

DucPhuc DEV @NHDPPhucIT

Follow

★ 399

👤 27

✍ 15

Published Jun 3rd, 2019 10:53 AM - 5 min read

👁 3.7K 💬 0 🔗 1

NodeJS Bài 4: Query Parameters

[JavaScript](#) [Node.js](#) [query](#) [Search](#) [Express](#)

...

[Series NodeJS căn bản cho người mới bắt đầu](#)

1. Query parameters là gì?

Hôm nay chúng ta sẽ tìm hiểu về query parameters trong NodeJS. Vậy nó là gì?

Query parameters là một chuỗi truy vấn được client gửi lên server. Server sẽ nhận các thông tin này để xử lý và trả về một kết quả phù hợp với truy vấn được gửi lên

2. Query trong URL

Các chuỗi truy vấn có thể được đính kèm trong một URL. Trong URL, các truy vấn sẽ bắt đầu từ sau dấu **?**, mỗi truy vấn là một cặp {key:value}, các cặp ngăn cách nhau bởi kí tự **&** tạo thành một object gọi là param và được gửi lên server. Ví dụ:

Với URL: <http://example.com/users?name=viblo&age=20> sẽ được chuyển thành object là:

```
{
  name: "viblo",
  age: "20"
}
```

3. Xây dựng chức năng tìm kiếm

3.1. Xây dựng chức năng tìm kiếm user theo tên



Let's register a Viblo Account to get more interesting posts.

[Login](#)[Register](#)

↑ +3 ↓



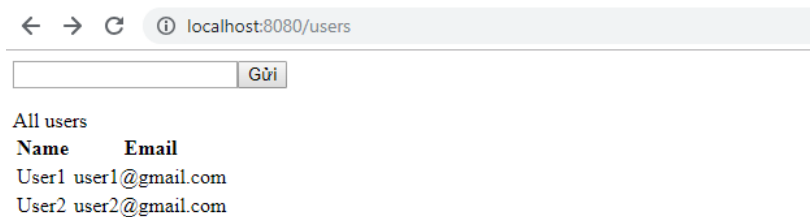
```
nodeapp/app.js

app.get('/users/search', (req, res) => {
  // search and return here
})
```

Ở trên mình đã viết hàm theo chuẩn [ES6](#), các bạn có thể tìm hiểu thêm nhé, hoặc viết như các bài học trước cũng được

Tạo một form HTML để search, mình sẽ tạo tại trang index của users

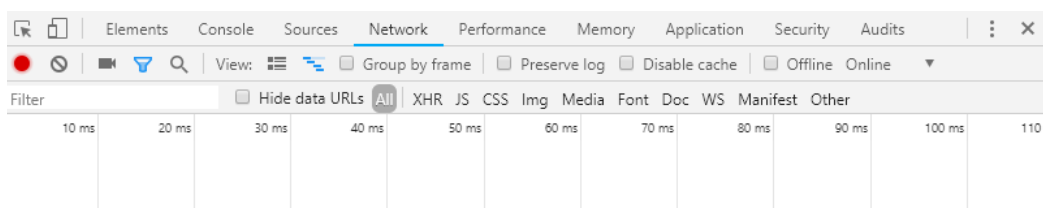
```
form.seach-form(action="/users/search")
input(name="name", type="text", placeholder="name")
input(type="submit")
```



Nhắc lại một chút kiến thức cũ về HTML

- Thuộc tính action quy định đường dẫn mà dữ liệu của form sẽ được gửi đến
- Với mỗi thẻ input, thuộc tính name sẽ tương ứng với key, giá trị của mỗi trường input tương ứng value trong mỗi cặp query

Hãy thử nhập một cái tên nào đó, ví dụ "User1", Inspect trang web và chọn tab Network, bạn sẽ thấy một request được gửi lên

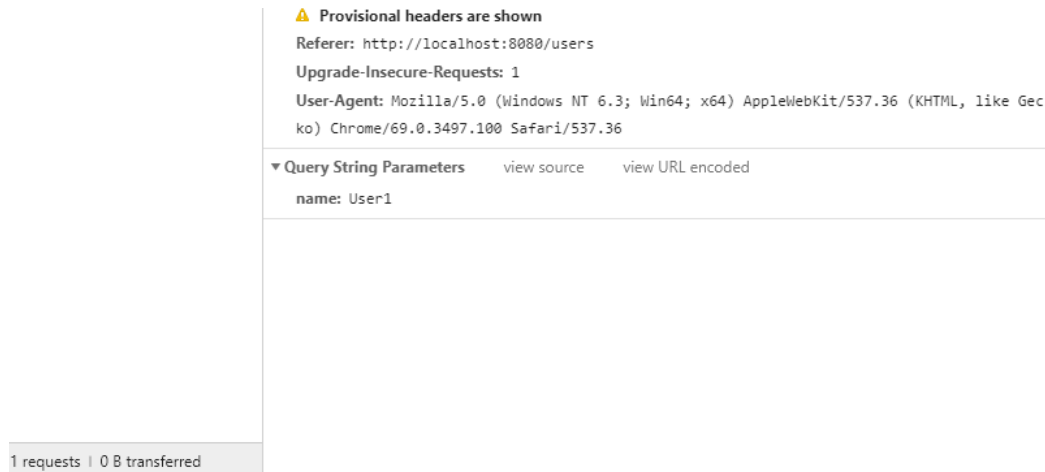


Let's register a Viblo Account to get more interesting posts.

Register

↑ +3 ↓





Bây giờ, ta sẽ xem server đã nhận được request này chưa nhé!!! Đối tượng req cung cấp cho ta một thuộc tính **query** để lấy được query mà client gửi lên. Hãy thử in nó ra nhé!!!

```
nodeapp/app.js

app.get('/users/search', (req, res) => {
  console.log(req.query);
})
```

Tại cửa sổ cmd, bạn sẽ thấy kết quả như sau:

```
{name: 'User1'}
```

Như vậy là server đã nhận được, Bây giờ ta sẽ thực hiện tìm kiếm user. Ở bài trước, mình đã khai báo một mảng array gồm các user trong route '/users'. Để các route khác có thể sử dụng, ta sẽ đưa array này ra ngoài. Mã nguồn file *app.js* bây giờ sẽ như sau:



Let's register a Viblo Account to get more interesting posts.



Login

Register

↑ +3 ↓



```
nodeapp/app.js

const express = require('express'); // Require module express vào project
const app = express(); // Tạo một app mới
const port = 8080; // Định nghĩa cổng để chạy ứng dụng NodeJS của bạn trên server

app.set('views', './views'); // Thư mục views nằm cùng cấp với file app.js
app.set('view engine', 'pug'); // Sử dụng pug làm view engine

var users = [
  {name: "User1", email: "user1@gmail.com"},
  {name: "User2", email: "user2@gmail.com"}
];

app.get('/', function(req, res){
  res.send("<h2>This is my first app</h2>");
})

app.get('/users', function(req, res){
  res.render('users/index',{
    users: users
  });
})

app.get('/users/search', (req,res) => {
  console.log(req.query);
})

app.listen(port, function(){
  console.log('Your app running on port '+ port);
})
```

Tiếp tục, ta sẽ xử lý việc tìm kiếm. Việc này bây giờ trở nên đơn giản, vì các bạn đã học JavaScript, và đã biết cách tìm "một chuỗi con trong chuỗi cha" rồi. Nên mình sẽ không giải thích chi tiết nữa nhé!!! Phần này các bạn có thể tìm hiểu một cách dễ dàng

```
nodeapp/app.js

app.get('/users/search', (req,res) => {
  var name_search = req.query.name // lấy giá trị của key name trong query parameters gửi lên

  var result = users.filter( (user) => {
    // tìm kiếm chuỗi name_search trong user name.
    // Lưu ý: Chuyển tên về cùng in thường hoặc cùng in hoa để không phân biệt hoa, thường khi tìm kiếm
    return user.name.toLowerCase().indexOf(name_search.toLowerCase()) !== -1
  })

  res.render('users/index', {
    users: result // render lại trang users/index với biến users bây giờ chỉ bao gồm các kết quả phù
  });
})
```

Như vậy là hoàn thành việc tìm kiếm và trả về kết quả. Ta sẽ hiệu chỉnh lại mã HTML một chút ứng với 2 trường hợp

- Nếu biến users có length > 0, tức là tồn tại user, thì hiển thị các user đó
- Ngược lại hiển thị "No user to display"



Let's register a Viblo Account to get more interesting posts.



Login

Register

↑ +3 ↓



```
nodeapp/views/users/index.pug
form.search-form(action="/users/search")
  input(name="name", type="text", placeholder="name")
  input(type="submit")

.users
  h2 All users
  if users.length > 0
    table
      thead
        tr
          th Name
          th Email
      tbody
        each user in users
          tr
            td= user.name
            td= user.email
  else
    p No user to display
```

Xong!!! Các bạn hãy thử test chức năng này nhé


The screenshot shows a web browser at the URL `localhost:8080/users/search?name=user1`. Below the address bar is a search form with an input field containing "user1" and a "Gửi" (Submit) button. Below the form, the heading "All users" is displayed. Underneath, there is a table with two columns: "Name" and "Email". The table contains one row with the values "User1" and "user1@gmail.com".

3.2 Xây dựng chức năng tìm kiếm user theo tên và theo tuổi

Ta sẽ thêm một trường tuổi vào array users

```
nodeapp/app.js
var users = [
  {name: "User1", email: "user1@gmail.com", age: 31},
  {name: "User2", email: "user2@gmail.com", age: 20},
  {name: "User1", email: "user1.2@gmail.com", age: 25}
];
```

Ta sẽ thấy có 2 user cùng có name là "User1", nhưng tuổi (age) khác nhau. Trường email là trường khóa chính, để

 Let's register a Viblo Account to get more interesting posts.



↑ +3 ↓

Register



```
nodeapp/views/users/create.pug

form.search-form(action="/users/search")
  input(name="name", type="text", placeholder="name")
  input(name="age", type="text", placeholder="age")
  input(type="submit")

.users
  h2 All users
  if users.length > 0
    table
      thead
        tr
          th Name
          th Email
          th Age
      tbody
        each user in users
          tr
            td= user.name
            td= user.email
            td= user.age
  else
    p No user to display
```

Chỉnh lại chức năng tìm kiếm (thêm điều kiện theo tuổi)

```
nodeapp/app.js

app.get('/users/search', (req,res) => {
  var name_search = req.query.name // lấy giá trị của key name trong query parameters gửi lên
  var age_search = req.query.age // lấy giá trị của key age trong query parameters gửi lên
  var result = users.filter( (user) => {
    // tìm kiếm chuỗi name_search trong user name.
    // Lưu ý: Chuyển tên về cùng in thường hoặc cùng in hoa để không phân biệt hoa, thường khi tìm ki
    return user.name.toLowerCase().indexOf(name_search.toLowerCase()) !== -1 && user.age === parseInt
  })
```

f t ...

Related

[Google Maps API JavaScript Services \(Places, Directions, Geocoder\).](#)

[NamNV609](#)

5 min read

👁 2999 🗣 5 💬 5 ⬆ 0

[Cookie,AJAX JavaScript and how learn ????](#)

[Nguyen The Vinh](#)

7 min read

👁 157 🗣 2 💬 0 ⬆ 1

[\[Trở lại cơ bản\] DOM attribute và property.](#)

[Hoàng Hà Phạm](#)

6 min read

👁 199 🗣 2 💬 0 ⬆ 0

[Filter trong AngularJs](#)

[Minh Do](#)

3 min read

👁 981 🗣 1 💬 0 ⬆ 0



Let's register a Viblo Account to get more interesting posts.



Login

Register

↑ +3 ↓



[NodeJS Bài 12: Static file \(Tập tin tĩnh\) trong NodeJS](#)[DucPhuc DEV](#)

5 min read

👁 332 🗒 0 💬 0 ⬆ 2

[Node JS Bài 11: Template layout](#)[DucPhuc DEV](#)

3 min read

👁 293 🗒 1 💬 0 ⬆ 2

[NodeJS Bài 10: Tích hợp Bootstrap](#)[DucPhuc DEV](#)

2 min read

👁 649 🗒 2 💬 0 ⬆ 5

[NodeJS Bài 9: Giới thiệu Nodemon Package](#)[DucPhuc DEV](#)

2 min read

👁 663 🗒 1 💬 0 ⬆ 1

Comments

[Login to comment](#)

RESOURCES

[Posts](#)[Questions](#)[Videos](#)[Discussions](#)[Tools](#)[System Status](#)[Organizations](#)[Tags](#)[Authors](#)[Recommend System](#)[Machine Learning](#)

SERVICES

[Viblo Code](#)[Viblo CV](#)[CTF Viblo CTF](#)

MOBILE APP



LINKS



© 2020 Viblo. All rights reserved.

[About Us](#)[Feedback](#)[Help](#)[FAQs](#)[RSS](#)[Terms](#)

Let's register a Viblo Account to get more interesting posts.

[Login](#)[Register](#)

↑ +3 ↓

