LẬP TRÌNH PYTHON NÂNG CAO

TÀI LIỆU PHỤC VỤ SINH VIÊN NGÀNH KHOA HỌC DỮ LIỆU

Nhóm biên soạn:

TP.HCM - Năm 2021

MỤC LỤC

[BÀI 8: LẬP TRÌNH DỮ LIỆU MẠNG INTERNET 3](#_Toc80626782)

[1. Dữ liệu Internet và mô hình Request/Response 3](#_Toc80626783)

[1.1. HTTP 3](#_Toc80626784)

[1.2. HTTP Message 4](#_Toc80626785)

[1.2.1. HTTP Request 4](#_Toc80626786)

[1.2.2. HTTP Response 5](#_Toc80626787)

[2. Thư viện urllib trong python 6](#_Toc80626788)

[2.1. urllib request 7](#_Toc80626789)

[2.1.1. urlopen 7](#_Toc80626790)

[2.1.2. urllib.parse 7](#_Toc80626791)

[2.2. Handling Exceptions 9](#_Toc80626792)

[3. request 9](#_Toc80626793)

[3.1. Phương thức get 9](#_Toc80626794)

[3.2. Phương thức post 10](#_Toc80626795)

[3.3. Phương thức put 10](#_Toc80626796)

[3.4. Phương thức delete 10](#_Toc80626797)

[4. JSON và XML 10](#_Toc80626798)

[BÀI TẬP CHƯƠNG 8 11](#_Toc80626799)

# BÀI 8: LẬP TRÌNH DỮ LIỆU MẠNG INTERNET

**Mục tiêu:**

* *Nắm vững được Python để viết các đoạn lệnh.*
* *Hiểu được khái niệm API và định dạng dữ liệu JSON/XML.*
* *Sử dụng được gói request, urllib để thực hiện các tác vụ truy vấn dữ liệu Internet.*

**Nội dung chính:**

# Dữ liệu Internet và mô hình Request/Response

## HTTP

HTTP (Hyper-Text-Transfer-Protocol) là một giao thức mức ứng dụng cho các hệ thống phân phối, cộng tác, đa phương tiện. Nó được thiết kế bởi Tim Berners Lee (1989). Giao thức này dựa trên giao thức TCP/IP. Đây là nền tảng đã được World-Wide Web (WWW) sử dụng từ năm 1990, cho phép tải (fetching) các nguồn tài nguyên trên mạng như tài liệu HTML, text, video, ảnh.

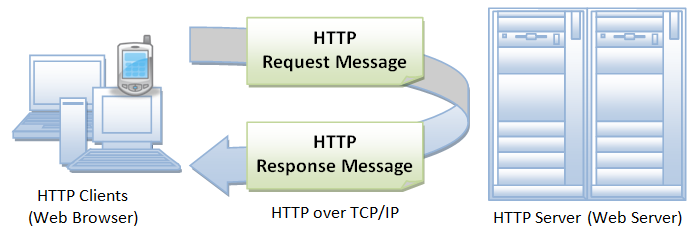
Nó cũng là nền tảng cho bất kỳ trao đổi dữ liệu nào được thực hiện trên Web với mô hình Client/Server. Các yêu cầu (Request/HTTP Request) từ client được gửi đến Server, sau đó Server xử lý yêu cầu và gửi lại (Response/HTTP Response) dữ liệu đến client. Các Request và Response là các Message (HTTP Message).

**Stateless**: HTTP là giao thức phi trạng thái (stateless), không có mối liên hệ nào giữa hai yêu cầu (request) được gửi đi, dù nó thực hiện trên cùng một kết nối (đến cùng 1 server). Thí dụ, client, thông qua web browser (như chrome/firefore) gửi yêu cầu đến <https://www.vanlanguni.edu.vn/dao-tao/nganh-dao-tao>, tiếp tục nó có thể gửi một yêu cầu khác đến <https://www.vanlanguni.edu.vn/trang-chu/lich-su>, cả hai yêu cầu này không có mối liên hệ gì với nhau.

**Cookie**: Đôi khi một ứng dụng cần chia sẻ trạng thái giữa những yêu cầu của phiên làm việc (session). Chẳng hạn trong ứng dụng bán hàng online, người dùng thêm 1 sản phẩm vào giỏ. Sau đó, quay lại trang chính để tìm kiếm và thêm sản phẩm khác vào giỏ. Do HTTP là stateless, nên HTTP Cookie sẽ hỗ trợ nhằm duy trì trạng thái của hai phiên làm việc trên. Phần Cookie này được thêm vào phần header của HTTP Message.

## HTTP Message

Thông điệp HTTP (HTTP Message) là thông tin được trao đổi qua lại giữa Client và Server. Trong đó, HTTP Request Message là thông điệp gửi từ Client đến Server, và ngược lại là HTTP Response Message.

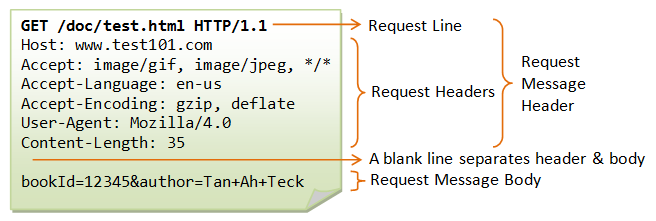


Phiên bản HTTP/1.1 của HTTP Message được chuẩn hóa vào năm 1997, đây là phiên bản giao thức HTTP mặc định trong các trình duyệt. Trong khi đó, HTTP/2.0 chuẩn hóa năm 2015 là phiên bản cải tiến tiếp theo. Nếu Server hỗ trợ HTTP/2.0 thì các trình duyệt sẽ tự động sử dụng phiên bản HTTP/2.0.

## HTTP Request

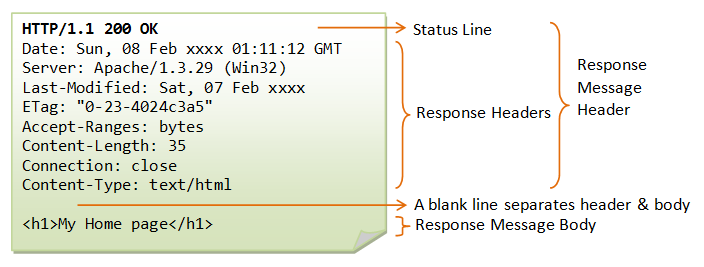
HTTP Request là thông báo yêu cầu từ máy khách gữi đến máy chủ. Nội dung thông báo bao gồm: Request Line, Request Header và Request Body (tùy chọn).

* **Request Line**:
  + Tên phương thức: GET, POST, PUT, DELETE
  + URL: là địa chỉ định danh tài nguyên
  + HTTP version: phiên bản của giao thức HTTP (ví dụ HTTP/1.0, HTTP/1.1)
* **Request Header**: cho phép client gửi thêm các thông tin bổ sung đính kèm với HTTP Request, Một số trường thông dụng như:
  + Accept loại nội dung có thể nhận được từ thông điệp response (ví dụ: text/plain, text/html…)
  + Accept-Encoding: các kiểu nén được chấp nhận (ví dụ gzip, deflate)
  + Connection: tùy chọn điều khiển cho kết nối hiện thời. Ví dụ: keepalive, Upgrade…
  + Cookie: thông tin HTTP Cookie từ server
  + User-Agent: thông tin về user agent của người dùng
* **Request Body** (tùy chọn): Các yêu cầu tìm nạp tài nguyên, như GET, HEAD, DELETE hoặc OPTIONS, thường không cần Request Body.



## HTTP Response

HTTP Response cũng là một HTTP Message, nên nó cũng có cấu trúc tương tự với HTTP Resquest, gồm các thành phần Response Line, Response Header và Response Body.



Response Line có cấu trúc như sau:

HTTP-version – Status Code – Reason Phrase

|  |  |
| --- | --- |
| HTTP-version | Phiên bản HTTP, thông dụng nhất là HTTP/1.1 |
| Status Code | Mã phản hồi do máy chủ gửi về. Ví dụ: 200, 301, 500 |
| Reason Phrase | Giải thích ngắn gọn về mã phản hồi |

# Thư viện urllib trong python

Module urllib trong Python 3 cho phép người lập trình để truy cập các trang web. Thông qua urllib, người phát triển ứng dụng có thể truy cập các trang web, tải xuống dữ liệu, phân tích cú pháp dữ liệu, sửa đổi tiêu đề và thực hiện bất kỳ yêu cầu GET và POST nào mà họ có thể cần thực hiện. Dưới đây là bảng liệt kê một số package của module urllib.

|  |  |
| --- | --- |
| **Modules** | **Mô tả** |
| urllib.request | mở và đọc |
| urllib.parse | phân tích cú pháp URL |
| urllib.error | cho các trường hợp ngoại lệ được nêu ra |
| urllib.robotparser | phân tích cú pháp tệp robot.txt |

(Các demo trong những thí dụ dưới đây sinh viên có thể tham khảo tại link: <https://github.com/huynhhoc/AdvancedPythonProgramming/tree/main/Chapter%208>)

Cài đặt urllib:

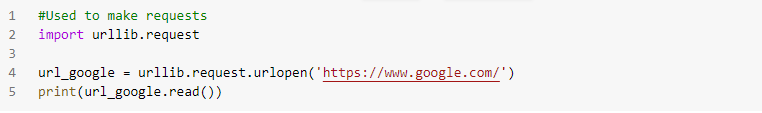
pip install urllib

# urllib request

# urlopen

urlopen dùng để mở một đối tượng mạng thông qua chuỗi URL. Trường hợp kết nối thành công, một đối tương giống như tập tin sẽ được trả về, nếu kết nối không thể được thực hiện, ngoại lệ IOError được đưa ra. Đôi tượng trả về hỗ trợ các phương thức sau: read (), readline (), readlines (), fileno (), close (), info (), getcode () và geturl ().

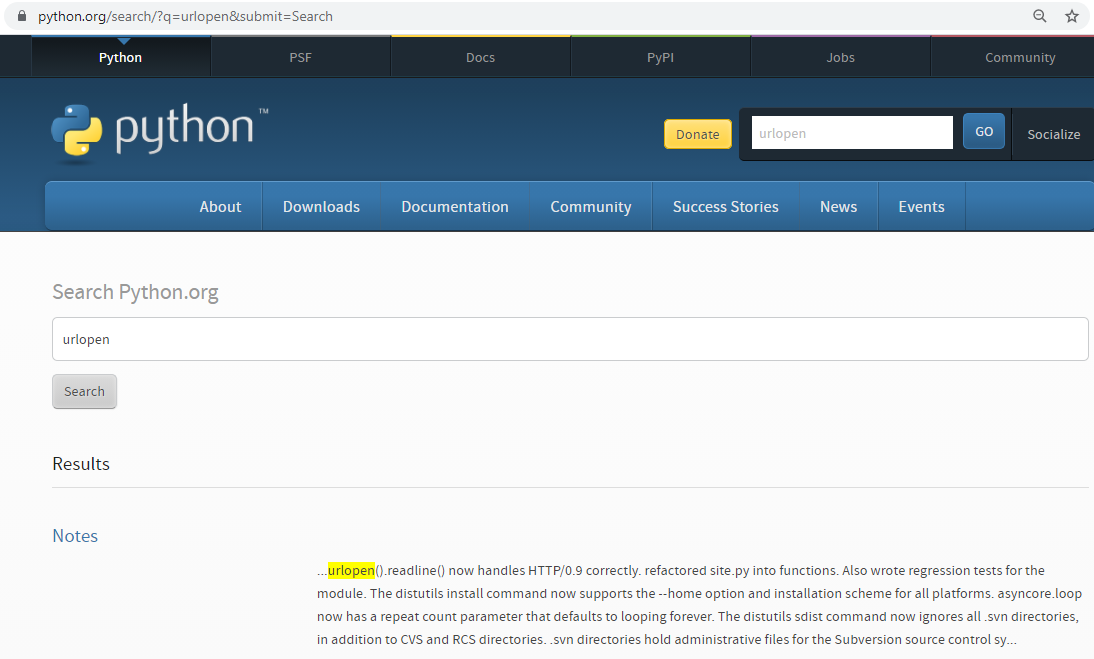
**Thí dụ 1**:



**Thí dụ 2**: Sinh viên hãy phát triển trên thí dụ 1 bằng cách sử dụng các hàm info(), getcode() và geturl(), sau đó ghi lại kết quả và giải thích ý nghĩa của từng hàm.

# urllib.parse

Giả sử chúng ta muốn tìm kiếm thông tin ‘basic’ từ trang web [www.python.org](http://www.python.org), chúng ta có thể vào trang đó và gõ từ **openurl** vào trong mục tìm kiếm, rồi nhấn vào nút **Seach,** khi đó địa chỉ trang có dạng là <https://www.python.org/search/?q=urlopen&submit=Search> (xem hình minh họa dưới đây).



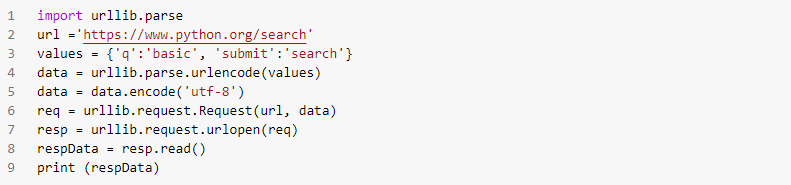
Khi đó để lấy được nội dung của kết quả tìm kiếm, ta xem thí dụ 2 dưới đây:

**Thí dụ 2**:



Bên cạnh đó, chúng ta có thể sử dụng thư viện **urllib.parse** để thay thế, hãy xem thí dụ 3 dưới đây (kết quả thí dụ 3 hoàn toàn tương tự thí dụ 2)

**Thí dụ 3**:



# Xử lý ngoại lệ

urlopen có thể gây ra ngoại lệ (exception) URLError trong trường hợp máy chủ tại địa chỉ url không phản hồi kết quả. Khi đó, người phát triển ứng dụng nên nắm bắt và xử lý các ngoại lệ này. Xem thí dụ 4 và 5 dưới đây.

**Thí dụ 4**:



**Thí dụ 5**:



# Rest API và Webservice

# Kiến trúc của Rest

REST là viết tắt của chuyển giao trạng thái biểu diễn và là một kiểu kiến ​​trúc phần mềm xác định một mẫu cho các giao tiếp máy khách và máy chủ qua mạng. REST cung cấp một tập hợp các ràng buộc cho kiến ​​trúc phần mềm để thúc đẩy hiệu suất, khả năng mở rộng, tính đơn giản và độ tin cậy trong hệ thống.

# Request

request là một thư viện HTTP được cấp phép của Apache2, được viết bằng Python. Nó được thiết kế để người lập trình sử dụng để tương tác với ngôn ngữ. Điều này có nghĩa là người lập trình không phải thêm các chuỗi truy vấn vào URL hoặc form-code của dữ liệu POST theo cách thủ công.

Cài đặt thư viện requests:

pip install requests

# Phương thức get

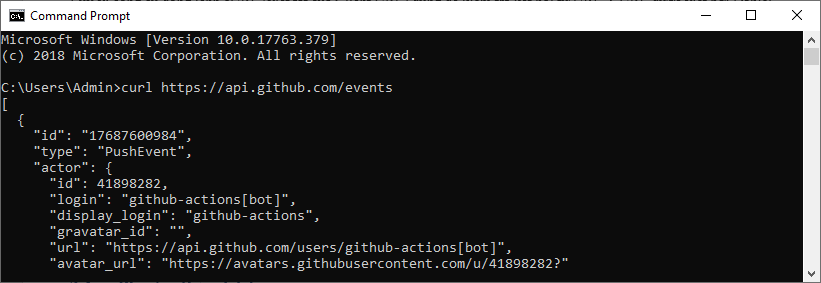
Lưu ý: công cụ dòng lệnh cURL (viết tắt của Client URL) dùng để kiểm tra kết nối từ URL. CURL được viết bởi Daniel Stenberg (1997), nó sử dụng nhiều giao thức hỗ trợ khác nhau như: HTTP, HTTPS, …

Cú pháp:

curl link\_api

Thí dụ:

curl https://api.github.com/events



# Phương thức post

# Phương thức put

# Phương thức delete

# JSON và XML

**Tài liệu tham khảo:**

1. <https://medium.com/geekculture/http-the-request-response-cycle-ddcb5bb70707>
2. <https://www.w3.org/Protocols/HTTP/1.1/rfc2616.pdf>
3. <https://xuanthulab.net/giao-thuc-http-va-cau-truc-co-ban-cua-http-message.html>
4. <https://docs.microsoft.com/en-us/dotnet/api/system.net.http.httprequestmessage.version?view=net-5.0>
5. <https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages>
6. <https://www.geeksforgeeks.org/python-urllib-module/>
7. <https://medium.com/geekculture/http-the-request-response-cycle-ddcb5bb70707>
8. <https://realpython.com/api-integration-in-python/>

|  |
| --- |
| BÀI TẬP CHƯƠNG 8 |

**[Yêu cầu sinh viên nộp bài]**

**Bài tập 1:** Sinh viên hãy hoàn thành thí dụ 1 bằng cách bổ sung vào đoạn chương trình để lưu kết quả hiện thị trong tập tin có định dạng html, có xử lý ngoại lệ **URLError** và **HTTPError**, và hãy nhận xét kết quả nhìn thấy được.

**Bài tập 2:** Sinh viên hãy cho biết kết quả của câu lệnh sau đây, sinh viên có nhận xét gì về kết quả này

x = urllib.request.urlopen('https://www.google.com/search?q=test')

**Bài tập 3:** Sinh viên hãy quan sát hai cách truy vấn dữ liệu từ trang <http://python.org> dưới đây, so sánh và nhận xét

**Cách 1:**

**import** **urllib.request**

url\_google = urllib.request.urlopen('http://python.org/')

html = url\_google.read()

**Cách 2:**

**import** **urllib.request**

**with** urllib.request.urlopen('http://python.org/') **as** response:

html = response.read()

**Bài tập 4:** Đoạn code dưới đây dùng để lưu trữ ảnh (**binary\_framing2.png**) lấy từ địa chỉ **https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages/binary\_framing2.png**, sinh viên hãy phát triển đoạn script này để có thể lưu hết toàn bộ các ảnh tại địa chỉ **https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages/**.

**import** **urllib.request**

urllib.request.urlretrieve("https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages/binary\_framing2.png", "binary\_framing2.png")

**Gợi ý**: Sinh viên sử dụng biểu thức regular-expession (thư viện re) để phân tích lấy url chứa ảnh.

**Bài tập 5:** Đoạn chương trình dưới đây đọc dữ liệu từ địa chỉ **'http://sixty-north.com/c/t.txt**' và kết quả lưu vào trong danh sách lstStory. Sinh viên hãy phát triển đoạn chương trình này để có thể lưu toàn bộ dữ liệu đọc được với định dạng tập tin lưu trữ là csv.

**from** **urllib.request** **import** **urlopen**

with urlopen('http://sixty-north.com/c/t.txt') as story:

lstStory = story.readlines()

print(lstStory[0])

print('done')

**Bài tập 6:**

**Phụ lục: Những Status Code thông dụng**

|  |  |  |
| --- | --- | --- |
| **Code** | **Status** | **Meaning** |
| 2xx Success | | |
| 200 | OK | Standard response for successful HTTP requests. The actual response will depend on the request method used. In a GET request, the response will contain an entity corresponding to the requested resource. In a POST request, the response will contain an entity describing or containing the result of the action. |
| 201 | Created | The request has been fulfilled, resulting in the creation of a new resource. |
| 204 | No Content | The server successfully processed the request and is not returning any content. |
| 206 | Partial Content | The server is delivering only part of the resource (byte serving) due to a range header sent by the client. The range header is used by HTTP clients to enable resuming of interrupted downloads, or split a download into multiple simultaneous streams. |
| 3xx Redirection | | |
| 301 | Moved Permanently | This and all future requests should be directed to the given URI.. |
| 302 | Found | The HTTP/1.0 specification (RFC 1945) required the client to perform a temporary redirect (the original describing phrase was "Moved Temporarily") |
| 303 | See Other | The response to the request can be found under another URI using a GET method. When received in response to a POST (or PUT/DELETE), the client should presume that the server has received the data and should issue a redirect with a separate GET message. |
| 4xx Client errors | | |
| 400 | Bad Request | The server cannot or will not process the request due to an apparent client error (e.g., malformed request syntax, size too large, invalid request message framing, or deceptive request routing). |
| 401 | Unauthorized | Similar to 403 Forbidden, but specifically for use when authentication is required and has failed or has not yet been provided. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource. |
| 403 | Forbidden | The request was valid, but the server is refusing action. The user might not have the necessary permissions for a resource, or may need an account of some sort. |
| 404 | Not Found | The requested resource could not be found but may be available in the future. Subsequent requests by the client are permissible. |
| 429 | Too Many Requests | The user has sent too many requests in a given amount of time. |
| 5xx Server errors | | |
| 500 | Internal Server Error | A generic error message, given when an unexpected condition was encountered and no more specific message is suitable. |
| 502 | Bad Gateway | The server was acting as a gateway or proxy and received an invalid response from the upstream server |
| 503 | Service Unavailable | The server is currently unavailable (because it is overloaded or down for maintenance). Generally, this is a temporary state. |