**API RESTFul**

Contents

[1) API 1](#_Toc17314860)

[**a)** **Định nghĩa:** 1](#_Toc17314861)

[**b)** **Chức năng** 1](#_Toc17314862)

[**c)** **Web API** 1](#_Toc17314863)

[2) Restful APIs 2](#_Toc17314864)

[**a)** **Khái niệm:** 2](#_Toc17314865)

[**b)** **Chức năng:** 2](#_Toc17314866)

[**c)** **Phương thức Http mặc định trong web API** 2](#_Toc17314867)

[**d)** **Quy tắt** 3](#_Toc17314868)

[3) Các khái niệm khác 4](#_Toc17314869)

# API

## **Định nghĩa:**

* API (application programing interface) là “giao diện” giữa phần mềm với phần mềm. Là cách để các phần mềm như (các hệ điều hành, ứng dụng...) giao tiếp với nhau.API cung cấp các resource cho Application, tiếp nhận các request từ client chia các service để xử lí từng request. API thường được ứng dụng vào Web, hệ điều hành, các thư viện phần mềm hay framework.

## **Chức năng**

* Sử dụng chức năng: như đọc file, ghi file, định vị không cần đụng trực tiếp tới code của hệ điều hành.
* Trao đổi dữ liệu: Lấy thông tin từ các ứng dụng khác mà không cần trực tiếp truy cập vào database.
* Bảo mật: API giống như 1 vật trung gian, nó sẽ cho chúng ta biết là trong các nguồn resource thì có thể lấy được những thứ nào sau đó nó sẽ trả dữ liệu về chứ không được lấy trực tiếp=> Do đó, nó có tính bảo mật.
* Tái sử dụng: Ví dụ như API của facebook thì nó có thể tái sử dụng trên nhiều thành phần khác nhau. Cũng như web app và mobile app khi cần nó có thể tái sử dụng lại API của facebook người ta chỉ cần biết là cái API của fb nó trả ra các loại data gì. Mà không cần quan tâm bằng cách nào.

## **Web API**

* Web API là một phương thức dùng để cho phép các ứng dụng khác nhau có thể giao tiếp và trao đổi dữ liệu qua lại. Dữ liệu của web API thường trả về dưới dạng JSON hoặc XML thông qua giao thức HTTP hoặc HTTPS
* Cơ chế:
* Đầu tiên là xây dựng 1 URL API để bên Client có thể gửi Request thông qua URL đó
* Tại web server sẽ cung cấp các nội dung và ứng dụng nguồn sẽ thực hiện việc kiểm tra và xác thực nếu có và tìm các resource thích hợp để trả về kết quả.
* Server sẽ trả về kết quả theo định dạng JSON hoặc XML thông qua giao thức HTTP/HTTPs
* Tại Web client(ứng dụng web hoặc ứng dụng di động), dữ liệu JSON /XML sẽ được parse để lấy data. Sau khi có được data thì thực hiện tiếp các hoạt động như lưu trữ dữ liệu xuống DB, hiển thị dữ liệu

# Restful APIs

## **Khái niệm:**

* RestAPI: Thực hiện gọi hàm từ client->server thông qua các hàm và có thể lấy dữ liệu thông qua giao thức HTTP. Là 1 dạng chuyển đổi cấu trúc dữ liệu, là 1 phong cách kiến trúc cho việc thiết kế các ứng dụng có kết nối. Thay vì sủ dụng 1 URL cho việc xử lý thông tin người dùng Rest gửi 1 yêu cầu HTTP như GET, POST, PUT, DELETE đến 1 URL để xử lý dữ liệu.
* RestFulAPI là một tiêu chuẩn dùng trong việc thiết kế các thiết kế API cho các ứng dụng web để quản lý các resource. RestFul là 1 trong những kiểu thiết kế API được sử dụng phổ biến nhất ngày nay. Trọng tâm của 1 Rest là quy định cách sử dụng HTTP method (GET, POST, PUT, DELETE) và các định dạng các URL cho các ứng dụng web để quản lý các resource. HTTP sẽ trả dữ liệu về dưới dạng XML hoặc JSON, thường là JSON (JSON là 1 dạng text nhưng hầu như ngôn ngữ nào cũng có thể đọc được, thành ra có thể viết API bằng bất cứ ngôn ngữ gì. Về phái client có thể là mobile hoặc Web nó đều đọc được cả)

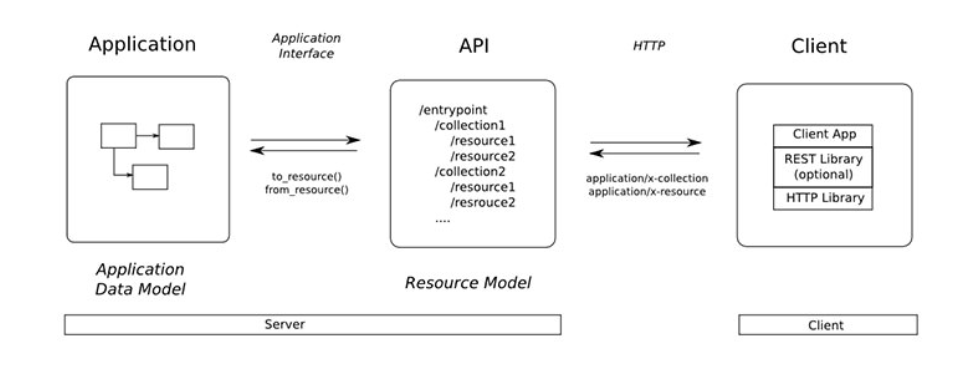
## **Chức năng:**

* Hệ thống hoạt động theo mô hình Client-server, trong đó server là tập hợp các service nhỏ lắng nghe request từ client, với từng request khác nhau thì có thể 1 hoặc nhiều service xử lý.
* Stateless(phi trạng thái): Đơn giản client và server không lưu trạng thái của nhau-> mỗi request lên server thì client phải đóng gói đầy đủ để thằng server hiểu được. Giúp bảo trì, kiểm thử, sữa chữa mà không cần CRUD trạng thái của client. Khuyết điểm, gia tăng lượng thông tin truyền tải giữa client và server, bất kì quản lý nào được yêu cầu phải diễn ra trên máy khách, không pải máy chủ.
* Cacheable: Client có thể lưu trữ các responses, các response đó phải chính xác và rõ ràng. Phải xác định là có thể lưu trong bộ nhớ cache hoặc không để ngăn khách hàng sử dụng lại dữ liệu cũ. Cache được quản lý tốt sẽ loại bỏ 1 phần hoặc hoàn toàn một số tương tác giữa client-server. Cải thiện hiệu suất.
* Uniform interface: Ràng buộc giao diện thống nhất cho phép mỗi phần phát triển độc lập.

## **Phương thức Http mặc định trong web API**

* Có 4 lệnh để truy cập RESTful APIs:
* GET: để truy vấn object
* POST: để tạo object mới
* PUT: để sửa đổi hoặc thay thế 1 object
* DELETE: loại bỏ 1 object
* Phương thức POST có thể thực hiện tất cả các action nhưng nó sẽ làm thay đổi giá trị sẽ dẫn đến trùng lặp dữ liệu.
* Thuật toán tìm phương thức HTTP của ASP.Net sẽ tuân theo các thứ tự sau:
* Nếu action có các attribute như [HttpGet] [HttpPost]… thì nó sẽ chấp nhận các phương thức này
* Nếu tên của action bắt đầu với các từ “Get”, “Post”, “Put”, “Delete”… thì nó cũng sẽ chấp nhận các phương thức này
* Còn nếu không thì phương thức mặc định là POST.
* Cần thận đặt tên vì nếu không server sẽ sử dụng phương thức POST dễ bị lỗi 405: Not Allowed

Mỗi API đều phải kết nối đến 1 server để trả dữ liệu. (viết code, format return data).



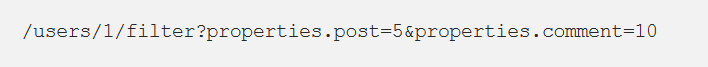
Ví dụ: muốn sử dụng 1 cái rest API của java thì muốn lấy 1 thông tin nào đó thì mình sẽ gửi 1 cái HTTP request tới cái trang theo đường dẫn URL. Sau đó server sẽ trả về dưới dạng JSON về cái thông tin yêu cầu.

## **Quy tắt**

Chât lượng thiết kế của API cần đáp ứng những yếu tố sau:

* Self-documenting (Nhìn vào API có thể đoán ra chúng dùng để làm gì).
* Flexible (Tính mở rộng cũng như tuỳ biến của API).
* Unifiled structure and atribute names (thống nhất về mặt cấu trúc cho resource cũng như cách đặt tên biến cho Attribute)
* Clear error message (Khi hệ thống xảy ra lỗi thì message phải rõ ràng và chi tiết để phục vụ cho kết quả fixbug)
* Mỗi resource chỉ được liên kết tối đa 2 đối tượng. VIệc liên kết với nhiều đối tượng sẽ trởi nên rối.

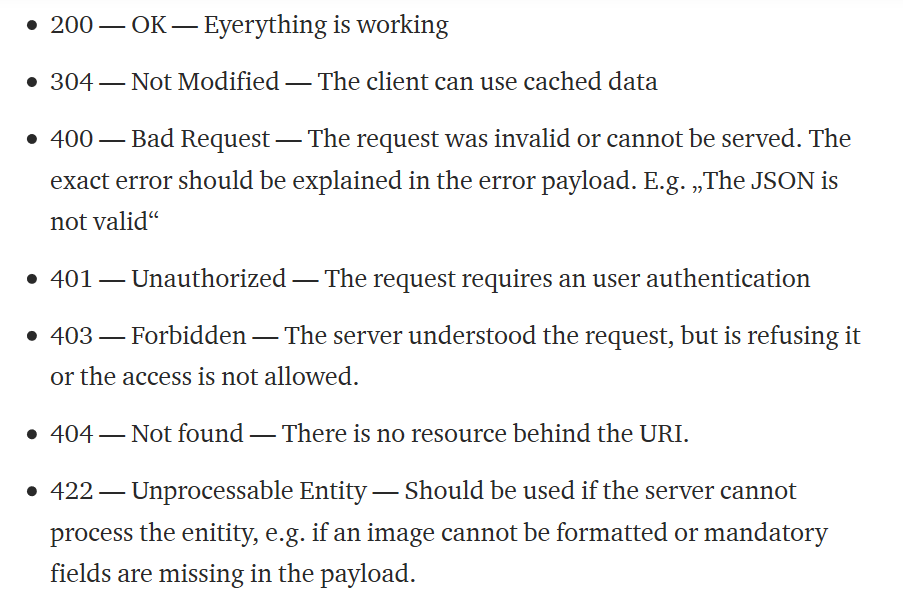
API truy vấn dữ liệu dạng filter:



Lấy 10 comment của post 5 có user là 1.

Xây dựng bộ điều kiện truy vấn đơn giản: neq: không bằng, gt: lớn hơn, gte: lớn hơn bằng, lt: nhơ hơn, lte: nhỏ hơn bằng, in: có trong, not\_in: không trong.

* Versioning: là điều kiện bắt buộc với mỗi resource, mỗi versioning thực hiện theo các quy tắt sau: bắt đầu bằng chữ “v” và kết thúc bằng 1 số nguyên dương, tránh dùng số thập phân. Version được đặt ở vị trí đầu tiên của resource
* Đăt tên cho các attribute: thống nhất đặt tên cho các attribute.
* Phân trang: offset 50 and limit 25 lấy 25 phần từ từ vị trí thứ 50.
* Tìm kiếm: attribute tên là “q”(query). EX: GET/search?q=fluffy+fur
* Lựa chọn 1 số filed trả về: Trong 1 vài trường hợp, client sẽ không cần đầy đủ thông tin của 1 object. EX: GET/users?fields=id,name,address
* Định dạng kiểu dữ liệu trả về. Đối với resource hỗ trợ nhiều định dạng kiểu dữ liệu trả về, HTTP-Header sẽ là nơi để xác định định dạng đó
* Content-type: Khai báo request format
* Accept: Khai báo response format
* HTTP status code và error message.



# Các khái niệm khác

* **Routing**: là cách mà API web khớp với URI. API hỗ trợ routing attribute. Routing attribute cho phép kiểm soát nhiều hơn các URI trong API web.
* **Attribute routing**: sử dụng các attribute để định nghĩa trực tiếp các rout trong controller action. Attribute rout có 3 ham số: URL pattern, tên và thứ tự.
* **FromBody và FromUri**: Sử dụng fromBody để web API nhận các kiểu đơn giản (int, bool, double), sử dụng FromUri để web API nhận các kiểu phức tạp từ URI. Nói cách khác FromBody sẽ xử lý trong body còn FromUri sẽ xử lý trên URI.