**Báo cáo**

*Số 18*

**Họ Tên**: Nguyễn Thái Sơn

**Ngày**: 12-12-2019

Task name

* Nội dung tìm hiểu : Tìm hiểu các module common nodejs
* Tiến độ (%) 60%
* Vấn đề gặp phải

**Module trong NodeJs là gì?**

* Các module trong NodeJs giống như các thư viện trong Javascript
* Nó là một tập hợp các chức năng mà muốn đưa vào trong ứng dụng

**Module tích hợp**

* NodeJs có một bộ các module tích hợp mà bạn có thể sử dụng mà không cần cài đặt thêm

**Nhúng Module**

* Để nhúng 1 module vào trong ứng dụng, sử dụng từ khóa require()

vd:

http.createServer(function (req, res) {  
  res.writeHead(200, {'Content-Type': 'text/html'});  
  res.end('Hello World!');  
}).listen(8080);

* Trong NodeJs có các loại Module sau:
  + Core module (là các module mà NodeJs đã xây dựng sẵn)
  + Local module là module do lập trình viên tự định nghĩa
  + Third party module không phải core module cũng không phải local module mà phải tải từ trên hub vd như : express

**Tạo một module riêng**

* Có thể tạo ra các module riêng và đưa vào sử dụng trong ứng dụng

Vd :

exports.myDateTime = function () {  
  return Date();  
};

* Sử dụng từ khóa exports để xuất các thuộc tính và phương thức có bên trong module
* Lưu đoạn code trên với tên “myfirstmodule.js”

**Nhúng module riêng vào trong ứng dụng**

Vd :

var http = require('http');  
**var dt = require('./myfirstmodule');**  
http.createServer(function (req, res) {  
  res.writeHead(200, {'Content-Type': 'text/html'});  
  res.write("The date and time are currently: " + **dt.myDateTime()**);  
  res.end();  
}).listen(8080);

* Lưu ý rằng ./ là định vị trí đến file myfirstmodule, điều đó có nghĩa file module riêng nằm trong cùng một thư mục với code NodeJs
* Lưu file nodejs trên với tên là demo\_module.js.
* Muốn chạy được thì vào trong command gõ lệnh node demo\_module
* Sau đó vào trình duyệt và truy cập [http://localhost:8080](http://localhost:8080/) sẽ xem được ví dụ.

**Module HTTP**

* NodeJs có một module tích hợp có tên là HTTP cho phép nodeJs truyền dữ liệu qua giao thức truyền văn bản Hyper (HTTP)
* Để sử dụng module HTTP hãy sử dụng phương thức require

var http = require('http');

* NodeJs là một máy chủ web
* Module HTTP có thể tạo ra một máy chủ HTTP lắng nghe các cổng máy chủ và trả lời lại cho máy khách
* Sử dụng phương thức createServer() để tạo ra một HTTP server
* Bất kỳ ứng dụng máy chủ web node nào tại một số điểm sẽ phải tạo một đối tượng máy chủ web.

const http = require('http');

const server = http.createServer((request, response) => {

// magic happens here!

});

* Hàm được truyền vào createServer được gọi một lần cho mỗi yêu cầu HTTP được thực hiện đối với máy chủ đó, do đó nó được gọi là hàm xử lý yêu cầu.
* Trong thực tế đối tượng máy chủ được tạo bởi máy chủ tạo là máy chủ sự kiện và những gì thấy ở đây chỉ là cách viết tắt để tạo đối tượng máy chủ và sau đó thêm trình nghe sau :

const server = http.createServer();

server.on('request', (request, response) => {

// the same kind of magic happens here!

});

* Khi xử lý một yêu cầu từ HTTP đến máy chủ, NodeJs gọi hàm xử lý yêu cầu với một vài đối tượng tiện dụng đê xử lý giao dịch, yêu cầu và phản hồi.
* Thêm một tiêu đề cho HTTP: Nếu phản hồi từ máy chủ HTTP được cho là hiển thị dưới dạng HTML bạn nên bao gồm tiêu đề HTTP với loại nội dung chính xác
* Khi xử lý một yêu cầu điều đầu tiên có thể muốn làm là xem xét phương thức và URL, để có thể thực hiện các hành động thích hợp.

var http = require('http');  
http.createServer(function (req, res) {  
**res.writeHead(200, {'Content-Type': 'text/html'});**  res.write('Hello World!');  
  res.end();  
}).listen(8080);

* Đối số đầu tiên của phương thức res.writeHead là mã trạng thái, 200 là tất cả đều ổn, đối số thứ 2 là một đối tượng chứa các tiêu đề phản hồi
* Đọc chuỗi truy vấn: hàm được truyền vào http.createServer() có đối số req đại diện cho yêu cầu từ máy khách, dưới dạng một đối tượng (đối tượng http. IncomingMessage )
* Đối tượng này có một thuộc tính được gọi là “url” chứa một phần của url đi sau tên miền
* var http = require('http');  
  http.createServer(function (**req**, res) {  
    res.writeHead(200, {'Content-Type': 'text/html'});  
    res.write(**req.url**);  
    res.end();  
  }).listen(8080);

**Request body**

* Khi nhận được yêu cầu POST hoặc PUT, phần body yêu cầu có thể quan trọng đối với ứng dụng.
* Lấy dữ liệu body có thể liên quan nhiều hơn một chút so với truy cập các tiêu đề yêu cầu. Đối tượng yêu cầu được chuyển đên một trình xử lý thực hiện giao diện ReadableStream
* Nguồn này có thể được nghe hoặc dẫn ở nơi khác giống như bất kỳ luồng nào khác. Chúng có thể lấy dữ liệu ra khỏi luồng bằng cách lắng nghe các sự kiện “data” và ‘end’ của luồng

**NodeJs File System Module**

* Module hệ thống tệp NodeJs cho phép bạn làm việc với hệ thống tệp trên máy tính của mình
* Để sử dụng module hệ thống file hãy sử dụng phương thức require()

Vd: var fs = require('fs');

* Các phương thức sử dụng phổ biến file system module
  + Read files
  + Create files
  + Update files
  + Delete files
  + Rename files

**Read Files**

* Phương thức fs.readFile() được sử dụng để đọc các tệp trên máy tính của bạn
* Giả sử ta có 1 file HTML sau (nằm trong cùng thư mục với NodeJs)
* <html>  
  <body>  
  <h1>My Header</h1>  
  <p>My paragraph.</p>  
  </body>  
  </html>
* Tạo một file NodeJs với nội dung như sau:

var http = require('http');  
var fs = require('fs');  
http.createServer(function (req, res) {  
**fs.readFile('demofile1.html', function(err, data) {**    res.writeHead(200, {'Content-Type': 'text/html'});  
    res.write(data);  
    res.end();  
  });  
}).listen(8080);

* Lưu file NodeJs trên với tên demo\_readfile.js
* Mở màn hình command gõ lệnh node demo\_readfile và test với đường dẫn [http://localhost:8080](http://localhost:8080/)

## Create Files

* Module hệ thống tệp có các phương thức để tạo tệp mới
  + Fs.appendFile()
  + Fs.open()
  + Fs.writeFile()
* Phương thức fs.appendFile () nối thêm nội dung được chỉ định vào một tệp. Nếu tệp không tồn tại, tệp sẽ được tạo

var fs = require('fs');  
  
fs.appendFile('mynewfile1.txt', 'Hello content!', function (err) {  
  if (err) throw err;  
  console.log('Saved!');  
});

* Phương thức fs.open () lấy “flag” làm đối số thứ hai, nếu flag là “w” cho “ghi” tệp đã chỉ định được mở để ghi. Nếu tệp không tồn tại, một tệp trống được tạo

var fs = require('fs');  
  
fs.open('mynewfile2.txt', 'w', function (err, file) {  
  if (err) throw err;  
  console.log('Saved!');  
});

* Phương thức fs.writeFile() thay thế tệp và nội dung đã chỉ định nếu nó tồn tại. Nếu tệp không tồn tại. một tệp mới, chứa nội dung được chỉ định, sẽ được tạo.

var fs = require('fs');  
  
fs.writeFile('mynewfile3.txt', 'Hello content!', function (err) {  
  if (err) throw err;  
  console.log('Saved!');  
});

**Update Files**

* Module hệ thống tệp có các phương thức để cập nhật tệp
  + Fs.appendFile()
  + Fs.writeFile()
* Phương thức fs.appendFile() nối thêm nội dung được chỉ định vào cuối tệp được chỉ định

var fs = require('fs');  
  
fs.appendFile('mynewfile1.txt', ' This is my text.', function (err) {  
  if (err) throw err;  
  console.log('Updated!');  
});

* Phương thức fs.writeFile() thay thế tệp và nội dung đã chỉ định

var fs = require('fs');  
  
fs.writeFile('mynewfile3.txt', 'This is my text', function (err) {  
  if (err) throw err;  
  console.log('Replaced!');  
});

**Delete Files**

* Để xóa một tệp bằng module hệ thống tệp, hãy sử dụng phương thức fs.unlink()

var fs = require('fs');  
  
fs.unlink('mynewfile2.txt', function (err) {  
  if (err) throw err;  
  console.log('File deleted!');  
});

**Rename Files**

* Để đổi tên tệp bằng module hệ thống tệp hãy sử dụng phương thức fs.rename()
* Phương thức fs.rename() đổi tên tệp đã chỉ định

var fs = require('fs');  
  
fs.rename('mynewfile1.txt', 'myrenamedfile.txt', function (err) {  
  if (err) throw err;  
  console.log('File Renamed!');  
});