**PROG04: Lập trình mạng/HTTP client**

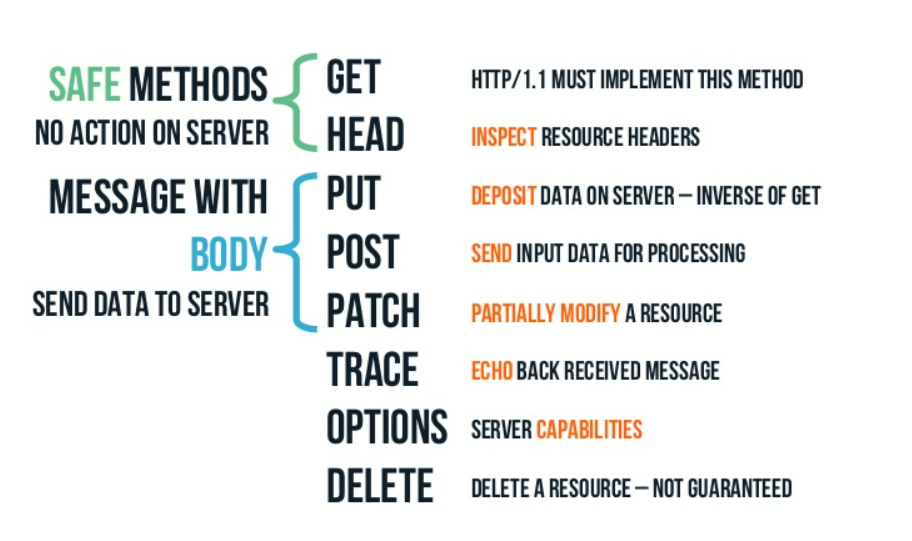
**Lý thuyết:**

**1. Tìm hiểu giao thức HTTP**

**- HTTP (HyperText Transfer Protocol - Giao thức truyền tải siêu văn bản)** là một trong các giao thức chuẩn về mạng Internet, được dùng để liên hệ thông tin giữa Máy cung cấp dịch vụ (Web server) và Máy sử dụng dịch vụ (Web client), là giao thức Client/Server dùng cho World Wide Web – WWW

HTTP là một giao thức ứng dụng của bộ giao thức TCP/IP (các giao thức nền tảng cho Internet).

**- Các method phổ biến:**

****

**+ GET:** là phương thức được Client gửi dữ liệu lên Server thông qua đường dẫn URL nằm trên thanh địa chỉ của Browser. Server sẽ nhận đường dẫn đó và phân tích trả về kết quả. Hơn nữa, nó là một phương thức được sử dụng phổ biến mà không cần có request body.

Một số đặc điểm chính:

* Giới hạn độ dài của các giá trị là 255 ký tự
* Chỉ hỗ trợ các dữ liệu kiểu String
* Có thể lưu vào bộ nhớ cache
* Các tham số truyền vào được lưu trong lịch sử trình duyệt
* Có thể được bookmark (đánh dấu rồi xem lại sau) do được lưu trong lịch sử trình duyệt

**+ POST:** Phương thức Post là phương thức gửi dữ liệu đến server giúp người dùng có thể thêm mới dữ liệu hoặc cập nhật dữ liệu đã có vào database. Gửi thông tin cần thêm hoặc cập nhật trong phần body request.

Một số đặc điểm chính:

* Dữ liệu cần thêm hoặc cập nhật không được hiển thị trong URL của trình duyệt
* Dữ liệu không được lưu trong lịch sử trình duyệt
* Không có hạn chế về độ dài của dữ liệu
* Hỗ trợ nhiều kiểu dữ liệu: String, binary, integers,….

**+ PUT:** Cách hoạt động tương tự như Post nhưng nó chỉ được sử dụng để cập nhật dữ liệu đã có trong database. Khi sử dụng PUT, phải sửa toàn bộ dữ liệu của một đối tượng.

**+ PATCH:** PATCH được sử dụng khi phải cập nhật một phần dữ liệu của đối tượng

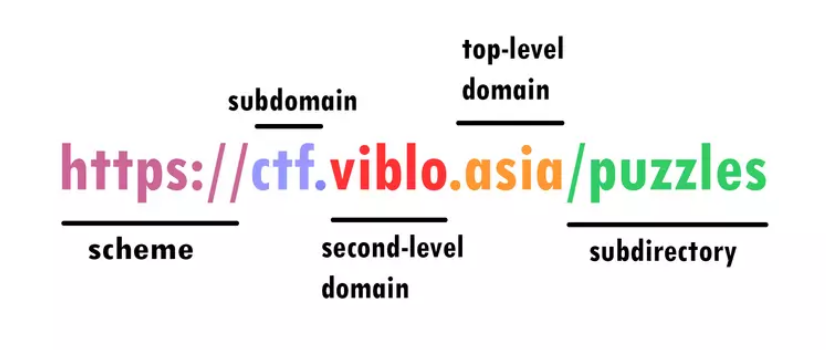
**+ DELETE:** Khi sử dụng phương thức DELETE sẽ xoá các dữ liệu của server về tài nguyên thông qua URI. Cũng giống như GET, phương thức này không có body request.

**+ HEAD:** HEAD gần giống giống với lại GET, tuy nhiên nó không có response body.

Truy vấn HEAD hữu ích khi chúng ta sử dụng nó để kiểm tra API có hoạt động không do không có response body nên thời gian phản hồi nhanh hơn so với phương thức Get. Và thường được sử dụng để kiếm tra trước khi download file do cứ gọi đến api dowload sẽ download file nên thêm phương thức head vào nó kiểm tra xem api có đang hoạt động tốt không tránh down nhiều.

**- Cấu trúc của một URL (Uniform Resource Locator):**

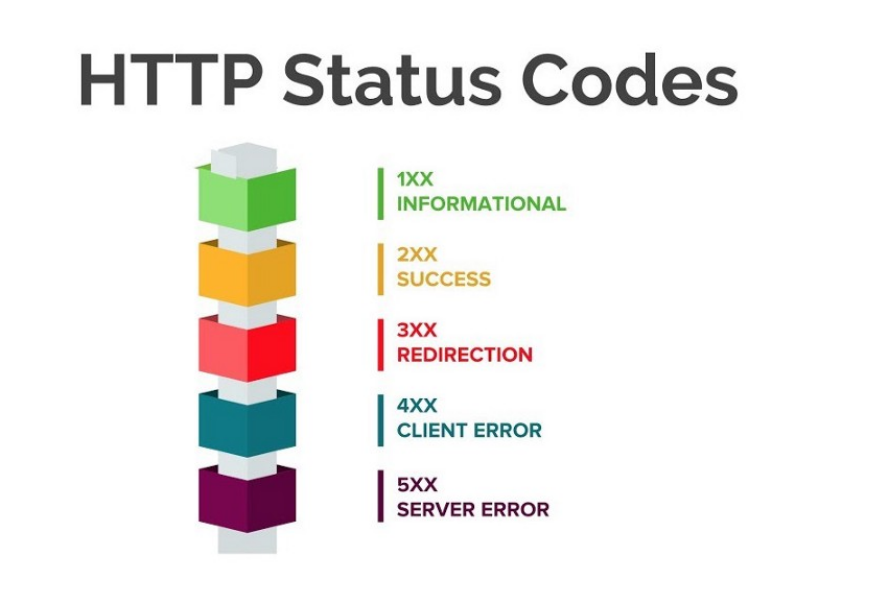
Một URL được sử dụng để xác định duy nhất một tài nguyên trên Web. Một URL có cấu trúc gồm 5 phần chính: **scheme**, **subdomain**, **top-level domain** , **second-level domain**, và **subdirectory**.

****

* **Scheme:** hiển thị cho máy chủ web sử dụng giao thức nào khi truy cập một trang trên web. HTTPS là giao thức phổ biến nhất
* **Subdomain:** Thường được gọi là tên miền con của tên miền chính. Chính vì là miền con của miền chính nên tên miền con này mang đầy đủ những tính chất như một miền chính và chúng ta có thể sử dụng nó giống như miền chính. Đặc điểm để nhận dạng là tên của chúng có chứa tiền tố đằng trước tên miền chính
* **Second-level domain(SLD):** Được coi như là tên của website mà người dùng đặt cho nó. SLD giúp người dùng nhận ra họ đang truy cập vào trang web của một thương hiệu hay cá nhân nhất định.
* **Top-level domain:** Hay còn được gọi với cái tên là tên miền chỉ định loại miền mà bạn đã đăng kí. Ví dụ:
* .com
* .net
* .org
* .gov
* .edu
* .vn
* .asia
* **Subdirectory:** (subfolder) cung cấp cho người dùng cái nhìn trực quan về phần cụ thể của trang web mà người dùng đang truy cập.

**- HTTP Status Code:**

HTTP status code là yếu tố quan trọng trong HTTP message response được server trả về cho client. Chia thành 5 loại:



**+ 1xx (100-199) Phản hồi thông tin:** Yêu cầu đã được chấp nhận và quá trình xử lý yêu cầu của người dùng đang được tiếp tục.

**+ 2xx (200-299): Phản hồi thành công:** Yêu cầu của người dùng đã được máy chủ tiếp nhận, hiểu và xử lý thành công.

**+ 3xx (300-399): Điều hướng:** Phía client cần thực hiện hành động bổ sung để hoàn tất yêu cầu.

**+ 4xx (400-499): Lỗi phía client:** Yêu cầu không thể hoàn tất hoặc yêu cầu chứa cú pháp không chính xác. 4xx sẽ hiện ra khi có lỗi từ phía client do không đưa ra yêu cầu hợp lệ.

**+ 5xx (500-599): Lỗi phía server:** Máy chủ không thể hoàn thành yêu cầu được cho là hợp lệ. Khi 5xx xảy ra, bạn chỉ có thể đợi để bên hệ thống máy chủ xử lý xong.

HTTP status code phổ biến:

**+ 200 OK:** Yêu cầu đã thành công

**+ 201 Created:** Yêu cầu đã thành công và kết quả là một tài nguyên mới đã được tạo. Đây thường là phản hồi được gửi sau các yêu cầu POST hoặc một số yêu cầu PUT.

**+ 400 Bad Request:** Máy chủ không thể hiểu yêu cầu do cú pháp không hợp lệ.

**+ 401 Unauthorized:** Request cần có auth. Có nghĩa là, client phải các tự xác thực chính mình để nhận được phản hồi đã yêu cầu.

**+ 404 Not found:** đối tượng không được lưu trên máy chủ (không tìm thấy)

**+ 500 Internal Server Error:** Một thông báo chung, được đưa ra khi máy chủ gặp phải một trường hợp bất ngờ, message cụ thể không phù hợp.

**+ 505 HTTP Version Not Supported:** Máy chủ không hỗ trợ phiên bản “giao thức HTTP”.

**- Tìm hiểu về một số trường:**

**+ header Host:** Chỉ định máy chủ và port của máy chủ mà yêu cầu đang được gửi. Nếu không có cổng nào được bao gồm, cổng mặc định của yêu cầu được sử dụng (443 cho URL HTTPS và 80 cho URL HTTP)

**+ User-Agent:** chứa thông tin về tác nhân người sử dụng tạo yêu cầu: ứng dụng, hệ điều hành, nhà cung cấp và phiên bản. Ví dụ: User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36

**+ Content-Type:** Trường này chỉ dẫn kiểu phương tiện của phần thân đối tượng được gửi tới người nhận. Ví dụ: Content-Type: application/json; charset=utf-8

**+ Content-Length:** Cho biết kích thước của thân thông báo, bằng byte, được gửi cho người nhận. Độ dài theo số thập phân của hệ 8. Ví dụ: Content-Length: 4514

**+ Cookie:** chứa cookie HTTP được lưu trữ được liên kết với máy chủ (tức là trước đây được gửi bởi máy chủ với Set-Cookie header hoặc đặt trong JS sử dụng Document.cookie). là tùy chọn và có thể được bỏ qua nếu, ví dụ: Cài đặt quyền riêng tư của trình duyệt block cookie.

Cú pháp: “Cookie: name=value”. Ví dụ: Cookie: PHPSESSID=298zf09hf012h2

**+ Location:** được sử dụng để điều hướng lại người nhận tới một vị trí khác ngoài Reqest-URI. Ví dụ: Location: <http://www.tutorialspoint.org../http/index.jsp>

**- Các kiểu encode data trong POST request:**

**+ application/x-www-form-urlencoded:** là kiểu mã hóa mặc định nếu thuộc tính enctype không có giá trị, đại diện cho URL Encoded Form. Tất cả các ký tự được mã hóa trước khi gửi (khoảng trắng được chuyển đổi thành ký hiệu "+" hoặc "%20" và các ký tự đặc biệt được chuyển đổi thành giá trị ASCII HEX).

Kiểu mã hóa được dùng là kiểu URL Encoded. Hiểu đơn giản thì dữ liệu được biểu diễn dưới dạng (key, value), nối với nhau bằng ký hiệu & thành một chuỗi (long string). Trong mỗi cặp (key, value), key và value tách nhau bởi dấu "="

Ở trường Content-Type trên header của request có xác định kiểu mã hóa là application/x-www-form-urlencoded

**+ multipart/form-data:** đại diện cho Multipart form, kiểu mã hóa này được sử dụng khi người dùng muốn tải tệp dữ liệu lên. Ngoài giá trị multipart/form-data, ở Content-Type còn có giá trị boundary. Giá trị này do trình duyệt tạo ra.

Mỗi cặp (key, value) được biểu diễn dưới dạng:

**--<<boundary\_value>>**

**Content-Disposition: form-data; name="<<field\_name>>"**

**<<field\_value>>**

* Như vậy, với kiểu mã hóa application/x-www-form-urlencoded, mỗi cặp (key, value) được phân cách với nhau bằng dấu & cho chép server biết nơi bắt đầu và kết thúc của một tham số. Còn với kiểu multipart/form-data, các giá trị boundary thực hiện công việc này.

**+ text/plain:** Là một kiểu mới trong HTML 5, dữ liệu gửi lên mà ko có mã hóa. Kiểu mã hóa này gần giống với kiểu URL encoded forms, ngoại trừ việc các trường của form không được mã hóa khi gửi lên server. Kiểu này không được dùng rộng rãi vì định dạng này có thể đọc được và kém bảo mật.

**- Tìm hiểu công cụ Burpsuite:**

Burp Suite là một trong những công cụ kiểm tra thâm nhập và tìm lỗ hổng phổ biến nhất và thường được sử dụng để kiểm tra bảo mật ứng dụng web. Burp Suite là một công cụ dựa trên proxy được sử dụng để đánh giá tính bảo mật của các ứng dụng dựa trên web và thực hiện kiểm tra thực hành. Nó có các mô đun mạnh mẽ và được đóng gói với các phần mở rộng tùy chọn có thể tăng hiệu quả kiểm tra ứng dụng web. Burp Suite là một công cụ pentest ứng dụng web. Ngoài ra, giao diện của Burp cũng rất trực quan và thân thiện. Chúng ta có thể nhìn rõ request được gửi (Request) cũng như phản hồi từ phía server (Respone)

Một số tính năng nổi bật của Burpsuite:

**+ Interception Proxy:** được thiết kế để bắt các request từ đó có thể tùy ý sửa đổi trước khi các request này được gửi lên server.

**+ Repeater:** cho phép sử dụng một request trước đó và tùy sửa đổi nội dung request một cách nhanh chóng nhiều lần khác nhau.

**+ Intruder:** tự động hóa việc gửi hàng loạt các request có chứa các payload tương tự nhau lên server.

**+ Decoder:** decode và encode string theo các format khác nhau (URL, Base64, HTML,…).

**+ Comparer:** chỉ ra sự khác nhau giữa các requests/responses

**+ Extender:** API để mở rộng chức năng của Burp Suite. Bạn có thể download các extensions thông qua Bapp Store.

**+ Spider & Discover Content:** crawl link có trong ứng dụng web.

**+ Scanner (chỉ có trong bản Pro):** đây là một mô đun khác mạnh mẽ, nó tự động quét các lỗ hổng trong ứng dụng web (XSS, SQLi, Command Injection, File Inclusion,…).

**Thực hành:**

1. Thực hiện GET trang chủ và in ra title của trang. Chương trình: **httpget.py**

2. Thực hiện POST vào trang đăng nhập để đăng nhập vào một tài khoản. In ra đăng nhập thành công hay thất bại.

Chương trình: **httppost.py**

3. Thực hiện upload một file ảnh lên Media Library. In ra đường dẫn file được upload. Chương trình: **httpupload.py**

4. Thực hiện GET để download một file ảnh trên máy chủ, hiển thị kích thước của file ảnh download được. Chương trình: **httpdownload.py**