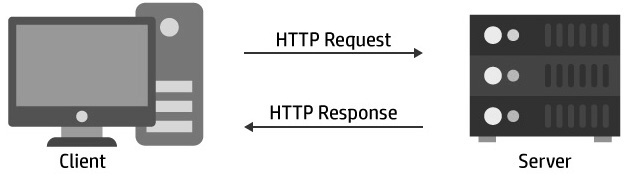
# Giao thức HTTP và cấu trúc cơ bản của HTTP Message

## HTTP là gì?

**HTTP (HyperText Transfer Protocol)** là *giao thức truyền tải siêu văn bản*, giao thức HTTP dựa trên giao thức TCP/IP nó cho phép lấy về các tài nguyên ví dụ như văn bản HTML, text, video, ảnh ...

**HTTP** là nền tảng dùng để trao đổi dữ liệu cho các ứng dụng Web với mô hình Client/Server. Các yêu cầu (request hay HTTP Request) được tạo bởi client (*thường là các Web browser như chrome, firefox ...*) được gửi đến server http, server sẽ gửi lại dữ liệu (response hay HTTP Response) để client nhận được. Các request, response là các message có cấu trúc đơn giản gọi chung nó là các HTTP Message.

Sự đơn giản của HTTP Message thể hiện ở chỗ nó có thể đọc được mà không nhất thiết phải có công cụ phân tích riêng biệt, nó giảm đi sự phức tạp khi phát triển, kiểm tra ứng dụng.



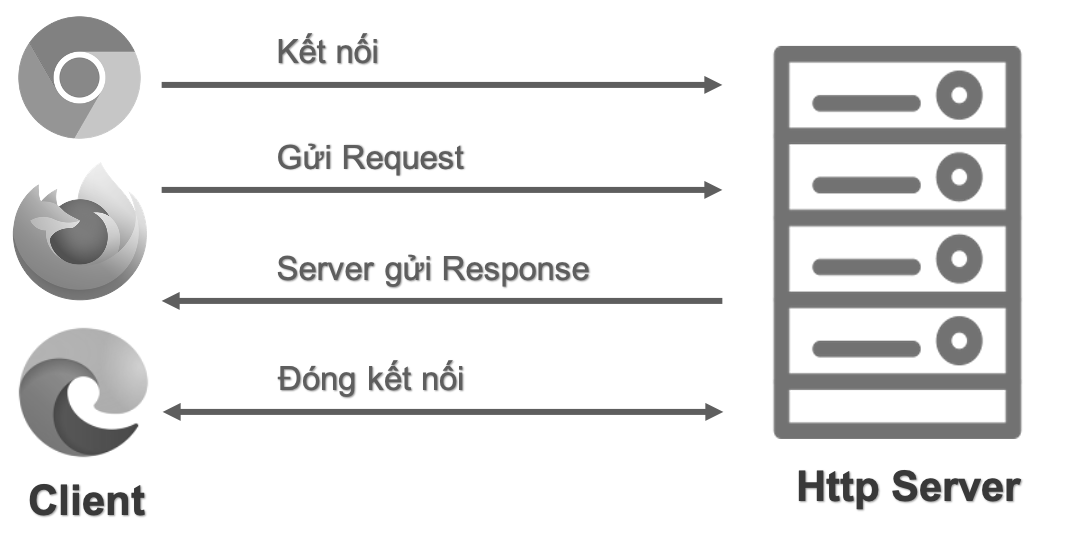
## Giao thức HTTP là phi trạng thái

**HTTP** là phi trạng thái (stateless), có nghĩa không có mối liên hệ nào giữa hai yêu cầu được gửi đi, dù nó là thực hiện trên cùng một kết nối - đến cùng một server. Ví dụ Web Browser gửi yêu cầu đến địa chỉ https://xuanthulab.net/ nhận về response, ngay sau đó nó gửi yêu cầu thứ 2 đến https://xuanthulab.net/abc/ lấy được response khác, thì hai truy vấn này không có mối liên hệ gì.

**Duy trì trạng thái với Cookie:**Rất nhiều trường hợp của ứng dụng cần biết trạng thái giữa các lần truy cập - chia sẻ trạng thái giữa những request của phiên làm việc, ví dụ ứng dụng đặt hàng - thêm sản phẩm vào giỏ, hay trang đăng nhập để người dùng đăng nhập - khi đã đăng nhập thì trạng thái đó phải được biết khi truy cập trang khác cùng phiên làm việc. Do Http là stateless nên để cung cấp khả năng này cần sử dụng tới HTTP Cookie, những dữ liệu nhằm duy trì trạng thái của phiên làm việc, nó được thêm vào phần header của HTTP Message (xem thêm [Sử dụng Cookie / Session trong PHP](https://xuanthulab.net/su-dung-session-va-cookie-trong-php.html))

## Sơ đồ làm việc HTTP truyền tải dữ liệu Client/Server

Khi client (*như web browser, các Http API trong ngôn ngữ lập trình ...*) kết nối đến server để gửi yêu cầu (request) và lấy về dữ liệu trả về (response) từ server thì nó trải qua các bước:



* **Bước 1: Mở kết nối TCP** - Kết nối TCP (*Giao thức HTTP dựa trên TPC*) trên địa chỉ xác định bởi URL (Uniform Resource Locator) và cổng thường là 80, kết nối này được dùng để gửi các yêu cầu (request) và nhận phản hồi (response). Client có thể mở ra kết nối TCP mới hoặc sử dụng kết nối đang có, thậm chí nó tạo ra nhiều kết nối TCP cùng lúc đến server.
* **Bước 2: Gửi HTTP Message (request)** - HTTP Message (request) chính là nội dung yêu cầu được client tạo ra và gửi đến server. HTTP Message có nội dung text (plain text) mà chúng ta có thể đọc được (người đọc được). Với phiên bản HTTP/2 thì nội dung HTTP Message được bao bọc trong các frame, nó làm cho người không đọc được một cách trực tiếp - tuy nhiên về mặt ý nghĩa nội dung không đổi so với HTTP/1.1

Ví dụ đây là nội dung một HTTP Message (request do client tạo ra)

GET /lap-trinh-php/ HTTP/1.0

Host: xuanthulab.net

Accept: text/html, application/xhtml+xml, application/xml;q=0.9, image/webp, \*/\*;q=0.8

* **Bước 3: Đọc HTTP Message nhận được từ servcer (response) -** Http Message (response) trả về từ server có cấu trúc tương tự Http Message (request), ví dụ đây là một response**:**

HTTP/1.0 200 OK

Date: Sat, 13 Feb 2021 09:59:11 GMT

Server: nginx

Accept-Ranges: bytes

Content-Length: 54545

Content-Type: text/html; charset=UTF-8

<!DOCTYPE **html**><**html** lang="vi">...

* **Bước 4: Đóng kết nối hoặc sử dụng lại cho các truy vấn khác**

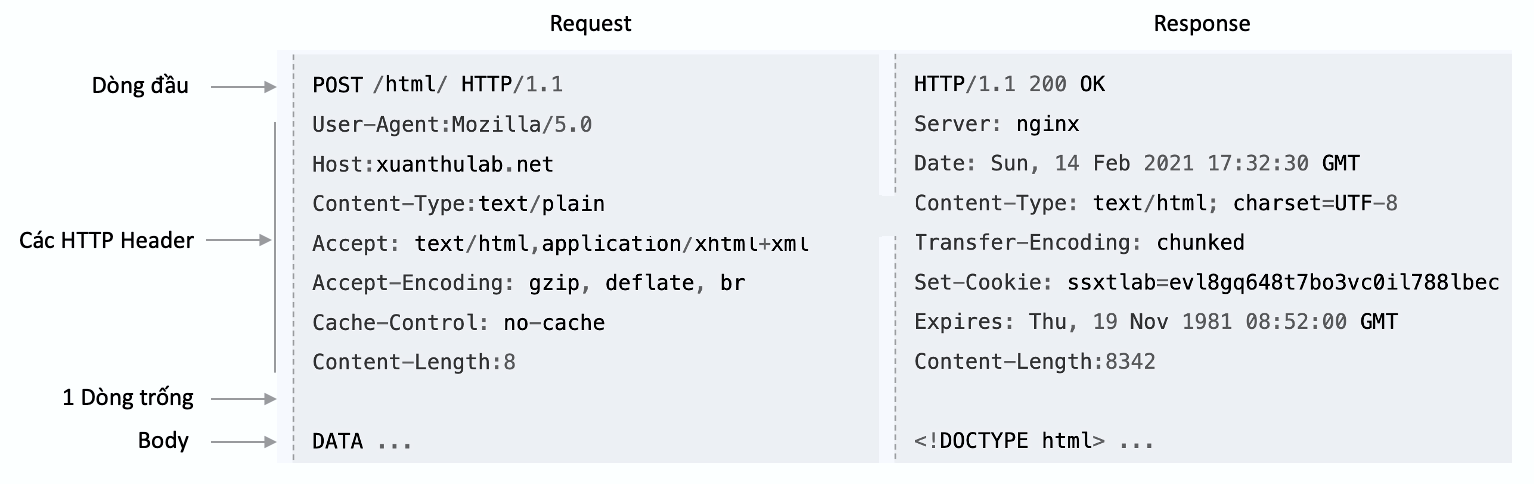
## HTTP Message

**HTTP Message** (*Thông điệp HTTP*) là các thông tin được trao đổi qua lại giữa client và server. Có hai kiểu HTTP Message: HTTP Request Message (Request) được gửi đi bởi client tới server - khi server nhận được nó biết phải thực hiện nhiệm vụ nào đó, HTTP Response Message (Response) là trả lời từ server về cho client.

**HTTP Message** trong phiên bản HTTP/1.1 có các thành phần dữ liệu trình bày trong định dạng text (plain text) mà người có thể đọc hiểu. Trong bản HTTP/2 thì các thành phần đưa vào định dạng nhị phân (binary) là các frame làm cho người không đọc được trực tiếp nữa. Tuy vậy ý nghĩa của các thành phần trong HTTP Message giữa hai phiên bản không đổi. Do vậy, ở đây trình bày về HTTP/1.1 nhưng vẫn có thể áp dụng cho HTTP/2

**HTTP/1.1** (chuẩn hóa năm 1997) là phiên bản giao thức HTTP mặc định trong các trình duyệt, server http ngày nay. **HTTP/2** (chuẩn hóa năm 2015) là phiên bản tiếp theo đang dần phổ biến (đặc biệt là tốc độ, an toàn, nén và ghép kênh), nếu server http của bạn hỗ trợ HTTP/2 thì hầu hết các browser sẽ tự động sử dụng được luôn.

**Request** và **Response** có những dữ liệu khác nhau nhưng chúng chia sẻ chung một cấu trúc, ví dụ dưới đây là một request và response (bạn có thể dùng Developer Tool trong trình duyệt Chrome hoặc ứng dụng Postman hoặc ứng dụng Fiddler để có thể đọc request/respone)



Cấu trúc chung của Request/Response có 4 thành phần

* **Dòng đầu tiên** (một dòng): Cho biết thông tin yêu cầu hoặc trạng thái (thành công hay lỗi) của response.
* **Phần header** (các dòng sau dòng đầu tiên trước một dòng trắng): Nó là một tập hợp các dòng chứa thông tin về HTTP Message, thông tin về phần body
* **Một dòng trắng** cho biết phần thông tin (dòng đầu và header) đã gửi hết
* **Phần body** chứa dữ liệu đính kèm với request (như HTML Form) hoặc nội dung văn bản đính kèm cùng response. Kích cỡ dữ liệu này (size) có được xác định bởi thông tin trong header.

Mặc dù có cấu trúc chung nhưng về chi chi tiết Request và Response có những đặc điểm riêng.

## HTTP Request Message

### Dòng đầu tiên:

Dòng này chứa thông tin để gửi tới server, dựa vào thông tin này mà server thực thi hành động phù hợp. Dòng này nó có chứa ba thông tin cách nhau bởi khoảng trắng, ví dụ như:

POST /html/ HTTP/1.1

**HTTP Method** (*Phương thức HTTP*) là thành phần thứ nhất (ví dụ trên là POST), nó có giá trị như POST, GET, PUT. Nó cho biết yêu cầu cần được thực hiện trên server đối với một tài nguyên nào đó, cụ thể như các phương thức:

* **GET** cho biết Request chỉ cần lấy dữ liệu (tài nguyên).
* **POST** được sử dụng để cho biết Request là chuyển dữ liệu lên server (thương HTTP Message trường hợp này có kèm dữ liệu như HTML Form ...)
* **PUT** ghi đè (thay thế) tài nguyên nào đó bằng dữ liệu trong Request.
* **DELETE** xóa tài nguyên

**Địa chỉ URL** - là thành phần thứ hai (ví dụ trên là /html/). Địa chỉ tài nguyên truy vấn, có thể là URL tương đối - tuyệt đối (kể cả cổng, nếu có cổng thì viết cổng sau ký hiệu :)

**HTTP Version** Thành phần thứ ba cho biết phiên bản HTTP (thường là HTTP/1.1)

### Header của Request:

Các header có cấu trúc đó là một chuỗi là tên header tiếp theo là dấu : và giá trị cho header. Mỗi header được viết trên một dòng. Có rất nhiều loại header (đã chuẩn hóa hoặc header do bạn tự đặt), có thể tham khảo danh sách các header tại [HTTP Request Header](https://en.wikipedia.org/wiki/List_of_HTTP_header_fields#Request_fields), ví dụ vài header như:

**Host** là header chỉ ra host (domain, IP) và cổng của server mà Request gửi đến. Nếu không chỉ rõ port thì mặc định là 80 với http và 443 với https. Ví dụ:

**Host: xuanthulab.net**

**Accept** trong Request cho biết kiểu nội dung trả về mà client có thể hiểu. (các kiểu cách nhau bởi , có độ ưu tiên mặc định 1, nếu muốn xác định độ ưu tiên cho kiểu nào thì cho thêm ;q=value) Ví dụ:

Accept: text/html, application/xhtml+xml, application/xml;q=0.9, image/webp, \*/\*;q=0.8

**User-Agent** là header của Request cho phép server xác định được ứng dụng nào, hệ điều hành gì gửi Request. Ví dụ:

User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:12.0) Gecko/20100101 Firefox/12.0

**Content-Length** cho biết độ dài (byte) của phần body (nếu có đính kèm trong request). Ví dụ:

**Content**-Length: 12345

**Content-Type** trong Request, header này cho server biết kiểu dữ liệu được đính kèm trong phần body. Ví dụ:

Content-Type: application/x-www-form-urlencoded

**Cookie** header này gửi giá trị cookie lấy được từ truy vấn trước để server phục hồi phiên làm việc. Có thể gửi nhiều cookie cách nhau bởi ;

**Cookie: name=value; name2=value2; name3=value3**

Trong ứng dụng PHP mặc định tên Cookie là PHPSESSID, vậy tại server php (backend) đọc được giá trị này - nó sẽ phục hồi Session.

### Body của Request:

Chứa dữ liệu của Request (dữ liệu này có kiểu xác định ở header Content-Type và độ dài Content-Length), không phải mọi Request đều có body, dữ liệu body thường là HTML Form hay JSON.

## HTTP Response Message

### Dòng đầu tiên:

Còn gọi là dòng trạng thái, nó chứa ba mẩu tin gồm:

* + Phiên bản HTTP, ví dụ HTTP/1.1
  + Mã trả về như 200, 404, 301, 302 ...
  + Đoạn text ngắn giải thích mã ví dụ OK, Not Found, Moved Permanently ...

Đây là một dòng trạng thái

HTTP/1.1 301 Moved Permanently

### Header của Response:

Tương tự như header của Request, các header thông dụng có thể tham khảo tại: [Một số header của Response](https://en.wikipedia.org/wiki/List_of_HTTP_header_fields#Response_fields), ví dụ như:

**Date** chứa thông tin ngày tháng thông tin (response) được phát sinh

Date: Wed, 21 Oct 2015 07:28:00 GMT

**Set-Cookie** header này chứa cookie của server gửi cho client. Client lưu lại để gửi lại cho server để phục hồi phiên làm việc.

**Set**-**Cookie**: **CookieID**=94583405kfds; **Max**-**Age**=3600;

Một số giá trị thiết lập như:

* Max-Age=number thời gian hết hạn cookie
* HttpOnly không cho JS truy cập Cookie

Có những header giống header của Request như: Content-Type, Content-Length ...

### Body của Response:

Chứa dữ liệu của Respone (dữ liệu này có kiểu xác định ở header Content-Type và độ dài Content-Length), không phải mọi Response đều có body.

# Lớp Uri Dns Ping và các lớp về Networking trong lập trình C# NET Core

Để kiểm tra 1 response của 1 trang web:

* Chúng ta có thể xem trong dev tool
* Hoặc chúng ta có thể xem trong cmd bằng câu lệnh: **curl -i <url>**

## Một số namespace, class về Networking

### Namespace:

**System.Net**  
**System.Net.Mail**  
**System.Net.NetworkInformation**  
**System.Net.Http**

### Class:

* Làm việc với máy chủ Dns, Uri, địa chỉ mạng có các lớp Dns, Uri, Cookie, IPAddress ...
* Làm việc với FTP Server có các lớp FtpStatusCode, FtpWebRequest, FtpWebResponse ...
* Làm việc với giao thức HTTP (máy chủ web) có các lớp như: HttpStatusCode, HttpWebRequest, HttpWebResponse, HttpClient, HttpMethod, HttpRequestMessage, HttpResponseMessage
* Làm việc với máy chủ SMTP gửi email: SmtpClient, MailMessage, MailAddress, MailAddress
* Làm việc với giao thức mạng: IPStatus, NetworkChange, Ping, TcpStatistics ...

## URI là gì?

**URI** *(Uniform Resource Identifier)* là một chuỗi ký tự để định vị tài nguyên (tài nguyên đó ở đâu, là gì trên internet). Các tài nguyên có để là tài liệu (document - text, html ...), hình ảnh (image) ..., **tài nguyên** chính là mục tiêu của HTTP Request.

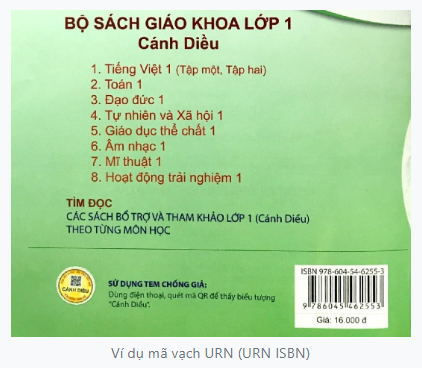
URI xác định tài nguyên bằng cách định vị bằng vị trí (locator - địa chỉ) hoặc bằng tên, hoặc cả hai. Từ đó URI phân ra làm hai loại là URL và URN. *(Có thể nói URL là URI, URN cũng là URI)*

## URN - Uniform Resource Name

**URN** (Uniform Resource Name) - *Định danh tài nguyên thống nhất*: là một loại URI, xác định tài nguyên thông qua tên hợp lệ, duy nhất (các tài nguyên ở trường hợp này có thể là bất kỳ thứ gì). Cấu trúc của URN thường có dạng: URN:NID:NSS ví dụ như URN:ISBN:8934974170617 trong đó:

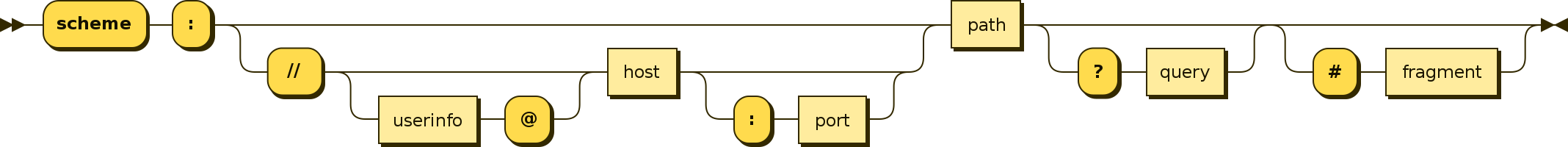
* URN chuỗi cho biết đây là URN
* NID không gian tên (namespace) do IANA cấp (Internet Assigned Numbers Authority). Ví dụ ISBN (Mã số sách quốc tế) là một namespace. Các namespace khác có thể xem tại [urn-namespaces](https://www.iana.org/assignments/urn-namespaces/urn-namespaces.xhtml)
* NSS chuỗi xác định đối tượng cụ thể

Ví dụ URN:ISBN:978-0531230145 là một URN, nó cho phép xác định cuốn sách là [President Donald Trump (A True Book)](https://isbnsearch.org/isbn/9780531230145). Còn URN URN:ISBN:8934974170617 tra cứu lưu chiểu thì đó là tiểu thuyết **CON CHIM XANH BIẾC BAY VỀ** của Nguyễn Nhật Ánh. Chú ý mỗi quốc gia có một cơ quan ISBN riêng, phải truy cập dữ liệu lưu chiểu của quốc gia đó mới tra được thông tin cuốn sách.



## URL - Uniform Resource Locator - Địa chỉ URL

**URL** (*định vị tài nguyên thống nhất*), nó là **địa chỉ xác định tài nguyên trên internet**, nó là một loại URI được dùng trong các siêu văn bản (Hypertext - HTML) và [giao thức HTTP](https://xuanthulab.net/giao-thuc-http-va-cau-truc-co-ban-cua-http-message.html), nó được sử dụng bởi các browser (client) để lấy về hay cập nhật tài nguyên trên web. URL là địa chỉ xác định tài nguyên (trang HTML, file JS, file CSS, file ảnh ....) duy nhất trên Web.





Nó có các thành phần:

* **scheme** ví dụ https://, http://, ftp:// ... cho biết giao thức sử dụng để yêu cầu tài nguyên
* **host** hoặc domain (ví dụ xuanthulab.net) có thể có port ví dụxuanthulab.net:80 ... không cần chỉ ra nếu sử dụng cổng tiêu chuẩn (cổng 80 với http và 443 với https)
* **path** (ví dụ /path/to/page/) đường dẫn trên server dẫn tới tài nguyên, hiện nay không hẳn là một đường dẫn thực mà có thể là một logic ánh xạ bởi web server
* **query** là chuỗi truy vấn, nó chứa các tham số ví dụ ?a=1&b=price, bắt đầu chuỗi query là dấu ? mỗi tham số thường gồm key=value, các tham số cách nhau bởi &
* **fragment** (ví dụ #section), trỏ đến một phần củ thể trong tài nguyên, ví dụ một vị trí nào đó trong văn bản HTML.

**URL** được sử dụng để nhập vào thanh địa chỉ trình duyệt, để lấy về tài nguyên. Nó sử dụng trong các [thẻ <a> HTML](https://xuanthulab.net/the-a-tao-lien-ket-html.html) để tạo ra các liên kết (đường link), thẻ [<img> trong HTML](https://xuanthulab.net/the-img-chen-anh-vao-html.html) để chèn ảnh ... URL trình bày đầy đủ ở trên là **đường dẫn tuyệt đối** (địa chỉ tuyệt đối), có đầy đủ thành phần scheme, host. Ngoài ra trong các trang HTML cũng hay sử dụng địa chỉ URL tương đối, đường dẫn tương đối

## Địa chỉ URL tương đối, đường dẫn tương đối

Ngoài địa chỉ URL tuyệt đối, còn có thể sử dụng địa chỉ URL tương đối - địa chỉ này không cần có scheme, host, nó cho biết tài nguyên nằm cùng trên một host của tài nguyên ban đầu, nó thường bắt đầu bằng /, ví dụ:

Địa chỉ tuyệt đối ban đầu là: **https://xuanthulab.net/pathA/pathB/fileA.html** (URL trên thanh địa chỉ trình duyệt), nếu vậy bên trong nó có thể có các địa chỉ tương đối, địa chỉ này được phân tích theo URL gốc chứa nó trên để tìm ra dạng tuyệt đối của nó. Cụ thể:

|  |  |
| --- | --- |
| **Địa chỉ (URL) tương đối** | **Địa chỉ tuyệt đối tương ứng** |
| /path | Nếu địa chỉ tương đối bắt đầu bằng **/** thì nó cùng HOST, SCHEME với URL ban đầu (tài nguyên nằm trong thư mục gốc - https://xuanthulab.net): Địa chỉ trang (tài nguyên fileA.html) chứa URL tương đối là https://xuanthulab.net/pathA/pathB/fileA.html vậy thư mục gốc của nó là https://xuanthulab.net/  https://xuanthulab.net/pathA/pathB/fileA.html (trang gốc chứa URL tương đối)  /abc/x1.png => https://xuanthulab.net/abc/x1.png  /abc-1.html => https://xuanthulab.net/abc-1.html  /a/b/c/d/e.gif => https://xuanthulab.net/a/b/c/d/e.gif |
| path | Địa chỉ tương đối không bắt đầu bởi **/** thì tài nguyên cùng thư mục chứa tài nguyên gốc. Địa chỉ trang (fileA.html) chứa URL tương đối là https://xuanthulab.net/pathA/pathB/fileA.html vậy thư mục chứa fileA.html là https://xuanthulab.net/pathA/pathB/  https://xuanthulab.net/pathA/pathB/fileA.html (trang gốc chứa URL tương đối)  xxx/zzz.jpg => https://xuanthulab.net/pathA/pathB/xxx/zzz.jpg  abc-1.html => https://xuanthulab.net/pathA/pathB/abc-1.html  a/b/c/d/e.html => https://xuanthulab.net/pathA/pathB/a/b/c/d/e.html |
| ../path | Kí hiệu ../ cho biết địa chỉ bắt đầu lùi một cấp thư mục với tài nguyên chứa nó Địa chỉ ban đầu là https://xuanthulab.net/pathA/pathB/fileA.html, thư mục chứa tài nguyên ban đầu (fileA.html) sẽ là https://xuanthulab.net/pathA/pathB/, lùi một cấp sẽ là https://xuanthulab.net/pathA/  https://xuanthulab.net/pathA/pathB/fileA.html (trang gốc chứa URL tương đối)  ../xxx.html => https://xuanthulab.net/pathA/xxx.html  ../abc-1.jpg => https://xuanthulab.net/pathA/abc-1.jpg  Tương tự có thể trở về gốc 2 cấp : ../../path |

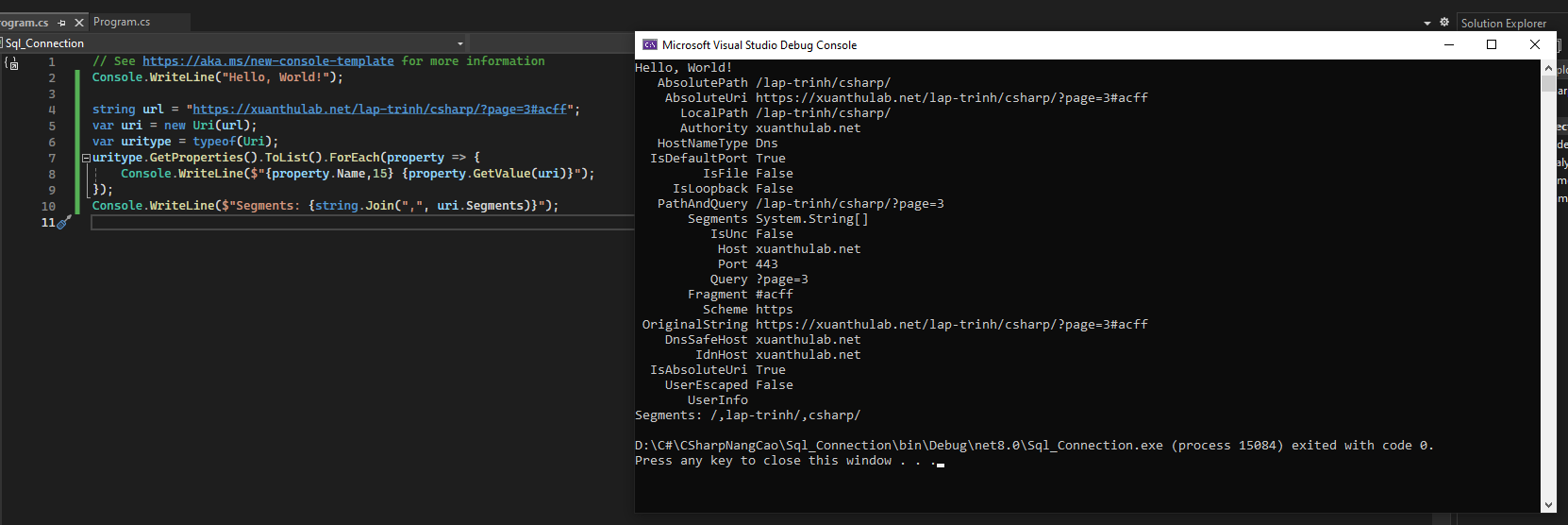
## Địa chỉ URL tới tài nguyên cục bộ

Nguyên tắc trên có thể áp dụng cho viết địa chỉ tới file trên máy tính cục bộ. Ví dụ trên Windows, có file tài nguyên ở đường dẫn: C:\pathA\pathB\fileA.html, thì địa chỉ URL dạng tuyệt đối là: file:///c:/pathA/pathB/fileA.html, các địa chỉ tương đối trong nó áp dụng như trên.

Với Linux, macOS: có file tài nguyên ở đường dẫn: /pathA/pathB/fileA.html, thì địa chỉ URL dạng tuyệt đối là: file:///pathA/pathB/fileA.html, các địa chỉ tương đối trong nó áp dụng như trên.

## Lớp URI

**System.Uri** là lớp biểu diễn về địa chỉ URI (URL) (xem thêm [Tìm hiểu URI, URL](https://xuanthulab.net/cau-truc-uri-urn-va-url.html)), nó giúp cho nhanh chóng lấy thông tin các thành phần của URL như host, path, query ... Đối tượng Uri còn sử dụng trong tham số để thực hiện các truy vấn HTTP Request ở các phần sau.



## Lớp tĩnh Dns và lớp IPHostEntry

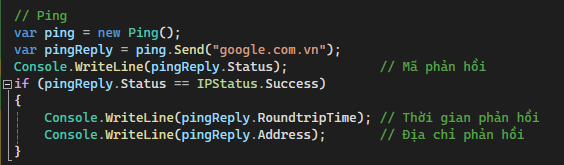
Lớp **Dns** *(System.Net.Dns)* cung cấp các phương thức tĩnh để lấy thông tin về host (địa chỉ website, server cung cấp các dịch vụ mạng) từ hệ thống phân giải tên miền (Dns). Các thông tin truy vấn được nó trả về một đối tượng giao diện IPHostEntry

**DNS** là hệ thống phân giải tên miền, giúp cho các trình client (như các trình duyệt) truy vấn để chuyển đổi một tên miền (như xuanthulab.net) sang địa chỉ IP vật lý tương ứng của tên miên đó. Sau đó địa chỉ IP này được dùng để kết nối client/server. Dữ liệu DNS được lưu trữ và phục vụ truy vấn từ các Server DNS được vận hành bởi các nhà cung cấp dịch vụ và các tổ chức

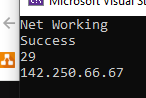
|  |  |
| --- | --- |
| Một số phương thức của lớp **Dns** | |
| GetHostName() | Lấy hostname của máy local |
| GetHostEntry(String) GetHostEntry(IPAddress) | Phân giải host hoặc IP thành đối tượng IPHostEntry. Đối tượng kiểu IPHostEntry nó chứa thông tin địa chỉ về host. |
| **IPHostEntry** có các thuộc tính để lấy thông tin về host như | |
| HostName | Chuỗi chứa hostname của Server |
| AddressList | Mảng các phần tử kiểu IPAddress chứa các địa chỉ IP |

## Lớp Ping

Lớp Ping (System.Net.NetworkInformation.Ping), lớp này cho phép ứng dụng xác định một máy từ xa (như server, máy trong mạng ...) có phản hồi không.



Kết quả:

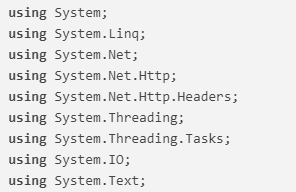


# Sử dụng HttpClient trong C# tạo các truy vấn HTTP

Sử dụng lớp HttpClient để thực hiện các truy vấn HTTP, sử dụng phương thức GET và POST với phương thức SendAsync, thiết lập các loại Content trong HTTP Request như FormUrlEncodedContent, StringContent, MultipartFormDataContent

## Giới thiệu HttpClient

Lớp HttpClient được sử dụng để gửi truy vấn HTTP (Http Request Message - Request) và nhận phản hồi Response (Http Response Message) từ các truy vấn đó. Lớp này thuộc namespace System.Net.Http, namespace này chứa các lớp giúp tạo ra sự liên lạc giữa client và server. Để làm việc với HttpClient dùng những namespace sau:



## Tạo truy vấn GET bất đồng bộ với HttpClient

Để tạo ra truy vấn GET tới một địa chỉ URL, thực hiện phương thức GetAsync(url), đây là phương thức async khi kết thúc nó trả về đối tượng HttpResponseMessage. Từ đối tượng này ta sẽ biết kết quả truy vấn, và ta có thể đọc được dữ liệu tải về.

// Khởi tạo http client

**using** **var** httpClient = **new** HttpClient();

// Thiết lập các Header nếu cần

httpClient.DefaultRequestHeaders.Add("Accept", "text/html,application/xhtml+xml+json");

// Thực hiện truy vấn GET

HttpResponseMessage response = **await** httpClient.GetAsync(url);

Khi có đối tượng lớp HttpResponseMessage có thể thực hiện các tác vụ đọc dữ liệu, tham khảo một số thuộc tính và phương thức:

Một số phương thức, thuộc tính lớp HttpResponseMessage:

|  |  |
| --- | --- |
| EnsureSuccessStatusCode() | Phát sinh Exception nếu truy vấn có mã trả về không thành công (khi thuộc tính IsSuccessStatusCode là false) |
| IsSuccessStatusCode | Nhận true khi mã trả về thành công, ví dụ mã trạng thái StatusCode nhận giá trị 200 |
| StatusCode | Thuộc tính có kiểu enum HttpStatusCode cho biết mã trạng thái của kết quả *(Như 301 - MovedPermanently, 200 - OK, 404 - NotFound ...)*, để chuyển sang số nguyên (int)HttpStatusCode |
| ReasonPhrase | Đoạn text, mô tả thông tin cho mã trạng thái như OK, MovedPermanently ... |
| Headers | Thuộc tính kiểu HttpResponseHeaders chứa tập hợp các header (xem thêm [các header của HTTP Message](https://xuanthulab.net/giao-thuc-http-va-cau-truc-co-ban-cua-http-message.html#httpmessage)) của Response. HttpResponseHeaders kế thừa từ HttpHeaders là tập hợp mà mỗi phần tử có kiểu KeyValuePair<String,IEnumerable<String>> (biểu diễn một header httpd - key là tên header và value là tập hợp các chuỗi giá trị của header)  Ví dụ xây dựng phương thực để hiện thị tất cả các header  **///** In ra thông tin các Header của HTTP Response  **public** **static** **void** **ShowHeaders**(HttpHeaders headers)  {  Console.WriteLine("CÁC HEADER:");  **foreach** (**var** header **in** headers)  {  **foreach** (**var** **value** **in** header.Value)  {  Console.WriteLine($"{header.Key,25} : {**value**}");  }  }  Console.WriteLine();  }  Ngoài ra nó còn có một số thuộc tính là các header thông dụng HTTP Respone như Age, Etag, Server ... mà nó đã phân tích các chuỗi header trả về thành các đối tượng tương ứng như TimeSpan, EntityTagHeaderValue, HttpHeaderValueCollection ... xem chi tiết các thuộc tính này tại [các header response](https://docs.microsoft.com/en-us/dotnet/api/system.net.http.headers.httpresponseheaders?view=net-5.0#properties) |
| Content | Thuộc tính kiểu HttpContent, trong đó chứa nội dung (content) và các header liên quan đến content nếu có (Headers như Content-Type, Content-Length ...), một số phương thức để lấy content:   * **ReadAsStringAsync** đọc nội dung (content) HTTP trả về chuỗi (có encoding). Đối với phương thức này, một số server trả về mã CharSet mà .NET Core không nhận ra sẽ gây lỗi, để encoding được cho trường hợp này thì trước khi thi hành thiết lập cho nó:   Content.Headers.ContentType.CharSet = @"utf-8"; //ISO-8859-1   * **ReadAsStreamAsync** trả về đối tượng Stream, từ stream dùng kỹ thuật đọc luồng để đọc nội dung: xem thêm [Đọc streams](https://xuanthulab.net/stream-trong-c-lam-viec-voi-filestream-lap-trinh-c-sharp.html#copy) Dữ liệu đọc được là các byte, bạn có thể lưu chúng ra file hoặc convert thành chuỗi * ReadAsByteArrayAsync đọc trả về mảng byte, từ mảng byte này có thể lưu nó ra file. Hoặc encoding nó thành chuỗi, ví dụ encoding utf-8   string content = System.Text.Encoding.UTF8.GetString(byteArray); |