Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

Lập trình ứng dụng web bằng NodeJS

Bài 5. *Express Framework*

http://csc.edu.vn/

2017

Nội dung

1. **Express Framework là gì?**
2. Cài đặt Express Framework
3. Routing là gì?
4. Routing trong Express
5. Middleware là gì?
6. Tại sao phải sử dụng Middleware
7. Tạo và sử dụng Middleware

1. Express Framework là gì?

- Express là một Framework đơn giản, linh hoạt, hữu ích hỗ trợ cho việc xây dựng website, ứng dụng trên điện thoại bằng NodeJS

- Express cung cấp cho chúng ta các thư viện hữu ích giúp xây dựng website nhanh chóng, hiệu quả. Nhưng không thay thế tất cả các tính năng đặc biệt, nổi bật trong NodeJS.

- Và từ Express đã hỗ trợ xây dựng ra rất nhiều Framework hỗ trợ them cho NodeJS như: **Sails**, **Bottr**, **Hydra-Express**, **LEAN-STACK**,…

Nội dung

1. Express Framework là gì?
2. **Cài đặt Express Framework**
3. Routing là gì?
4. Routing trong Express
5. Middleware là gì?
6. Tại sao phải sử dụng Middleware
7. Tạo và sử dụng Middleware

2. Cài đặt Express Framework

- **Bước 1: Tạo thư mục đồ án**

- **Bước 2: tạo file cấu hình package.json**
  - Dùng lệnh “npm init” để bắt đầu chạy quá trình tạo file cấu hình package.json
  - Lúc này chương trình sẽ tự động yêu cầu nhập các thông số cho cấu hình, ví dụ như: name(tên đồ án), version (phiên bản),…
  - Chú ý bước quan trọng nhất là đặt file js chạy mặc định của ứng dụng khi yêu cầu nhập “entry point:(index.js)”.

2. Cài đặt Express Framework

- Cài đặt xong file cấu hình package.json sẽ có cấu trúc như hình:

```
{
  "name": "bai_5",
  "version": "1.0.0",
  "description": "test express",
  "main": "app.js",
  "scripts": {
    "test": "run"
  },
  "repository": {
    "type": "git",
    "url": "git"
  },
  "keywords": [
    "123"
  ],
  "author": "Hung Nguyen",
  "license": "ISC"
}
```

2. Cài đặt Express Framework

❑ Bước 3 gõ lệnh cài đặt Express:

- Có thể dung lệnh:
  - "npm install express --save" hoặc "npm install express".
  - Với lệnh –save thì khi cài đặt Express xong thì sẽ được ghi chú đã cài đặt Express vào mục dependencies trong file package.json. Như hình:

```
"dependencies": {
  "express": "^4.15.2"
}
```

Nội dung

1. Express Framework là gì?
2. Cài đặt Express Framework
3. **Routing là gì?**
4. Routing trong Express
5. Middleware là gì?
6. Tại sao phải sử dụng Middleware
7. Tạo và sử dụng Middleware

3. Routing là gì?

- Route có nghĩa là tuyến đường, Routing là định tuyến, và khi nghe đến từ này sẽ dễ nhầm lẫn với định tuyến trong Network (Quản trị mạng)

- Route trong Express cũng tương tự như trong các framework của các ngôn ngữ khác, nắm vai trò chỉ đường cho yêu cầu (request) đi đến phương thức nào chịu trách nhiệm xử lý yêu cầu phù hợp.

Nội dung

1. Express Framework là gì?
2. Cài đặt Express Framework
3. Routing là gì?
4. **Routing trong Express**
5. Middleware là gì?
6. Tại sao phải sử dụng Middleware
7. Tạo và sử dụng Middleware

4. Routing trong Express

Các loại Route cơ bản trong Express:

- Route phương thức GET
- Route phương thức POST
- Route phương thức PUT
- Route phương thức DELETE
- Route tất cả phương thức

4.1 Route cơ bản

❑ Cú pháp của route phương thức GET:

Tên_đối_tượng_express.get("đường dẫn", hàm xử lý)

- Ví dụ:

```
app.get("/", function(req, res){
    res.send("trang chủ");
});
```

❑ Cú pháp của route phương thức POST:

- Tương tự như cú pháp của phương thức get, thay đổi hàm get thành hàm post.

- Ví dụ:

```
app.post("/them-san-pham-moi", function(req, res){
    res.send("vào trang thêm sản phẩm");
});
```

4.1 Route cơ bản

- Riêng hai phương thức PUT, DELETE trong Route thường chỉ sử dụng để làm service API.
- Với phương thức PUT thường được dùng làm API service cập nhật dữ liệu.
- Phương thức DELETE thì thường dùng để làm API service xóa dữ liệu
- Chúng ta sẽ nhắc lại 2 loại route này khi đến bài Restful API.

4.1 Route cơ bản

- Để route toàn bộ các phương thức mà NodeJS hỗ trợ cho website (bao gồm 4 phương thức trên), ta sử dụng route all trong NodeJS, với ví dụ sau:

```
app.all("/ds-san-pham", function(req, res){
    res.send("tất cả phương thức đều vào được");
})
```

4. 2 Demo xây dựng Route và chạy server HTTP

4.3 Route Paths trong NodeJS

- Route Path: đưa ra một số yêu cầu bắt buộc đối với đường dẫn request nhận từ phía người dùng trong NodeJS.

- Route Path giúp developer dễ dàng tạo một route có một số ràng buộc nhất định mà nếu viết bằng một số ngôn ngữ khác sẽ tương đối khó khăn.

4.3 Route Paths trong NodeJS

- **Ký tự ? trong Route Path NodeJS nghĩa là có thể có hoặc không một ký tự nào đó.**

- **Sẽ dễ dàng hiểu hơn về Route Path với ví dụ sau:**

```
app.get("/ab?c", function(req, res){
    res.send("Route Path với ký tự ?");
});
```

  - URL đúng với route này sẽ là: “/abc” hoặc “/ac”

4.3 Route Paths trong NodeJS

- Với ký tự ? nhưng áp dụng cho có thể có hoặc không một chuỗi ký tự, với ví dụ:

```
app.get("/test(xem)?ne", function(req, res){
    res.send("Route Path một chuỗi với ký tự ?");
});
```

- Trường hợp này URL phù hợp sẽ là “testxemne” hoặc là “testne”.

4.3 Route Paths trong NodeJS

- Route Paths: với ký tự + đóng vai trò đại diện cho tất cả ký tự giống với ký tự phía trước nó.

- Ví dụ:

```
app.get("/lap-ky-tu-m+", function(req, res){
    res.send("Route Path với ký tự +");
});
```

- Với ví dụ, URL phù hợp sẽ là chuỗi “lap-lai-ky-tu-m”, với ký tự m trong URL có thể lặp lại bao nhiêu cũng được

4.3 Route Paths trong NodeJS

- Tương tự với ký tự ?, ký tự + cũng có thể lặp cho 1 chuỗi như ví dụ sau:

```
app.get("/lap-chuoi-(em)+", function(req, res){
    res.send("Route Path một chuỗi với ký tự +");
});
```

- Ở đây, URL phù hợp sẽ là “lap-chuoi-em”, với chuỗi “em” có thể lặp n lần.

4.3 Route Paths trong NodeJS

- Route Paths với ký tự *, sẽ có ý nghĩa đại diện cho một chuỗi bất kỳ trong chuỗi URL và không bị giới hạn số lượng ký tự.

- Ví dụ:

```
app.get("/chuoi-*-ne", function(req, res){
    res.send("Route Path với ký tự *");
});
```

- URL phù hợp có thể là: “chuoi-123-ne”, với vị trí ký tự “123” có thể là bất kỳ chuỗi nào khác.

4.4 Route Parameters trong NodeJS

- **Route Parameters có trách nhiệm lấy các parameters (có các biến hiển thị trên đường dẫn URL Friendly).**

- **Ví dụ URL:**

  "/12-lap-top-asus/9-asus-rog-g751jy-dh72x"

  - Với các số phía trước tên chính là ID mà chúng ta cần lấy: "12" ID của loại laptop, "9" id của sản phẩm laptop asus rog.

4.4 Route Parameters trong NodeJS

- Ví dụ code thực tế:

```
app.get("/:id_loai_sp-:alias_loai_sp/:id_sp-:alias_sp", function(req, res){
    res.send(req.params);
});
```

- Kết quả:

```
{"id_loai_sp":"12","alias_loai_sp":"lap-top-
asus","id_sp":"9","alias_sp":"asus-rog-g751jy-dh72x"}
```

- Lúc này chúng ta thấy dễ dàng sử dụng các biến lấy từ URL Parameter thông qua đối tượng request.

Nội dung

1. Express Framework là gì?
2. Cài đặt Express Framework
3. Routing là gì?
4. Routing trong Express
5. **Middleware là gì?**
6. Tại sao phải sử dụng Middleware
7. Tạo và sử dụng Middleware

5.1 Middleware trong công nghệ phần mềm

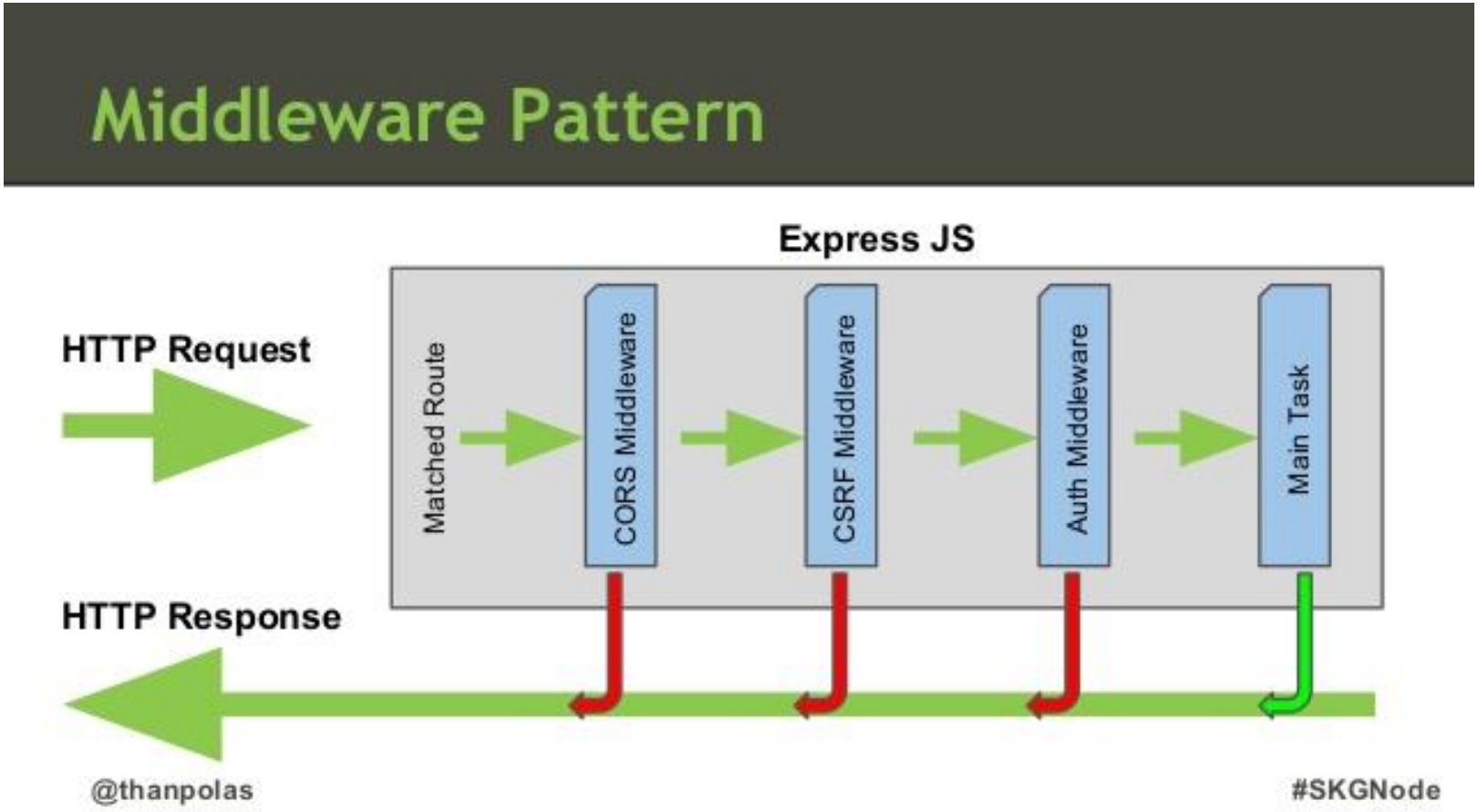
- Middleware trong ngành công nghệ phần mềm được định nghĩa là một phần mềm có nhiệm vụ làm cầu nối (bridge).
- Cung cấp các dịch vụ từ phía hệ điều hành đến các ứng dụng, giúp các ứng dụng có thể tương tác với các thành phần được hệ điều hành cho phép.
- Middleware được coi là chất kết dính dữa các phần mềm với nhau

5.2 Middleware trong Express Framework?

- Với tư tưởng chung là cầu nối giữa tương tác của người dùng và phần nhân của hệ thống, trong lập trình Web, Middleware sẽ đóng vai trò trung gian giữa **request/response** (tương tác với người dùng) và các xử lý logic bên trong web server.

- Do đó, Middleware trong các Framework lập tình Web (Django, Rails, ExpressJS), sẽ là các hàm được dùng để tiền xử lý, lọc các request trước khi đưa vào xử lý logic hoặc điều chỉnh các response trước khi gửi về cho người dùng.

5.2 Middleware trong Express Framework?



Nội dung

1. Express Framework là gì?
2. Cài đặt Express Framework
3. Routing là gì?
4. Routing trong Express
5. Middleware là gì?
6. **Tại sao phải sử dụng Middleware**
7. Tạo và sử dụng Middleware

6. Tại sao phải sử dụng Middleware

- Một hệ thống nhất quán, không phải viết lặp lại các code được sử dụng nhiều lần cho một số trang khác nhau
- Dễ dàng hiệu chỉnh khi hệ thống thay đổi quá trình đăng nhập, kiểm tra hợp lệ,...
- Các thư viện bên thứ 3 cũng đơn giản hóa việc đưa vào sử dụng, mà không làm thay đổi tính nhất quán của ứng dụng hiện hành.

Nội dung

1. Express Framework là gì?
2. Cài đặt Express Framework
3. Routing là gì?
4. Routing trong Express
5. Middleware là gì?
6. Tại sao phải sử dụng Middleware
7. **Tạo và sử dụng Middleware**

7. Tạo và sử dụng Middleware

- **Trong Express có 5 loại Middleware:**
  - Application-level Middleware (Middleware cấp ứng dụng)
  - Router-level Middleware (Middleware cấp điều hướng)
  - Error-handling middleware (Middleware xử lý lỗi)
  - Buildt-in Middleware (Middleware sẵn có)
  - Third-party Middleware (Middleware bên thứ 3)

7.1 Application-level Middleware (Middleware cấp ứng dụng)

- Với Middleware cấp ứng dụng, chúng ta sẽ có ví dụ như sau:

```
app.get("/", function(req, res, next){
    console.log("Chạy Middleware nè");
    next();
})
```

- Khi người dùng vào đường dẫn "/" trước khi vào route xử lý hiển thị trang sẽ đi vào Middleware này trước.

7.1 Application-level Middleware (Middleware cấp ứng dụng)

- Thực hiện demo đơn giản Middleware cấp ứng dụng

7.1 Application-level Middleware (Middleware cấp ứng dụng)

- Để gọi nhiều Middleware lần lượt theo thứ tự thì dùng dấu “,” ngăn cách thêm một Middleware kế tiếp với ví dụ như sau:

```
app.get("/", function(req, res, next){
    console.log("Chạy Middleware nè");
    next();
},
function(req, res, next){
    console.log("Chạy Middleware thứ 2!");
    next();
})
```

7.2 Router-level Middleware

- Chức năng thì tương tự như Application-level Middleware, nhưng code sẽ có thể rõ ràng dễ hiểu hơn, đặc biệt phù hợp với việc code chia thành từng module nhỏ.

- Lưu ý phải tạo một đối tượng Router để điều khiển như sau:

```
route = express.Router();
```

7.2 Router-level Middleware

- Ví dụ sau, sẽ giúp nắm rõ cách tạo Router-level Middleware:

```
route.use("/lien-he", function(req,res,next){
    console.log("Route-level Middlware đã chạy");
    next();
});
```

7.2 Router-level Middleware

- Thực hiện demo đơn giản Middleware cấp ứng dụng

7.3 Error-handling middleware

- **Middleware này phục vụ cho việc xử lý lỗi trong quá trình người dùng truy cập vào website.**

- **Tuy cách viết code cũng gần như tương tự các Middleware ở trên nhưng có một vài điểm chú ý sau:**
  - Error-handling Middleware sẽ có 4 tham số đầu vào chứ không phải 3 như 2 Middleware trên. Đó là: err, req, res, next, lưu ý khi tạo Middleware phải đúng thứ tự các tham số đầu vào.
  - Nếu không có 4 tham số đầy đủ thì Error-handling Middleware sẽ có thể trở thành 2 Middleware ở trên
  - Không cần truyền tham số đường dẫn khi app gọi Middleware này

7.3 Error-handling middleware

- Ví dụ bắt lỗi sau sẽ giúp các bạn nắm rõ hơn:

```
app.use(function(err,req,res,next){
    res.status(500)
    res.send(err);
});
```

- Ví dụ này sẽ báo lỗi khi server gặp một số vấn đề nên không chạy website nữa.

Thảo luận