**Ôn tập môn Dịch vụ Web**

# [**Chủ đề 1: Web services**](https://legiacong.blogspot.com/2021/03/web-services-1-tong-quan.html)

- Tổng quan về Web services (hay web service) dịch ra tiếng Việt là dịch vụ web.

## **Định nghĩa**

- Là một dịch vụ, do thiết bị điện tử này cung cấp cho một thiết bị điện tử khác, quá trình trao đổi được thực hiện trên môi trường web (world wide web, WWW, Internet, HTTP).

- Một máy server trên mạng sẽ luôn lắng nghe các yêu cầu từ một cổng cụ thể, để cung cấp các tài nguyên web như HTML, JSON, XML, images và tạo ra các dịch vụ cho các ứng dụng có sử dụng web services.

Web services là một hệ thống phần mềm, được thiết kế để hỗ trợ khả năng tương tác giữa các ứng dụng trên các máy tính khác nhau, thông qua mạng Internet, giao diện chung và sự gắn kết của nó được mô tả bằng XML.

## **Cơ chế hoạt động**

- Một ứng dụng web services gồm hai thành phần là client (service consumer) và server (service provider). Chúng giao tiếp với nhau bằng giao thức HTTP.

- Client gửi yêu cầu thông qua HTTP request tới server.

- Server gửi kết quả trả về thông qua HTTP response.

- Có hai loại công nghệ web services được sử dụng phổ biến là SOAP và REST.

Dịch vụ web có thể là một lời gọi hàm, được viết dưới dạng một URL.

## **Ưu điểm**

* Cung cấp khả năng hoạt động rộng với các ứng dụng phần mềm khác nhau, chạy trên những nền tảng khác nhau.
* Sử dụng các giao thức và chuẩn mở
* Nâng cao khả năng tái sử dụng
* Thúc đẩy đầu tư các hệ thống phần mềm đã tồn tại
* Giúp tương tác linh hoạt giữa các thành phần trong hệ thống, dễ dàng phát triển các ứng dụng phân tán.
* Giúp tích hợp các hệ thống, giảm sự phức tạp, hạ giá thành hoạt động, phát triển hệ thống nhanh và tương tác hiệu quả với hệ thống của các doanh nghiệp khác
* **Nhược điểm**
* Nếu hệ thống cung cấp web services không hoạt động sẽ ảnh hưởng đến hoạt động của các máy client
* Lập trình viên phải học các chuẩn để có thể làm việc được với web services
* Phải quan tâm nhiều hơn đến vấn đề an toàn và bảo mật

# [**Chủ đề 2: Giao thức HTTP**](https://legiacong.blogspot.com/2021/03/web-services-2-hoat-ong-cua-web-services.html)**:** HyperText Transfer Protocol – Giao thức truyền tải siêu văn bản. Giao thức lớp ứng dụng – Giao thức là cách thức dùng để giao tiếp. Là nền tảng của truyền thông dữ liệu cho WWW.

Giao thức HTTP nằm ở tầng Application của mô hình OSI, và TCP/IP. Mô hình TCP/IP là mô hình của hệ thống mạng đang vận hành. Trong khi mô hình OSI là mô hình tham chiếu (mô hình lý thuyết).

## - Hoạt động của web services

+ Mô hình client – server: là mô hình hoạt động của mọi trang web/ứng dụng web.

* + Máy khách (client) gửi một yêu cầu (request) đến máy chủ (server), máy chủ sẽ xử lý và trả kết quả về cho máy khách.
  + Khi môi trường truyền đã được kết nối và sẵn sàng, thì client luôn là nơi khởi phát của ứng dụng. Client sẽ gửi một HTTP request tới server. Khi server nhận được request, nó sẽ xử lý và trả kết quả về cho client bằng một HTTP response.
  + Khi client và server sử dụng giao thức HTTP để giao tiếp, chúng gửi qua lại cho nhau các gói tin, gọi là gói HTTP. Gói HTTP gồm 3 phần: Header, Body và Trailer.
  + Client là thành phần chủ động. Server là thành phần bị động.
  + Server: có vai trò cung cấp dịch vụ, xử lý và trả về kết quả cho máy client.
  + Client: khởi phát yêu cầu, gửi yêu cầu tới server, nhận kết quả server trả về. Client có thể là trình duyệt, ứng dụng viết bằng python/javascript hoặc bất kỳ ứng dụng nào mà có phát sinh ra HTTP request.
  + Môi trường truyền thông tin: hạ tầng mạng (LAN, Internet), bao gồm cả phần cứng và phần mềm
  + Giao thức truyền thông tin: các chuẩn công nghệ giúp giao tiếp và truyền thông tin giữa server-client.

+ Giao thức: Tạo ra lỗi, phát hiện ra lỗi

Do ứng dụng web hoạt động theo mô hình client-server, nên để có trang web trên trình duyệt, cần trải qua các bước sau:

Bước 1: Người dùng nhập địa chỉ trang web (URL) vào thanh địa chỉ.

Bước 2: Trình duyệt dựa vào URL trong thanh địa chỉ, kết nối tới máy web server, gửi yêu cầu tới web server.

Bước 3: Web server xử lý yêu cầu, gửi trả kết quả về cho trình duyệt.

Bước 4: Trình duyệt thực thi mã HTML, CSS, JavaScript và hiển thị trang web ra màn hình.

Trong kiến trúc client-server, client và server giao tiếp với nhau bằng cách gửi thông điệp (gọi là HTTP message). HTTP message gồm hai loại là HTTP request message (thường được gọi tắt là HTTP request) và HTTP response message (thường được gọi tắt là HTTP response) :

## **HTTP request**

- HTTP request được client tạo ra, ví dụ khi người dùng nhập một URL vào thanh địa chỉ của trình duyệt và bấm phím Enter.

- Cấu trúc một HTTP request gồm bốn phần:

Request line

Request headers

Blank line

Request body

## **HTTP response**

- HTTP response là thông điệp do server gửi về cho client.

- Cấu trúc một HTTP response gồm bốn phần:

Status line

Response headers

Blank line

Response body

## **HTTPS**

- HTTPS viết tắt của HyperText Transfer Protocol Secure (giao thức truyền tải siêu văn bản bảo mật) là phiên bản an toàn của HTTP, giao thức mà nhờ đó dữ liệu được gửi giữa trình duyệt và trang web bạn đang kết nối.

- HTTPS thường sử dụng một trong hai giao thức bảo mật để mã hóa thông tin liên lạc - SSL (Secure Sockets Layer, tầng ổ bảo mật) hoặc TLS (Transport Layer Security, bảo mật tầng truyền tải). Cả hai giao thức TLS và SSL đều sử dụng hệ thống PKI (Public Key Infrastructure, hạ tầng khóa công khai) không đối xứng

* Dịch vụ web sử dụng giao thức HTTP làm phương tiện giao tiếp giữa client và server.

Mã trả về của server gồm một bộ-ba-số, trong đó số đầu tiên cho biết kết quả xử lý của server, hai số kế tiếp là thông tin bổ sung cho kết quả xử lý.

Ví dụ:

### **– 1xx: thông tin, có nghĩa là server đã nhận được yêu cầu, và tiến trình đang tiếp tục**

+ 100: cho biết máy chủ đã nhận được phần đầu tiên của yêu cầu và đang chờ đợi cho các phần còn lại

– **2xx: thành công, server đã nhận được yêu cầu, hiểu và xử lý thành công**

+ 200 OK: request đã được xử lý thành công

+ 202 Accepted: request đã được nhận, nhưng không có kết quả nào trả về, thông báo cho client tiếp tục chờ đợi

+ 204 No Content: request đã được xử lý nhưng không có thành phần nào được trả về

+ 205 Reset: giống như 204 nhưng mã này còn yêu cầu client reset lại document view

+ 206 Partial Content: server chỉ gửi về một phần dữ liệu, phụ thuộc vào giá trị range header của client đã gửi

**– 3xx: cần chuyển hướng, client phải thực hiện thêm thao tác để hoàn tất request**

+ 301 Moved Permanently: tài nguyên đã được chuyển hoàn toàn tới địa chỉ Location trong HTTP response

+ 303 See other: tài nguyên đã được chuyển tạm thời tới địa chỉ Location trong HTTP response

+ 304 Not Modified: tài nguyên không thay đổi từ lần cuối client request, nên client có thể sử dụng nội dung đã lưu trong cache

**– 4xx: lỗi bên client, cho biết HTTP request bị lỗi cú pháp hoặc không thực hiện được**

+ 400 Bad Request: request không đúng dạng, không đúng cú pháp

+ 401 Unauthorized: client chưa xác thực

+ 403 Forbidden: client không có quyền truy cập

+ 404 Not Found: không tìm thấy tài nguyên

+ 405 Method Not Allowed: phương thức không được server hỗ trợ

**– 5xx: lỗi bên server, cho biết server không thể xử lý được yêu cầu**

+ 500 Internal Server Error: có lỗi trong quá trình xử lý của server

+ 501 Not Implemented: server không hỗ trợ chức năng client yêu cầu

+ 503 Service Unavailable: server bị quá tải, hoặc bị lỗi xử lý

# **Chủ đề 3: Trải nghiệm làm một web services**

Lỗi 1:

Khai báo class ko có (), hàm mới có ()

Lỗi 2:

Chưa khai báo biến

Lỗi 3:

Đã dùng console để đổi mật khẩu của root,

Cách sửa: cập nhật mật khẩu mới đổi vào trong tập tin config.inc.php.

## **RESTful API**

RESTful API là một tiêu chuẩn dùng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) để tiện cho việc quản lý các resource. Nó chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video, hoặc dữ liệu động…), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP.

RESTful là một trong những kiểu thiết kế API được sử dụng phổ biến ngày nay để cho các ứng dụng (web, mobile…) khác nhau giao tiếp với nhau.

Chức năng quan trọng nhất của REST là quy định cách sử dụng các HTTP method (như GET, POST, PUT, DELETE…) và cách định dạng các URL cho ứng dụng web để quản các resource. RESTful không quy định logic code ứng dụng và không giới hạn bởi ngôn ngữ lập trình ứng dụng, bất kỳ ngôn ngữ hoặc framework nào cũng có thể sử dụng để thiết kế một RESTful API.

## **Web API**

Web API là một phương thức dùng để cho phép các ứng dụng khác nhau có thể giao tiếp, trao đổi dữ liệu qua lại. Dữ liệu được Web API trả lại thường ở dạng JSON hoặc XML thông qua giao thức HTTP hoặc HTTPS.

***Đặc điểm:***

* Tự động hóa sản phẩm
* Khả năng tích hợp linh động
* Cập nhật thông tin thời gian thực
* Có tiêu chuẩn chung để sử dụng

***Cách hoạt động***

1. Đầu tiên là xây dựng URL API để bên thứ ba có thể gửi request dữ liệu đến máy chủ cung cấp nội dung, dịch vụ thông qua giao thức HTTP hoặc HTTPS.

2. Tại web server cung cấp nội dung, các ứng dụng nguồn sẽ thực hiện kiểm tra xác thực nếu có và tìm đến tài nguyên thích hợp để tạo nội dung trả về kết quả.

3. Server trả về kết quả theo định dạng JSON hoặc XML thông qua giao thức HTTP/HTTPS.

4. Tại nơi yêu cầu ban đầu là 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 dữ liệu xuống Cơ sở dữ liệu, hiển thị dữ liệu…

# [**Chủ đề 4: XML, JSON, Ajax**](https://legiacong.blogspot.com/2023/11/web-services-6-xml-json-ajax.html)

## **XML: Extensible Markup Language**

* Là ngôn ngữ đánh dấu mở rộng, có chức năng truyền dữ liệu và mô tả nhiều loại dữ liệu khác nhau.
* Mục đích chính của XML là đơn giản hóa việc chia sẻ dữ liệu giữa các hệ thống khác nhau, đặc biệt là các hệ thống được kết nối với Internet.
* XML cung cấp một phương tiện dùng văn bản (text) để mô tả thông tin và áp dụng một cấu trúc kiểu cây cho thông tin đó.
* Các ngôn ngữ dựa trên XML (ví dụ: RDF, RSS, MathML, XHTML, SVG, GML, và cXML) được định nghĩa theo cách thông thường, cho phép các chương trình sửa đổi và kiểm tra hợp lệ bằng các ngôn ngữ này mà không cần có hiểu biết trước về hình thức của chúng.
* **Cú pháp**
* Một tập tin chứa tài liệu XML thường có phần mở rộng là .xml.
* Một tài liệu XML thường bắt đầu bằng dòng chỉ thị xử lý (processing instruction). Chỉ thị được đặt trong cặp dấu <? ?>. Nó cho biết phiên bản đặc tả XML được sử dụng (version=“1.0”), bản mã hóa văn bản được sử dụng (encoding=“utf-8”), và tài liệu XML có cần đến một tài liệu khác không (standalone=“yes”, nghĩa là không cần đến tài liệu khác).
* Dòng chú thích được đặt trong cặp dấu <!-- -->.
* **Thẻ XML**
* Thẻ (tag) là thành phần để tạo ra các phần tử XML, chúng xác định phạm vi một phần tử XML. Ngoài ra thẻ cũng có thể được sử dụng để chèn các chú thích (comment), khai báo các thiết lập và tạo ra các chỉ thị lệnh đặc biệt.
* Thẻ gồm hai loại: thẻ mở (start tag) và thẻ đóng (end tag).
* Thẻ mở được sử dụng để đánh dấu bắt đầu một phần tử (phần tử không rỗng). Ví dụ: <SinhVien>
* Thẻ đóng được sử dụng để đánh dấu kết thúc một phần tử (phần tử không rỗng). Ví dụ: </SinhVien>
* **Phần tử XML**
* Phần tử XML (XMLelement) được định nghĩa là các khối để tạo ra một tài liệu XML. Mỗi khối được bắt đầu bằng <thẻ mở>, tiếp theo là phần nội dung, và cuối cùng là thẻ đóng.

Ví dụ: <Ten>Teo</Ten>

* Phần tử có thể chứa văn bản (text), thuộc tính, phần tử khác, đối tượng media.
* Cú pháp của một phần tử:

<ten-phan-tu thuoc-tinh1=“gia-tri1” thuoc-tinh2=“gia-tri2”>nội dung</ten-phan-tu>

Ví dụ:

<Sach the-loai=“VanHoc”>

<TacGia>NVTeo</TacGia>

</Sach>

* Phần tử không có phần nội dung được gọi là phần tử rỗng (empty element). Cú pháp của phần tử rỗng:

<element></element>

hoặc,

<element />

* **Thuộc tính**
* Thuộc tính là thông tin được khai báo trong thẻ mở, bao gồm các thông tin bổ sung cho mỗi phần tử XML.

Cú pháp viết thuộc tính:

<element-name attribute1 attribute2 >content</element-name>

* Cú pháp của attribute: name = “value”
* Tên thuộc tính là duy nhất, giá trị (value) được bao bằng dấu nháy đơn (‘’) hoặc nháy kép (“”).
* **Lưu ý**
* Một tài liệu XML luôn phải có một phần tử gốc (root element) duy nhất, nó chứa tất cả các phần tử còn lại. Phần tử gốc thường cho biết nội dung chính của tài liệu XML.
* Tên của thẻ mở và thẻ đóng phải đồng nhất.
* Mỗi thẻ mở luôn có thẻ đóng đi kèm.
* Phần tử con phải nằm trọn trong phần tử cha của nó.
* Giá trị của thuộc tính được đặt trong cặp dấu nháy đơn (‘’) hoặc nháy kép (“”).
* **Cấu trúc cây**
* XML sử dụng cấu trúc cây để tổ chức dữ liệu. Một tài liệu XML luôn bắt đầu bằng một phần tử gốc (root element) hay nút gốc, phần tử gốc này sẽ chứa một hoặc nhiều phần tử con (hay nút con). Một tài liệu XML không được có hai nút gốc.
* Mỗi nút con lại có thể chứa một hoặc nhiều các nút con khác.
* Nút con cấp cuối cùng (nút lá) sẽ chứa dữ liệu.

## **JSON: JavaScript Object Notation**

* Là "sử dụng cú pháp/ký hiệu của đối tượng trong JavaScript để biểu diễn dữ liệu" ; là một định dạng dữ liệu đơn giản, dùng để trao đổi thông tin.
* **Đối tượng**
* Object trong Json được thể hiện bằng dấu ngoặc nhọn {}. Khái niệm Object trong Json cũng khá tương đồng với Object trong Javascript. Tuy nhiên, Object trong Json vẫn có những giới hạn như:
  + Key: phải luôn nằm trong dấu ngoặc kép, không được phép là biến số.
  + Value: Chỉ cho phép các kiểu dữ liệu cơ bản: numbers, String, Boolean, arrays, objects, null. Không cho phép function, date, undefined.
  + Không cho phép dấu phẩy cuối cùng như Object trong Javascript.
* **Đặc điểm**
* Bản thân JSON luôn ở dạng văn bản (text), là chuỗi, nên hay được gọi là chuỗi JSON
* Một đối tượng dữ liệu cần biểu diễn, được đặt trong cặp dấu ngoặc nhọn {}
* Không chứa phương thức (function)
* Dữ liệu được biểu diễn bằng các cặp name:value, trong đó name là tên của dữ liệu, value là giá trị của dữ liệu.
* name phải được đặt trong cặp dấu nháy kép (“”), lưu ý không được dùng cặp dấu nháy đơn (‘’).
* value có kiểu là string được đặt trong cặp dấu nháy kép (“”).
* Giữa name và value phải có dấu hai chấm (:).
* Sau mỗi cặp name:value có dấu chấm phẩy (,), trừ cặp name:value sau cùng.
* Các kiểu dữ liệu JSON có thể biểu diễn (chứa), hay cụ thể hơn là giá trị của value, gồm: string, number, boolean, null, array và object.
* Tập tin chứa dữ liệu JSON có phần mở rộng là .json.
* Mảng được đặt trong cặp dấu ngoặc vuông [].

## **AJAX: Asynchronous JavaScript And XML**

* Là một phương pháp, sử dụng JavaScript để trao đổi dữ liệu giữa trình duyệt và máy chủ (web server).
* Để trao đổi dữ liệu, AJAX sử dụng đối tượng XMLHttpRequest, đây là đối tượng có sẵn của JavaScript.
* Vì XMLHttpRequest là một đối tượng của JavaScript, nên nó sẽ có sẵn các thuộc tính và phương thức đi kèm, nhằm thực hiện các chức năng cần thiết.
* Hoạt động của AJAX:

Bước 1: Trình duyệt hoặc người dùng kích hoạt một sự kiện để bắt đầu một phiên làm việc của AJAX

Bước 2: Server nhận được request, nó sẽ thực hiện các xử lý và gửi dữ liệu trả về cho trình duyệt.

Bước 3: Trình duyệt nhận được dữ liệu từ server, thực hiện xử lý và hiển thị dữ liệu lên giao diện bằng JavaScript.

* Xử lý dữ liệu tại trình duyệt: tại bước 3, dữ liệu server trả về thuộc dạng JSON (dạng text), do vậy cần chuyển dữ liệu JSON về dạng đối tượng của JavaScript. Sử dụng hàm JSON.parse() của JavaScript.

# **Chủ đề 5: JavaScript nâng cao**

- Javascript là ngôn ngữ chạy một luồng duy nhất hay đơn luồng

- JavaScript nâng cao:

+ Lập trình bất đồng bộ (Asynchronous Programming) là một phương pháp lập trình cho phép các tác vụ được thực hiện độc lập và song song mà không cần chờ đợi các tác vụ khác hoàn thành trước đó.

* Ưu điểm

- Tăng hiệu suất cho các ứng dụng, tăng tốc độ thực thi của các tác vụ trong ứng dụng

- Tăng hiệu suất và giảm thời gian thực hiện cho các tác vụ lớn.

* Nhược điểm

- Khó khăn trong việc xử lý lỗi và cả gỡ lỗi.

+ Tại sao lại có nó: để đáp ứng nhu cầu của các ứng dụng web hiện đại, cải thiện hiệu suất và tương thích tốt với cấu trúc của trình duyệt.

+ Nó chạy như thế nào.

+ C++, C#: lập trình bất đồng bộ như thế nào, là gì

+ Cơ chế gọi hàm: gọi hàm như thế nào, ra làm sao, định nghĩa hàm

## **Callback**

Để lập trình bất đồng bộ sử dụng kĩ thuật callback, bạn cần thực hiện các việc sau:

– Bước 1: sử dụng kĩ thuật lập trình callback của JavaScript

– Bước 2: định nghĩa hàm, nhằm xử lý một tác vụ cụ thể, trong đó có gọi tới hàm callback cho trường hợp xử lý thành công và xử lý thất bại

– Bước 3: gọi hàm (đã định nghĩa ở Bước 2) để thực thi, trong đó định nghĩa thêm các xử lý dựa trên dữ liệu trả về của hàm callback trong trường hợp thành công và thất bại

## **Hiện tượng callback hell**

Callback hell là hiện tượng quá nhiều hàm callback lồng nhau, làm cho mã nguồn khó đọc, khó bảo trì.

## **Promise**

Là một cơ chế (hay một kĩ thuật lập trình) của JavaScript, nó giúp bạn lập trình bất đồng bộ mà không rơi vào tình trạng callback hell.

Khi khởi tạo đối tượng promise, bạn sẽ truyền vào cho nó một hàm callback gồm 2 tham số (resolve, reject) => {}; trong hàm callback này bạn sẽ viết mã nguồn xử lý nghiệp vụ.

Bản thân tham số resolve và reject lại là hai hàm, dùng để xử lý cho hai tình huống: “lời hứa thực hiện thành công” (resolve) và “không thực hiện được lời hứa” (reject). Vì vậy, về bản chất kỹ thuật promise cũng vẫn dựa trên kỹ thuật callback, chỉ khác nhau ở cách thức thực hiện.

Hàm resolve sẽ được gọi khi “lời hứa thực hiện thành công” hay nghiệp vụ xử lý thành công. Hàm reject sẽ được gọi khi “lời hứa không thực hiện được” hay nghiệp vụ xử lý thất bại.

Khi gọi hàm xử lý nghiệp vụ, kết quả trả về của promise sẽ được gửi tới phương thức .then() hoặc .catch(). Bạn sẽ viết các hàm để xử lý tùy theo kết quả trả về của promise.

## **Async/await**

Với async/await, dùng từ khóa async (viết tắt của asynchronous – bất đồng bộ) để khai báo một hàm sẽ chạy theo kiểu bất đồng bộ,

Trong trường hợp muốn thực hiện một hàm mà hàm trước đó phải thực hiện xong nghiệp vụ thì mới được thực thi, ta phải đặt trước đó từ khóa await để báo hiện rằng hàm này phải chạy xong, có kết quả trả về thì mới đến hàm hoặc các bước thực hiện tiếp theo.

**Một số lưu ý:**

– Kỹ thuật async/await hoạt động trên nền tảng promise

– Từ khóa async đặt trước một hàm sẽ làm cho hàm đó thực thi theo cơ chế bất đồng bộ

– Từ khóa await chỉ được sử dụng trong các hàm có từ khóa async

– Với async/await, cách viết đoạn chương trình chạy bất đồng bộ nhưng trông có vẻ giống như lập trình đồng bộ

– Từ khóa await đứng trước một promise (một hàm), nghĩa là chương trình sẽ chờ cho promise đó được xử lý xong và trả về giá trị, trước khi thực thi các lệnh kế tiếp

## **Fetch**

Fetch API là một thành phần của JavaScript, được sử dụng để trao đổi thông tin giữa client-server. Bạn có thể sử dụng Fetch API để tạo HTTP request, nhằm trao đổi dữ liệu giữa trình duyệt web và server.

Fetch API hoạt động dưới hình thức một promise; cho phép trao đổi thông tin theo kiểu bất đồng bộ. Với Fetch API, việc lập trình giao tiếp giữa trình duyệt và server được thực hiện đơn giản, rõ ràng và linh hoạt hơn so với XMLHttpRequest. Hầu hết các trình duyệt đều đã hỗ trợ Fetch API.

Fetch hoạt động dưới dạng một promise, nên nó sẽ tồn tại ở một trong 3 trạng thái: trạng thái chờ (pending), trạng thái thực hiện thành công, và trạng thái thực hiện thất bại.

Dữ liệu server trả về sẽ ở dạng luồng dữ liệu (stream), nên bạn có thể chuyển nó sang dạng JSON để xử lý. Lưu ý: phương thức response.json() không trả về dữ liệu dạng JSON, mà nó trả về một promise mới, phương thức resolve của promise này sẽ trả về dữ liệu dạng JSON.

# **Chủ đề 6: AWS.**

- Đọc/hiểu tài liệu tiếng Anh và triển khai

## **Điện toán đám mây (cloud computing) là gì?**

Điện toán đám mây là một mô hình cho phép truy cập vào mạng theo yêu cầu một cách thuận tiện, phổ biến vào một nhóm tài nguyên được thiết lập để chia sẻ (mạng, máy ảo, lưu trữ, ứng dụng và dịch vụ) mà được cung cấp và phát hành nhanh chóng với nỗ lực quản lý tối thiểu hoặc tương tác giữa các nhà cung cấp dịch vụ

## **Phương pháp triển khai điện toán đám mây?**

**Public**—Được quản lý bởi một tổ chức và dùng chung cho tất cả khách hàng.

**Private**—Ảo hóa và phân phối cơ sở hạ tầng CNTT cho riêng một tổ chức.

**Hybrid**—Sự kết hợp giữa 2 phương pháp public và private.

AWS sử dụng phương pháp triển khai điện toán đám mây theo kiểu public

## **Dịch vụ điện toán đám mây được chia thành các loại:**

**IaaS** (Infrastructure as a service): Cơ sở hạ tầng dưới dạng dịch vụ

Cung cấp các tài nguyên cơ bản như điện toán, lưu trữ, khả năng kết nối mạng, sử dụng máy ảo như là Amazon EC2, Google Compute Engine và Microsoft Azure

**PaaS** (Platform as a service): Nền tảng dưới dạng dịch vụ

Cung cấp nền tảng để triển khai các ứng dụng tùy chỉnh đến đám mây, chẳng hạn như AWS Elastic Beanstalk, Google App Engine, và Heroku.

**SaaS** (Software as a service): Phần mềm dưới dạng dịch vụ

Kết hợp cơ sở hạ tầng và phần mềm chạy trên đám mây, bao gồm các ứng dụng văn phòng như Amazon WorkSpaces, Google Apps for Work, and Microsoft Office 365

## **Có thể làm gì với AWS**

Có thể chạy tất cả các loại ứng dụng trên AWS bằng cách sử dụng một hoặc kết hợp các dịch vụ.