Bài 7
Thao tác được với file & Cơ chế Stream

Module: Web backend development with NodeJS

Mục tiêu

- Trình bày được các phương pháp làm việc với file trong Node.js
- Thực hiện được thao tác với file và thư mục theo phương pháp bất đồng bộ
- Thực hiện được thao tác với file và thư mục theo phương pháp đồng bộ
- Thực hiện được các thao tác đọc và ghi với file text
- Trình bày được cơ chế Stream trong Node.js
- Trình bày được 4 kiểu Stream trong Node.js
- Thực hiện được thao tác với file theo cơ chế Stream

Làm việc với file trong NodeJS

Thảo luận

Làm việc với file

- NodeJS cung cấp các module để có thể làm việc với file trong hệ thống
- Để sử dụng module thao tác với file, sử dụng phương thức `require()`

```
const fs = require('fs');
```

Các thao tác với file

- Các thao tác với file bao gồm:
  - Read - Đọc file
  - Create – Tạo file
  - Update – Cập nhật file
  - Delete – Xoá file
  - Rename – Đổi tên file

Read – Đọc file

- Phương thức `fs.readFile()` được sử dụng để đọc các tệp trên máy tính của bạn.
- Ví dụ:

```
const http = require('http’);
const 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);
    return res.end();
  });
}).listen(8080);
```

Create – Tạo mới file

- Có thể sử dụng các phương thức sau tạo mới file
  - fs.appendFile()
  - fs.open()
  - fs.writeFile()

Create – Tạo mới file

- Phương thức `fs.appendFile()`nối nội dung được chỉ định vào một tệp. Nếu tệp không tồn tại, tệp sẽ được tạo:

```
const fs = require('fs');

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

Create – Tạo mới file

- Phương thức `fs.open()` nhận "cờ" làm đối số thứ hai, nếu cờ là "w" cho ”ghi", tệp được chỉ định sẽ được mở để ghi. Nếu tệp không tồn tại, một tệp trống sẽ được tạo:

```
const fs = require('fs');

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

Create – Tạo mới file

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

```
const fs = require('fs');

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

Update – Cập nhật file

- Để cập nhật file sử dụng các phương thức
  - fs.appendFile()
  - fs.writeFile()

Update – Cập nhật file

- Để cập nhật file sử dụng các phương thức
  - fs.appendFile()
  - fs.writeFile()

Delete – Xoá file

- Để xoá file sử dụng phương thức `fs.unlink()`

```
const fs = require('fs');

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

Rename – Đổi tên file

- Phương thức `fs.rename()` đổi tên tệp được chỉ định:

```
var fs = require('fs');

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

Cơ chế Stream

Thảo luận

Khái niệm Stream

- Là một cách để xử lý việc đọc/ghi tệp, truyền thông mạng hoặc bất kỳ loại trao đổi thông tin đầu cuối nào một cách hiệu quả.
- Thay vì đọc toàn bộ file, Stream đọc từng đoạn một, xử lý nội dung của nó mà không cần lưu tất cả vào bộ nhớ.

Các loại Stream

- **Readable stream:** Là loại stream mà từ đó bạn có thể nhận và đọc dữ liệu theo cách có thứ tự.
- **Writable stream:** Đây là stream mà bạn có thể gửi dữ liệu nhưng bạn không được phép nhận lại.
- **Duplex stream:** Đây là loại stream vừa có thể đọc được vừa có thể ghi. Vì vậy, bạn có thể gửi và nhận dữ liệu cùng nhau.
- **Transform stream:** Đây là stream được sử dụng để sửa đổi dữ liệu hoặc chuyển đổi dữ liệu khi nó được đọc.

Lợi ích sử dung Stream

- **Hiệu quả bộ nhớ** : Không cần phải tải một lượng lớn dữ liệu vào bộ nhớ trước khi có thể xử lý nó
- **Hiệu quả về thời gian** : mất ít thời gian hơn để bắt đầu xử lý dữ liệu, vì có thể bắt đầu xử lý ngay khi có, thay vì đợi cho đến khi toàn bộ dữ liệu có sẵn

Tạo một Stream

- Sử dung phương thức createReadStream() của module **fs**

```
const http = require('http');
const fs = require('fs');

const server = http.createServer((req, res) => {
    const stream = fs.createReadStream(`${__dirname}/data.txt`);
    stream.pipe(res);
});
server.listen(3000);
```

Tóm tắt bài học

- NodeJS cung cấp một số cách thức để làm việc với file: Đọc, ghi, tạo mới hay xoá một file.
- Stream là cơ chế xử lý với các file có dữ liệu lớn thông qua cách thức “luồng”

Hướng dẫn

- Hướng dẫn làm bài thực hành và bài tập
- Chuẩn bị bài tiếp: Cơ sở dữ liệu