# **VARIABLES TRÊN POSTMAN**

1. Đặt vấn đề
   1. Ví dụ thực tế

Giả sử bạn là HR quản lý chấm công cho 50 nhân viên IT của 1 công ty, trước đây có quy định thời gian làm của nhân viên IT là 9h -18h. Bạn sẽ dựa vào thời gian quy định để xét xem ai đi muộn sẽ bị phạt. Tuy nhiên hiện tại công ty thay đổi chính sách, yêu cầu nhân viên IT phải đi làm từ 8h – 17h. Vậy giờ làm sao để cập nhật mới thời gian chấm công quy định cho từng nhân viên ?

Nếu bạn vào từng profile của từng nhân viên để sửa thời gian chấm công quy định sẽ rất mất thời gian và có thể bị sót nhiều người.

Thay vào đó bạn có thể gom tất cả nhân viên đó vào 1 phòng ban là phòng IT, bạn sửa thời gian chấm công quy định của phòng IT đó từ 9h - 18h thành 8h - 17h. Vậy là giờ tất cả nhân viên thuộc phòng IT sẽ áp dụng thời gian chấm công mới của phòng IT mà bạn không cần vào profile từng người để sửa

* 1. Liên hệ với Postman

Trong postman có 1 collection gồm nhiều request, ví dụ có thể là

* <https://dev-umee.ssf.vn/cms/getAllUser>
* https://dev-umee.ssf.vn/cms/getAllAccount
* https://dev-umee.ssf.vn/profile/api/commune/v1/searchCommunesByName
* https://dev-umee.ssf.vn/nickname/api/shop/internal/register-nickname-shop
* https://dev-umee.ssf.vn/smartbank/api/savings/v1/onlineSavings/22420683
* <https://dev-umee.ssf.vn/smartbank/api/savings/v1/periods>
* <https://dev-umee.ssf.vn/nickname/api/edit-nickname>
* https://dev-umee.ssf.vn/cms/searchModule
* https://dev-umee.ssf.vn/cms/  
  smartbank/province?keyword=&page=0&size=10&deleted=&klbCode=&nameEn=

…

Bạn có thể thấy https/dev-umee.ssf.vn được lặp lại nhiều lần, vì tất cả các api trên đều nằm trên 1 server, nếu 1 ngày server được build lại và người quản trị đổi tên domain từ dev-umee.ssf.vn thành dev-flex-umee.ssf.vn, khi đó muốn chạy các api kia bạn phải vào từng api để đổi từng cái domain từ dev-umee.ssf.vn thành dev-flex-umee.ssf.vn, điều đó rất mất thời gian, cũng như dễ xảy ra sai sót và lỗi, vì nếu giả sử có 100 request bạn phải đổi 100 lần

Thay vào đó, bạn có thể gán <https://dev-umee.ssf.vn> với 1 biến (Variables) {{umeeDev}}, các api khác sẽ gọi biến {{umeeDev}} và map giá trị <https://dev-umee.ssf.vn> vào url, nếu có thay đổi bạn chỉ cần thay đổi giá trị của biến {{umeeDev}}, tất cả các api có biến {{umeeDev}} sẽ lấy giá trị bạn vừa thay đổi kia để map

Cách viết: <https://dev-umee.ssf.vn>/cms/getUser → {{umeeDev}}cms/getUser

1. Variables scope (biến)

Phạm vi truy cập:

A diagram of a variety of variables

Description automatically generated

Có 5 loại biến trong postman nhưng trong bài này chúng ta chỉ tìm hiểu 3 loại là global, collection và environment

* Global Variables:
* Collection Variables: biến có thể truy cập từ một request nằm trong collection tương ứng.
* Environment Variables: biến có thể truy cập với Environment tương ứng.(Environment trong Postman sẽ được đề cập tới ở phần dưới)
* Data Variables
* Local Variables

Dưới đây là sự giống và khác nhau giữa Collection Variables và Environment Variables

|  |  |  |
| --- | --- | --- |
|  | Collection Variables | Environment Variables |
| Giống nhau | * đều cho phép bạn lưu trữ các biến để sử dụng trong các api của bạn. * đều có thể được tham chiếu và sử dụng trong các yêu cầu bằng cách sử dụng cú pháp {{variable\_name}}. * đều cho phép bạn cập nhật và thay đổi giá trị của các biến. | |
| Khác nhau | * Được lưu trữ trong phạm vi của một collection. * Chỉ có thể sử dụng trong 1 collection | * Được lưu trữ trong phạm vi của environment. * Sử dụng ở nhiều collection khác nhau |

* 1. Global Varibles

Biến toàn cục: biến có thể truy cập ở bất kỳ đâu

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

B2: Kiểm tra biến đã tạo

A screenshot of a computer

Description automatically generated

* 1. Collection Variables

B1: Tạo biến

A screenshot of a computer

Description automatically generated

B2: Kiểm tra biến đã tạo

A screenshot of a computer

Description automatically generated

* 1. Environment Variables

B1: Tạo 1 Environment

A screenshot of a computer

Description automatically generated

B2: Tạo variable

* *Cách 1*: Chọn 1 api bất kỳ → ở góc bên trên bên phải chọn environment muốn tạo biến (ở đây là môi trường: umee)

A screenshot of a computer

Description automatically generated

Sau đó làm theo các bước dưới đây

A screenshot of a computer

Description automatically generated

Kiểm tra biến

A screenshot of a computer

Description automatically generated

* *Cách 2*: Thêm trực tiếp vào environment

A screenshot of a computer

Description automatically generated

1. Tạo biến trong api login để mỗi lần login không cần thêm access\_token:

Vấn đề: Mỗi khi gọi 1 api cần có author, ta đều phải thêm access\_token vào header của request, nếu token hết hạn, ta phải call lại api login và phải copy paste token vào value của Authorization, nếu có 100 request, ta phải thực hiện copy paste 100 lần → mất thời gian

A screenshot of a computer

Description automatically generated

Ý tưởng: mỗi khi call api login sẽ trả về access\_token, ta lấy access\_token đó update lại vào trong biến {{login}} ta vừa tạo, rồi gán biến đó vào header của api, mỗi khi api call sẽ lấy giá trị token từ biến {{login}}

* + B1: Call api login để lấy access\_token

A screenshot of a computer

Description automatically generated

* + B2: Tạo 1 biến để lấy access\_token (bạn có thể chọn loại collection hay environment tùy bạn, ở đây tôi chọn tạo trên environment)

A screenshot of a computer

Description automatically generated

Thêm access\_token vừa lấy từ bước 1 vào Initial value[[1]](#footnote-1) và Current value[[2]](#footnote-2) rồi ấn Save

* B3: Vào api login, ta chọn set an environment variable, nó sẽ kết quả như hình dưới, ở đây có variable\_key, variable\_value

A screenshot of a computer

Description automatically generated

* B4: thêm dòng code sau

var jsonData = JSON.parse(responseBody);

pm.environment.set("loginDev", jsonData.data.access\_token);

A screenshot of a computer

Description automatically generated

Giải thích code:

* + **var** là từ khóa hỗ trợ khai báo biến mà không cần kiểu dữ liệu, kiểu dữ liệu sẽ được xác định khi gán giá trị cho biến, lúc đó chương trình sẽ tự ép kiểu cho biến
  + **jsonData**: tên biến
  + **responseBody**: là một biến có sẵn được sử dụng để truy cập vào nội dung (body) của phản hồi của yêu cầu. Nếu phản hồi của yêu cầu là một chuỗi JSON, chúng ta cần chuyển đổi nó thành một đối tượng JavaScript để có thể truy cập và sử dụng dữ liệu trong nó.
  + **JSON.parse()** là một phương thức JavaScript được sử dụng để phân tích (parse) một chuỗi JSON và chuyển đổi nó thành một đối tượng JavaScript.
  + Khi câu lệnh JSON.parse(responseBody) được thực thi, nó sẽ chuyển đổi giá trị của biến responseBody thành một đối tượng JavaScript và gán cho biến jsonData.
  + pm.environment.set("loginDev", jsonData.data.access\_token);
  + loginDev: biến {{loginDev}} ta vừa tạo
  + jsonData.data.access\_token: get ra dữ liệu access\_token từ response và ghép chuỗi token với Bearer

**→** 2 dòng code trên sẽ lấy dữ liệu access\_token mới nhất được sinh ra mỗi khi call api, ghép thành 1 chuỗi, vd là Bearer eyJhbGciOiJSUzI1NiIsInR5c… rồi gán vào biến {{loginDev}}, biến {{loginDev}} sẽ cập nhật vào current value

* B5: Chọn tab Authorization của api, Type chọn Bearer token, ô token điền {{loginDev}}, api sẽ tự lấy giá trị token mới nhất từ {{loginDev}} để xác thựcA screenshot of a computer

  Description automatically generated

1. Initial value (giá trị ban đầu): Đây là giá trị mà bạn đặt cho biến khi bạn tạo nó. Giá trị ban đầu có thể được định nghĩa tùy ý và không thay đổi trong quá trình chạy kịch bản hoặc thực thi yêu cầu. Initial value chỉ là một giá trị mặc định, nhưng không bị ảnh hưởng bởi bất kỳ thay đổi nào trong quá trình thực thi. [↑](#footnote-ref-1)
2. Current value (giá trị hiện tại): Đây là giá trị của biến tại thời điểm hiện tại, có thể thay đổi trong quá trình chạy kịch bản hoặc thực thi yêu cầu. Current value có thể được cập nhật hoặc thay đổi bằng cách gán giá trị mới cho biến trong quá trình thực thi kịch bản hoặc yêu cầu. Giá trị hiện tại của biến có thể được sử dụng và truyền đi trong các yêu cầu và kịch bản khác nhau.

   Ví dụ: Nếu bạn tạo một biến có tên là "accessToken" với giá trị ban đầu là "null", thì giá trị ban đầu của biến là "null". Khi bạn thực hiện một yêu cầu để lấy mã truy cập (access token) từ một API, bạn có thể gán giá trị mới cho biến "accessToken" bằng cách lấy mã truy cập từ phản hồi (response) và gán nó cho biến. Giá trị hiện tại của biến "accessToken" sẽ được cập nhật thành mã truy cập mới và có thể được sử dụng trong các yêu cầu và kịch bản khác. [↑](#footnote-ref-2)