Phát triển ứng dụng web

NodeJs
part 1

Nội dung

- Khái niệm NodeJs
- Các module thông dụng
- Express Framework

Nội dung

- **Khái niệm NodeJs**
- Các module thông dụng
- Express Framework

Khái niệm NodeJs

- **NodeJs** là một ***event-based***, ***non-blocking***, ***asynchronous I/O framework***, được xây dựng dựa trên JavaScript V8 engine của Google
- NodeJS là một mã nguồn mở được sử dụng rộng rãi, NodeJS có thể chạy trên nhiều nền tảng hệ điều hành khác nhau từ WIndow cho tới Linux, OS X. NodeJS cung cấp các thư viện phong phú ở dạng Javascript Module khác nhau giúp đơn giản hóa việc lập trình và giảm thời gian ở mức thấp nhất.

Đặc trưng của NodeJs

- **None-blocking I/O**: những task có liên quan về input/output không thể có quyền ngăn chặn những task khác (task I/O là một trong những task tốn tài nguyên nhất, chậm nhất).
- **Thực thi nhanh**: nhờ nền tảng Javascript Engine V8.
- **Đơn luồng nhưng khả năng mở rộng cao:** NodeJs sử dụng một mô hình luồng duy nhất với sự kiện lặp.
- **Không cache**: NodeJS không cache bất kì một dữ liệu nào.

NodeJs - Module

- Giúp quản lý code dễ dàng hơn. Trong hệ thống NodeJs module thì mỗi file được xử lý như 1 module riêng biệt. Có 3 loại module:
  - Module tích hợp
  - Module chia sẻ
  - Module tự tạo

Module - exports

myPI.js

```
module.exports = 3.14156;
```

MyFunc.js

```
module.exports = n => {
    return Math.round(Math.random() * n);
};
```

main.js

```
const pi = require('./myPI');
console.log(`The value of PI is ${pi}`);

const func = require('./myFunc');
const n = 1000;
console.log(`Random < ${n} = ${func(n)}`);
```

Module – exports object

myCalculator_1.js

```
module.exports = {
    add: (a, b) => {
        return a + b;
    },
    subtract: (a, b) => {
        return a - b;
    }
};
```

myCalculator_2.js

```
exports.add = (a, b) => {
    return a + b;
};

exports.subtract = (a, b) => {
    return a - b;
};
```

main.js

```
const cc1 = require('./myCalculator_1');
const cc2 = require('./myCalculator_2');
const a = 6, b = 11;
console.log(`${a} + ${b} = ${cc1.add(a, b)} / ${cc1.add(a, b)}`);
console.log(`${a} - ${b} = ${cc1.subtract(a, b)} / ${cc2.subtract(a, b)}`);
```

Node Package Manager

- Node Package Manager (NPM): là trình quản lý **package** cho NodeJs. Một package chứa tất cả các file cần thiết cho 1 module.
- NPM cung cấp 2 chức năng chính:
  - Kho package (free): *npmjs.com*
  - Tiện ích dòng lệnh (command) để cài đặt các module, package Node.js quản lý phiên bản và sự phụ thuộc của các package node.js

NPM - Command

- Cài đặt toàn cục

```
npm install -g <package name>
```

- Cài đặt chỉ phục vụ cho việc phát triển

```
npm install --save-dev <package name>
```

- Lựa chọn phiên bản muốn cài đặt

```
npm install <package name>@<version>
```

- Gỡ bỏ

```
npm uninstall <package name>
```

- Cập nhật

```
npm update <package name>
```

Nội dung

- *Khái niệm NodeJs*
- **Các module thông dụng**
- Express Framework

Module - HTTP

```
const http = require('http');
const server = http.createServer((req, res) => {
    res.statusCode = 200;
    res.setHeader('Content-Type', 'text/plain; charset=UTF-8');
    res.end('Hello, World!\n');
});

const port = 3000;
const hostname = '127.0.0.1';
server.listen(port, hostname, () => {
    console.log(`Server running at http://${hostname}:${port}/`);
});
```

Module - URL

- Giúp phân giải và phân tích URL.

Property	Mô tả
href	đường dẫn URL
host	tên server và port
hostname	tên server
port	port
path	phần đường dẫn URL gồm cả query
pathname	phần đường dẫn URL
auth	
protocol	
query	phần query
search	

Ví dụ

```
const http = require('http');
const url = require('url');
const cc = require('./myCalculator_2');
const server = http.createServer((req, res) => {
    res.writeHead(200, {'Content-Type': 'text/plain; charset=UTF-8'});
    const q = url.parse(req.url, true).query;
    const a = parseInt(q.a); b = parseInt(q.b);
    res.end(`${a} + ${b} = ${cc.add(a, b)}`);
});

const port = 3000;
const hostname = '127.0.0.1';
server.listen(port, hostname, () => {
    console.log(`Server running at http://${hostname}:${port}/`);
});
```

http://127.0.0.1:3000/?a=12&b=23 → 12 + 23 = 35

Module – xử lý file

- Node.js cung cấp module **fs** để có thể truy cập hệ thống file. Module **fs** chịu trách nhiệm cho tất cả các hoạt động I/O đồng bộ hoặc bất đồng bộ.
- Module **path** cung cấp rất nhiều chức năng hữu ích để truy cập và tương tác với hệ thống file.

Module - EventEmitter

- Module **events** cung cấp lớp đối tượng **EventEmitter** cho phép kết hợp lập trình hướng sự kiện trong NodeJs.

```
const events = require('events');
const eventEmitter = new events.EventEmitter;

function addNumHandler(n) {
    console.log(n);
}
eventEmitter.on('addNum', addNumHandler);

for(let i=0; i< 5; i++) {
    eventEmitter.emit('addNum', i * i);
}
eventEmitter.removeAllListeners('addNum');
```

Nội dung

- *Khái niệm NodeJs*
- *Các module thông dụng*
- **Express Framework**

Express framework

- **Express** là một framework web Node.js phổ biến nhất và là thư viện cơ bản cho một số framework Node.js phổ biến khác.
- Express cung cấp:
  - Thao tác xử lý với các request HTTP với đầy đủ các method, address thông qua cơ chế **router**.
  - Tích hợp Views engine (Template engine) để xây dựng hiển thị dễ dàng hơn.
  - Thêm các yêu cầu xử lý trung gian “middleware” tại bất kỳ điểm nào luồng xử lý yêu cầu.

Ví dụ

```
const express = require('express');
const app = express();
const cc = require('./myCalculator_2');
app.get('/', (req, res) => {
    const a = parseInt(req.query.a); b = parseInt(req.query.b);
    res.end(`${a} + ${b} = ${cc.add(a, b)}`);
});
const port = 3000;
const hostname = '127.0.0.1';
app.listen(port, hostname, () => {
    console.log(`Server running at http://${hostname}:${port}/`);
});
```

Express – basic routing

- **Routing** tức là cách ứng dụng hồi đáp cho yêu cầu của client đến 1 endpoint cụ thể đó là 1 URI với HTTP request method.
- Cơ bản xử lý thông qua cấu trúc

```
app.METHOD(PATH, HANDLER)
```

  - app là express instance
  - METHOD là HTTP request method
  - PATH là path server
  - HANDLER là callback function.

Express – router paths, params

- Có thể sử dụng pattern, regular expression để kết hợp tạo path cho router. Ví dụ:
  - 'ab?cd' → acd, abcd
  - 'ab+cd' → abcd, abbcd, abbbcd,...
  - 'a(bc)?d' → ad, abcd
- Sử dụng params để truyền tham số trong request URL.
  - Route path: */users/:userId/books/:bookId*
  - Request URL: *<hostname>/users/34/books/8989*
  - req.params: { "userId": "34", "bookId": "8989" }

Express – Static resources

- Sử dụng hàm tích hợp sẵn ***express.static*** để cung cấp tài nguyên tĩnh như image, css,...

```
express.static(root, [options])
```

- Ví dụ

```
app.use(express.static(__dirname + '/public'));
```

- Đôi khi cần xác định sử dụng static page thay vì xử lý routing thì cần gọi thiết lập lại route

```
app.use(app.route);
```

Nội dung

- *Khái niệm NodeJs*
- *Các module thông dụng*
- *Express Framework*
- **Bài tập**

Bài tập

- Xây dựng ứng dụng web giúp ghi nhận và phản hồi về việc đăng ký tham gia Hội nghị với các chức năng:
  - Trang chủ giới thiệu thông tin về Hội nghị
  - Trang xác nhận tham gia Hội nghị với các thông tin: Họ tên, Email, Điện thoại và xác nhận tham gia hay không.
  - Trang phản hồi kết quả.