Hướng dẫn sử dụng JSON-Server để tự build API và API Testing bằng JMeter.

Contents

[1. Hướng dẫn sử dụng JSON Server: 3](#_Toc478573673)

[1.1. JSON-Server là gì? 3](#_Toc478573674)

[1.1. Yêu cầu cần thiết để chạy JSON-Server: 3](#_Toc478573675)

[1.3. Chạy JSON-Server: 4](#_Toc478573676)

[2. API Testing với Jmeter: 7](#_Toc478573677)

[2.1. Chuẩn bị file dữ liệu: 7](#_Toc478573678)

[2.2. Tạo file input.json: 8](#_Toc478573679)

[2.3. Tạo Test Plan JMeter: 9](#_Toc478573680)

[2.3.1. Thêm 1 config element User Defined Variable: 9](#_Toc478573681)

[2.3.2. Thêm 1 config element CSV Data Set Config: 9](#_Toc478573682)

[2.3.3. Thêm HTTP Request Default: 10](#_Toc478573683)

[2.3.4. Thêm Thread Group: 10](#_Toc478573684)

[2.3.5. Thêm HTTP Sampler: 10](#_Toc478573685)

[2.3.6. Thêm 1 JSR223 Pre-Processor: 10](#_Toc478573686)

[2.3.6. Thêm 1 Assertion: 11](#_Toc478573687)

[2.3.7. Run Test Plan: 11](#_Toc478573688)

# 1. Hướng dẫn sử dụng JSON Server:

## JSON-Server là gì?

JSON-Server là công cụ giúp khởi tạo 1 REST API fake mà không yêu cầu kỹ năng coding. Việc khởi tạo fake API này tốn chưa tới 30 s.

Để tìm hiểu thêm thông tin, truy cập đường link : <https://github.com/typicode/json-server>

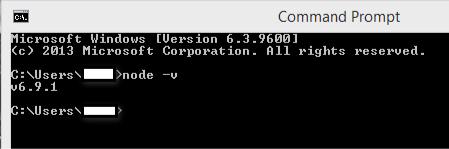
## Yêu cầu cần thiết để chạy JSON-Server:

Để chạy được JSON-Server thì trước hết phải đảm bảo trên máy tính local đã cài đặt NodeJS. Ở phần này mình sẽ hướng dẫn các bạn cài JSON-Server trên Windows.

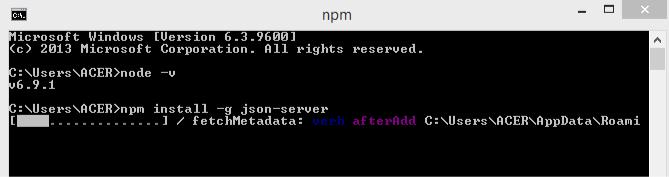
Để kiểm tra máy của bạn đã cài đặt NodeJS hay chưa, start command line và

gõ : node –v

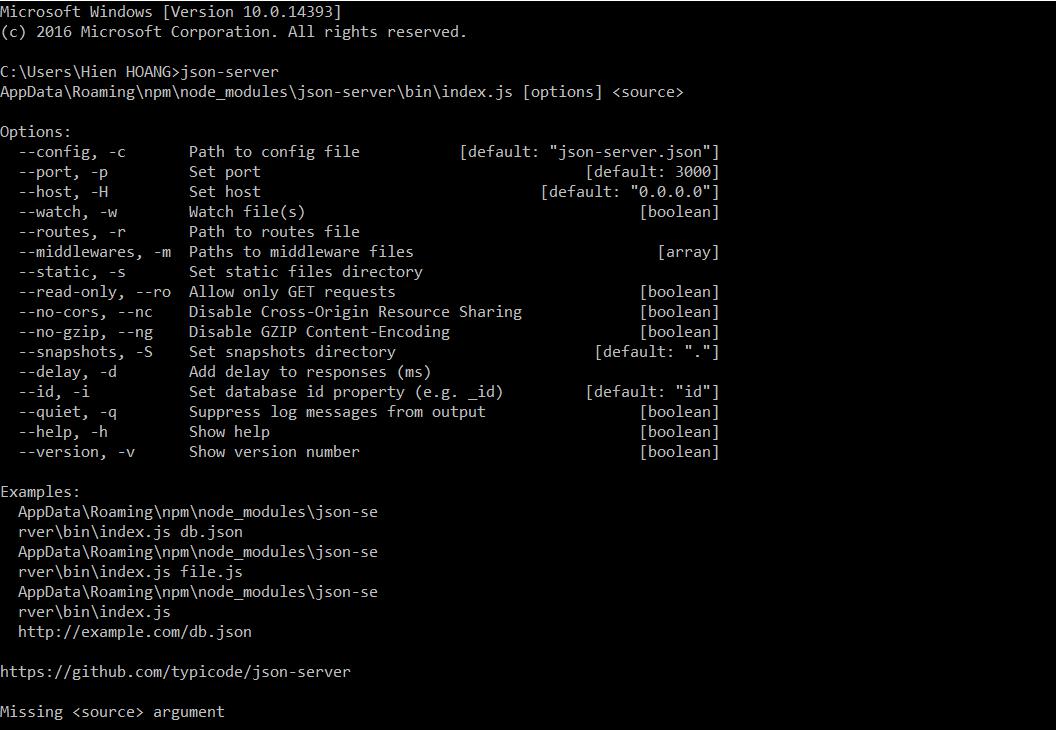
Nếu máy bạn đã cài đặt NodeJS thì bạn sẽ nhìn thấy thông tin sau :



Nếu máy bạn chưa cài đặt NodeJS, vui lòng cài đặt. Link download ở đây: [www.nodejs.org](http://www.nodejs.org/)



Sau khi JSON-Server đã tải về hoàn tất, để kiểm tra, bạn gõ vào command line : json-server



## 1.3. Chạy JSON-Server:

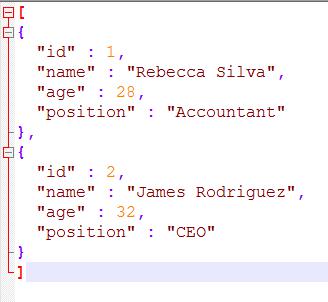
Sau khi bạn đã cài đặt xong JSON-Server, tiếp đến bạn khởi tạo 1 file \*.json, chẳng hạn : book.json.

Sau đó bạn chỉ việc launch file \*.json này bằng command line :

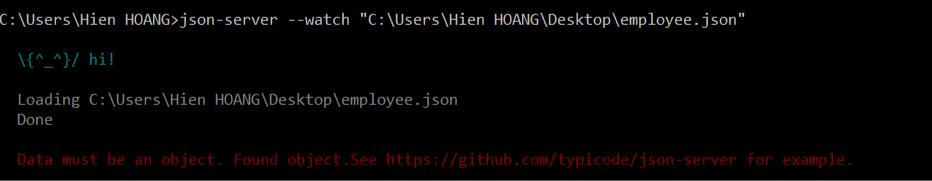
json-server –watch “path/to/file.json”

Khi đó, các bạn có thể truy cập tới fake API ở địa chỉ : localhost:3000/json\_filename

Các bạn nên lưu ý 1 điều là JSON-Server nó chỉ đọc được file \*.json với kiểu syntax global là Object. Chẳng hạn nếu file \*.json của bạn có cấu trúc kiểu Array thế này:



Thì khi run command, bạn sẽ thấy JSON-Server thông báo như sau :



Syntax cuả file \*.json :

{

“objects” :

[

{object1},

{object2}

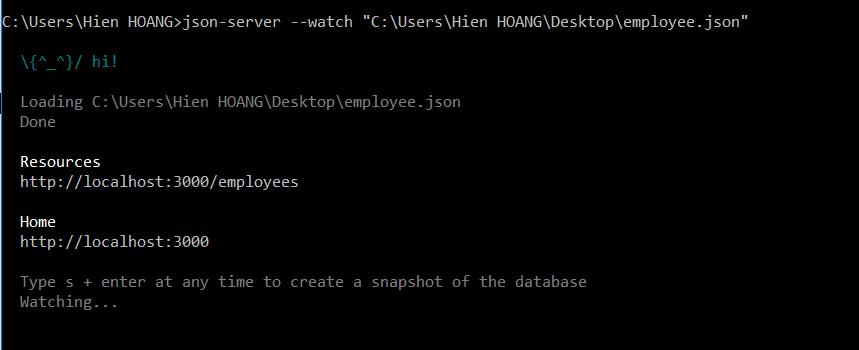
]

}

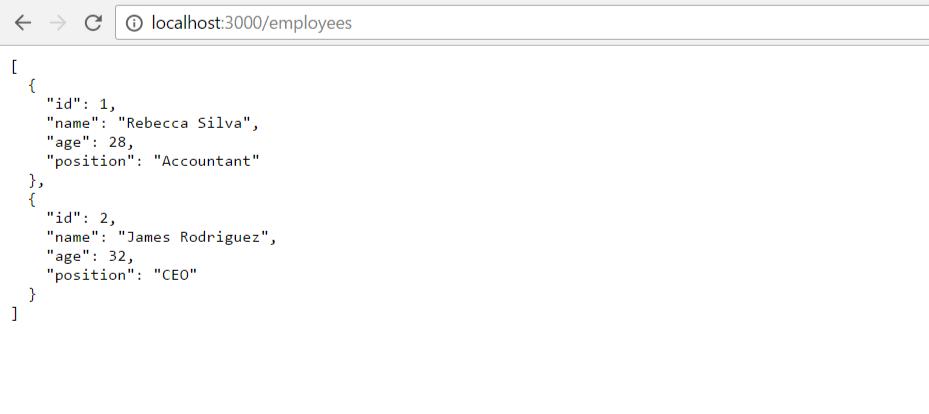
Trong trường hợp trên, nếu sửa lại thành như sau :



Thì khi chạy command line, bạn sẽ thấy không còn dòng thông báo “Data must be an object” nữa.



Nếu như truy cập vào địa chỉ : localhost:3000/employees, bạn sẽ thấy:



Giờ các bạn đã hiểu syntax của JSON-Server rồi.Giờ chuyển qua phần API Testing nào!

Trước hết, mình khởi tạo 1 file JSON trống nhé:

{

"employees": []

}

Và khi chạy JSON-Server, mình có 1 cái API thế này:



# 2. API Testing với Jmeter:

Phần này mình sẽ hướng dẫn test API theo hướng data-driven.

Tại sao mình lại nói hướng data-driven? Bởi vì nếu như trong trường hợp bạn có 1 Test Plan với 1 GET Request, sau đó 5 POST Requests, tiếp đó 5 DELETE requests, bạn sẽ phải tạo 1 Test Plan với….11 requests, dẫn đến việc maintain những requests này tương đối phức tạp. Chưa kể việc nếu trật tự của các requests thay đổi, mình còn phải sửa lại Test Plan.

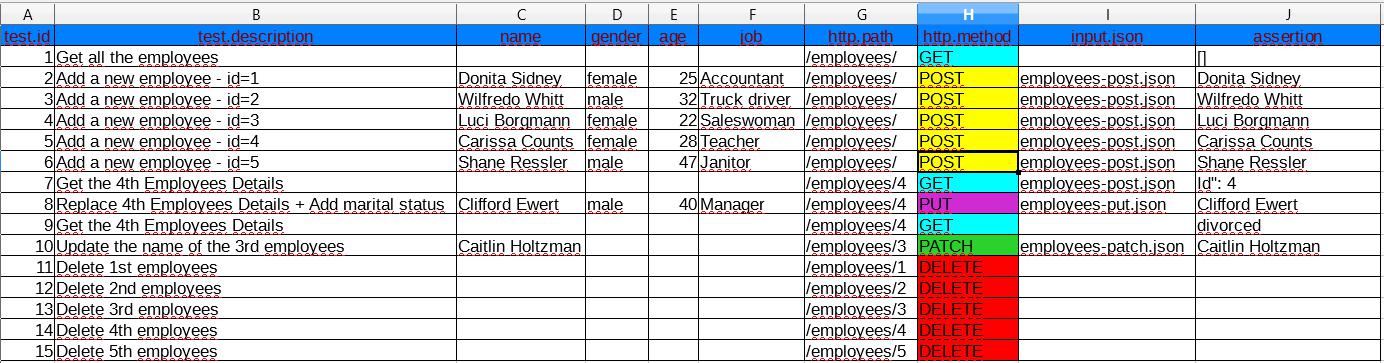
Nếu tạo 1 Test Plan theo hướng data-driven, bạn có thể tránh được vấn đề đau đầu này.

## 2.1. Chuẩn bị file dữ liệu:

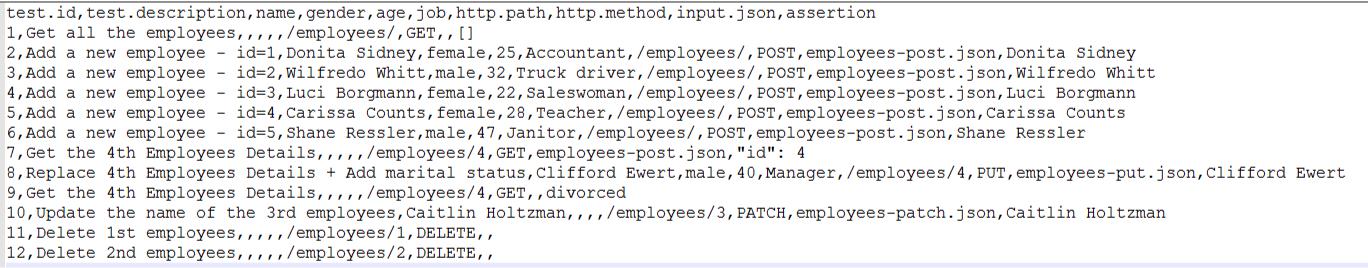
Ở đây, file dữ liệu sẽ chứa hết tất cả những dữ liệu có thể thay đổi ở những requests, bao gồm:

* Path
* Method
* JSON body data
* Assertion

Lấy ví dụ, để chuẩn bị dữ liệu cho việc test fake API employees, mình sẽ có file data như sau:



Lưu ý: Trên đây là nội dung mà mình copy từ file csv vào spreadsheet của Excel. Mình làm thế này để cho các bạn có thể hiểu rõ cấu trúc dữ liệu trong file CSV. File CSV của mình nhìn như thế này:



## 2.2. Tạo file input.json:

* Method = GET (#1):
  + Request GET có nhiệm vụ hiện thị tất cả các items trong danh sách employees. Với test case đầu tiên, do ban đầu khi khởi tạo, mình chỉ có 1 object JSON rỗng, nên API của mình sẽ trả về 1 response data với 1 list rỗng.
* Method = POST (#2 → #6):
  + Request POST có nhiệm vụ add thêm 1 employee vào danh sách employee. Ở đây 1 employee của mình sẽ gồm có : name, gender , age , job. Body data để truyền vào trong request POST sẽ có cú pháp như sau. Lưu ý ở đây mình sẽ truyền các thông số này in run time bằng các dữ liệu có trong file csv.

{

"name" : "${name}", "gender" : "${gender}", "age" : "${age}",

"job" : "${job}"

}

* Method = GET (#7 - #9):
  + Request GET có nhiệm vụ trả về 1 employee cụ thể dùng để kiểm tra xem các request POST/PUT/PATCH có chạy đúng hay không.
* Method = PUT(#8):
  + Request PUT có nhiệm vụ cập nhật dữ liệu của item hiện tại vào DB.

{

"name" : "${name}", "gender" : "${gender}", "age" : "${age}",

"job" : "${job}"

}

* Method = PATCH(#10):
  + Request PATCH có nhiệm vụ cập nhật 1 phần dữ liệu của item hiện tại vào DB.

{

"name" : "${name}"

}

* Method = DELETE(#11 - #12):
  + Request DELETE có nhiệm vụ xóa 1 item khỏi DB.

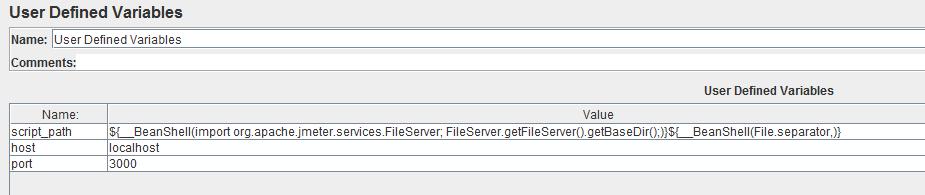
Lưu ý nhỏ:

PUT = thay thế TOÀN BỘ RESOURCE bằng 1 resource mới với 1 format mới.

PATCH = thay thế 1 phần của resource bằng các giá trị mà user cung cấp VÀ| HOẶC những phần khác của resource sẽ được cập nhật dù user không truyền dữ liệu (ví dụ: timestamps) VÀ|HOẶC cập nhật những resources khác bị ảnh hưởng bởi resource hiện tại(relationships).

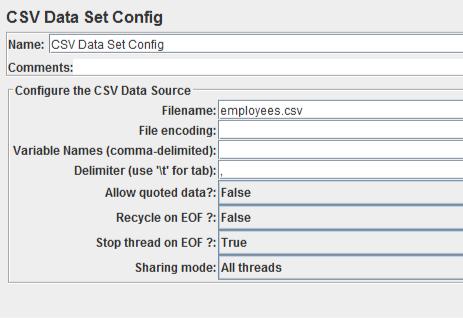
## 2.3. Tạo Test Plan JMeter:

### 2.3.1. Thêm 1 config element User Defined Variable:

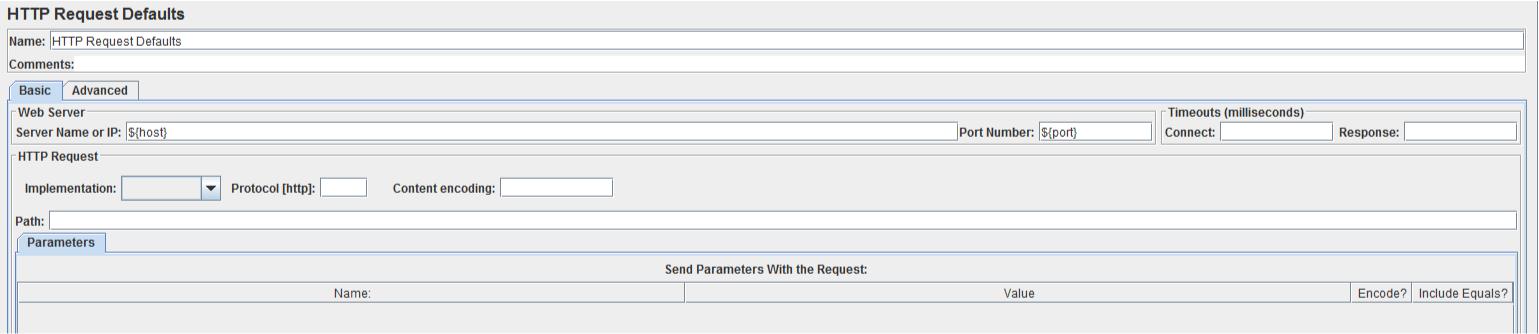


### 2.3.2. Thêm 1 config element CSV Data Set Config:

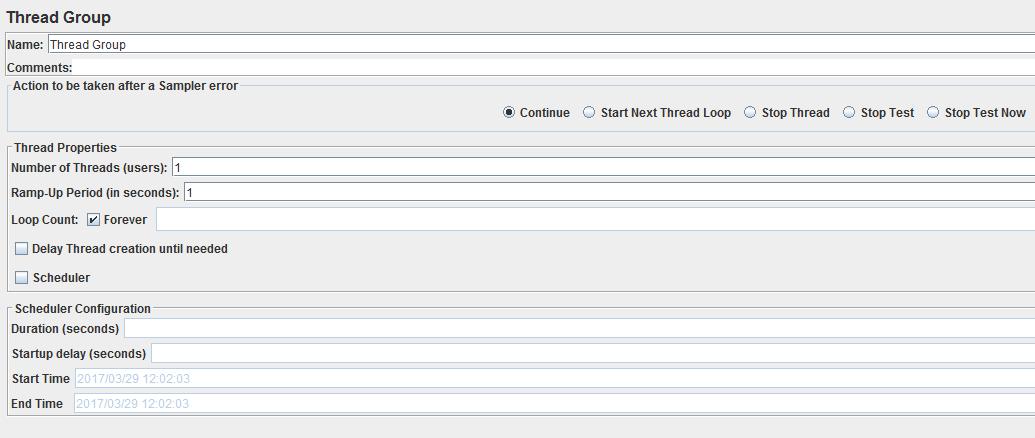
Đọc dữ liệu từ file CSV.



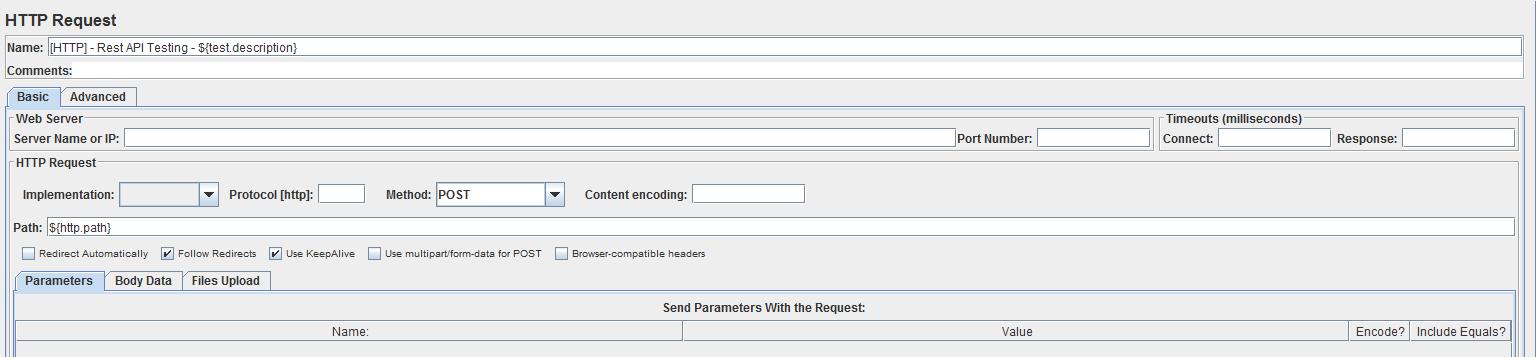
### 2.3.3. Thêm HTTP Request Default:



### 2.3.4. Thêm Thread Group:



### 2.3.5. Thêm HTTP Sampler:



### 2.3.6. Thêm 1 JSR223 Pre-Processor:

- Element này sẽ chịu trách nhiệm thay đổi HTTP Method dựa trên giá trị nhận được từ file CSV. Nó cũng chịu trách nhiệm trong việc set body data cho các request POST/PUT/PATCH:



- Dòng lệnh thay đổi method của request HTTP:

def httpMethod = vars.get("http.method");

sampler.setMethod(httpMethod);

- Những dòng lệnh dưới đây sẽ đọc tên của file json mình sẽ sử dụng để parse dữ liệu, tiếp đó thay thế các biến bằng giá trị tương ứng trong file csv.

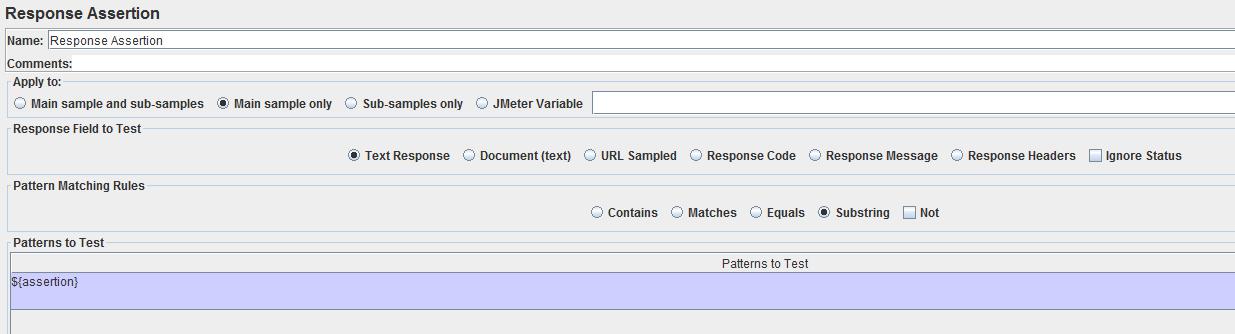
def dataToBePosted = new CompoundVariable(new File(vars.get("script\_path") + vars.get("input.json")).text).execute();

def arg= new HTTPArgument("", dataToBePosted, null, true);

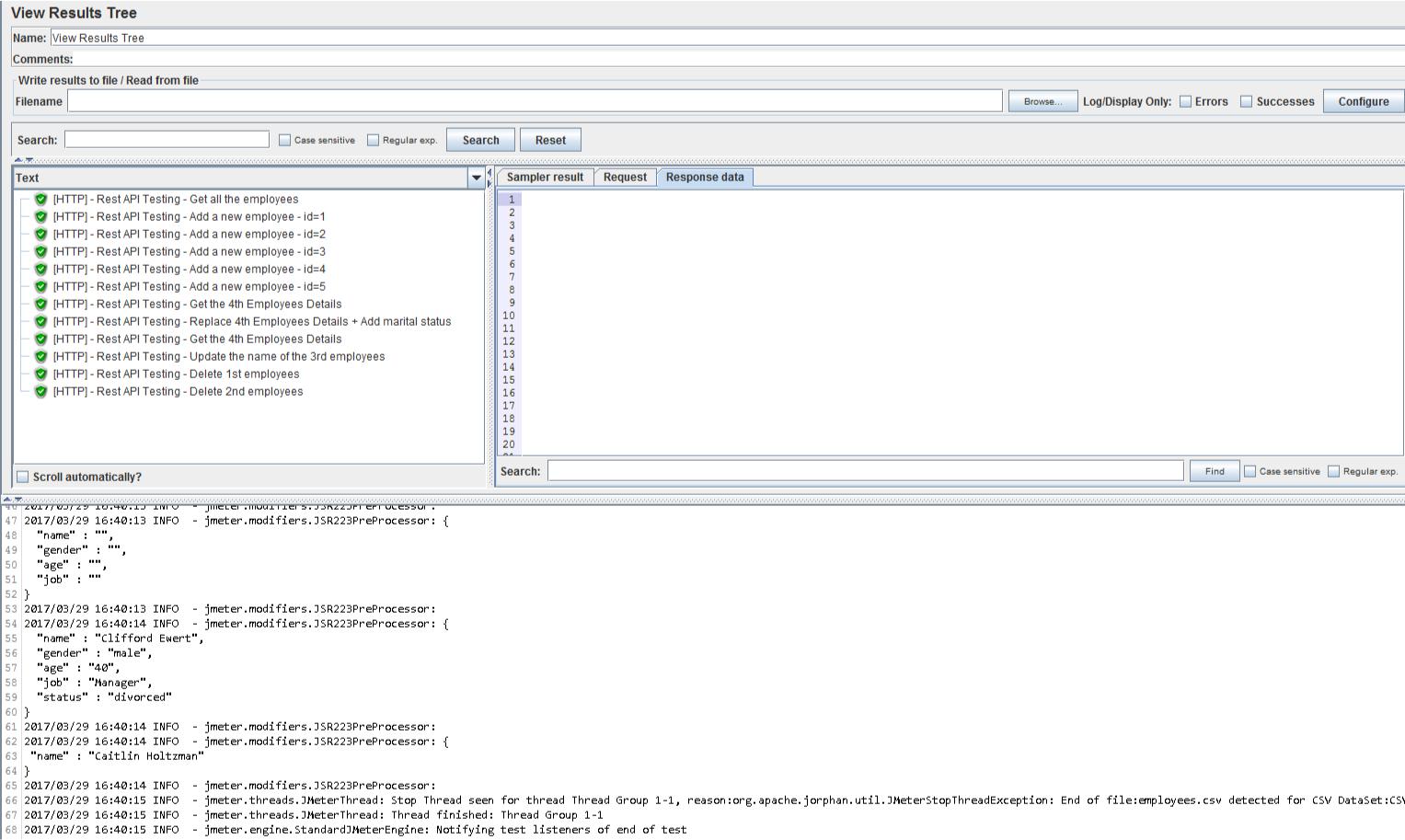
arg.setAlwaysEncoded(false);

sampler.getArguments().addArgument(arg);

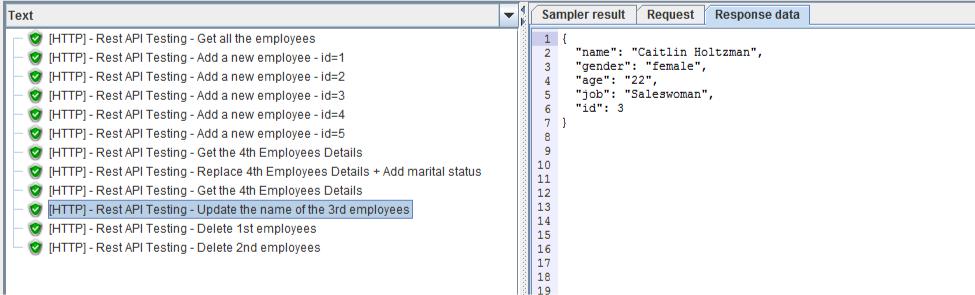
### 2.3.6. Thêm 1 Assertion:



### 2.3.7. Run Test Plan:



Kiểm tra response data:



Kiểm tra kết quả trên API:

