int status, String message) { no usages
9          this.status = status;
10         this.message = message;
11         this.timestamp = System.currentTimeMillis();
12     }
13
14     public int getStatus() { no usages
15         return status;
16     }
17
18     public void setStatus(int status) { no usages
19         this.status = status;
20     }
```

Xử lý các ngoại lệ Exception Handling

```
16 public class SinhVienController {
32     @GetMapping("/{id}")
33     public SinhVien getSinhVien(@PathVariable int id){
34         SinhVien result = null;
35         for(SinhVien sv : danhSach){
36             if(sv.getId() == id){
37                 return sv;
38             }
39         }
40         if(result == null){
41             throw new SinhVienException("Không tìm thấy sinh viên với ID "+id);
42         }
43         return null;
44     }
45     @GetMapping("/index/{index}")
46     public SinhVien getSinhVien2(@PathVariable int index){
47         SinhVien result = null;
48         try {
49             result = danhSach.get(index);
50         } catch (Exception e){
51             throw new SinhVienException("Không tìm thấy sinh viên với ID "+index);
52         }
53         return result;
54     }
55     @ExceptionHandler
56     @ public ResponseEntity<ErrorResponse> batLoi(SinhVienException ex){
57         ErrorResponse er = new ErrorResponse(HttpStatus.NOT_FOUND.value(), ex.getMessage());
58         return ResponseEntity.status(HttpStatus.NOT_FOUND).body(er);
59     }
60 }
```

Xử lý các ngoại lệ Exception Handling



HTTP <http://localhost:8080/sinhvien/index/099> Save Share

GET <http://localhost:8080/sinhvien/index/099> Send

Params Auth Headers (6) Body Scripts Tests Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body 404 Not Found 105 ms 258 B Save as example ...

Pretty Raw Preview Visualize JSON ...

```
1 {  
2   "status": 404,  
3   "message": "Khong tim thay sinh vien voi ID 99",  
4   "timestamp": 1720187983845  
5 }
```



TRƯỜNG ĐẠI HỌC
VĂN LANG

KHOA CÔNG NGHỆ THÔNG TIN

