Đọc ghi File trong Node.js

Trang trước (../nodejs/stream_trong_nodejs.jsp) Trang sau (../nodejs/doi_tuong_toan_cuc_trong_nodejs.jsp)

Trong các chương trước, bạn thấy rằng mình đã sử dụng rất nhiều cú pháp **require("fs")**. Vậy cú pháp để làm gì ? Đây là cú pháp để khai báo **fs** Module để triển khai các hoạt động về File I/O trong Node.js. Cú pháp như sau:

```
var fs = require("fs")
```

Khái niệm Đồng bộ vs Không đồng bộ trong Node.js

Mỗi phương thức trong fs Module có các form đồng bộ và các form không đồng bộ. Các phương thức không đồng bộ nhận một tham số cuối cùng là một hàm callback thực thi khi kết thúc và nhận tham số đầu tiên là một hàm callback để xử lý lỗi. Việc sử dụng các phương thức không đồng bộ là tốt hơn các phương thức đồng bộ, bởi vì các phương thức không đồng bộ không bao giờ khóa trình thực thi chương trình trong khi phương thức đồng bộ thì có.

Ví dụ

Để minh họa hoạt động I/O trong Node.js, đầu tiên bạn tạo **input.txt** có nội dung:

```
VietJack la trang Web huong dan cac bai lap trinh
hoan toan mien phi cho tat ca moi nguoi!!!!!
```

Tạo **main.js**. Như trên đã trình bày, mỗi phương thức của **fs** Module đều có hai form là đồng bộ và không đồng bộ. Để đọc dữ liệu, mình sử dụng phương thức readFile() của form không đồng bộ và readFileSync() của form đồng bộ để đọc dữ liệu. Hai phương thức này nhận tham số đầu tiên là tên file để đọc dữ liệu từ đó.

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

// Phuong thuc doc file khong dong bo
fs.readFile('input.txt', function (err, data) {
   if (err) {
       return console.error(err);
   }
   console.log("Phuong thuc doc file khong dong bo: " + data.toString());
});

// Phuong thuc doc file dong bo
var data = fs.readFileSync('input.txt');
console.log("Phuong thuc doc file dong bo: " + data.toString());

console.log("Ket thuc chuong trinh");
```

Chạy main.js để xem kết quả:

```
$ node main.js
```

Kiểm tra kết quả:

```
Phuong thuc doc file dong bo: VietJack la trang Web huong dan cac bai lap trinh
hoan toan mien phi cho tat ca moi nguoi!!!!!

Ket thuc chuong trinh
Phuong thuc doc file khong dong bo: VietJack la trang Web huong dan cac bai lap trinh
hoan toan mien phi cho tat ca moi nguoi!!!!!
```

Mở một File trong Node.js

Cú pháp

Để mở một file trong chế độ không đồng bộ, bạn sử dụng phương thức open() có cú pháp:

```
fs.open(path, flags[, mode], callback)
```

Tham số

path - Đây là một chuỗi biểu diễn tên file cũng như đường dẫn tới file đó.

flags - Biểu diễn hành vi của file được mở. Tất cả các giá trị có thể sẽ được trình bày trong bảng dưới đây.

mode - Thiết lập chế độ cho file, các chế độ này chỉ được thiết lập khi file đã được tạo. Giá trị mặc định là 0666, tức là readable và writeable.

callback - Hàm callback nhận hai tham số, ví dụ (err, fd).

Các Flag được sử dụng cho hoạt động Đọc/Ghi file trong Node.js

Flag	Miêu tả
r	Mở file để đọc. Xuất hiện Exception nếu file không tồn tại.
r+	Mở file để đọc và ghi. Xuất hiện Exception nếu file không tồn tại.
rs	Mở file để đọc trong chế độ đồng bộ.
rs+	Mở file để đọc và ghi, báo cho Hệ điều hành mở nó trong chế độ đồng bộ.
w	Mở file để ghi. Nếu file không tồn tại, nó sẽ tạo file mới.
wx	Giống 'w' nhưng hoạt động này thất bại nếu file không tồn tại (tức là nó không tạo file mới).
w+	Mở file để đọc và ghi. Nếu file không tồn tại, nó sẽ tạo file mới.
wx+	Giống 'w+' nhưng hoạt động này thất bại nếu file không tồn tại
a	Mở file để append. File sẽ được tạo nếu nó không tồn tại.
ax	Giống 'a' nhưng hoạt động này thất bại nếu file không tồn tại.
a+	Mở file để đọc và append. File sẽ được tạo nếu nó không tồn tại.
ax+	Giống 'a+' nhưng hoạt động này thất bại nếu file không tồn tại.

Ví dụ

Ví dụ sau minh họa cách mở một file để đọc và ghi. Đầu tiên bạn tạo **main.js** có nội dung như dưới đây. Nội dung file khá giống ví dụ trên, bạn chú ý vào phần flag đã sử dụng ở đây.

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

// Hoat dong mo File theo cach thuc khong dong bo
console.log("Chuan bi mo File hien tai!");
fs.open('input.txt', 'r+', function(err, fd) {
   if (err) {
       return console.error(err);
   }
  console.log("File duoc mo thanh cong!");
});
```

Chạy main.js để xem kết quả:

```
$ node main.js
```

Kiểm tra kết quả:

Lấy thông tin File trong Node.js

Cú pháp

Để lấy thông tin về một file trong Node.js, bạn sử dụng phương thức stat() của **fs** Module có cú pháp:

```
fs.stat(path, callback)
```

Chi tiết về tham số

path - Đây là một chuỗi biểu diễn tên file cũng như đường dẫn tới file đó.

callback - Là hàm callback nhận hai tham số (err, stats), trong đó **stats** là một đối tượng của **fs.Stats** được in ra như trong ví dụ sau.

Ngoài các thuộc tính quan trọng được in ra như trong ví dụ sau, lớp **fs.Stats** còn có một số phương thức hữu ích có thể được sử dụng để kiểm tra kiểu file. Đó là:

Phương thức	**Miêu tả**
stats.isFile()	Trả về true nếu đó là một file
stats.isDirectory()	Trả về true nếu đó là một thư mục
stats.isBlockDevice()	Trả về true nếu đó là một Block Device.
stats.isCharacterDevice()	Trả về true nếu đó là một Character Device.
stats.isSymbolicLink()	Trả về true nếu đó là một Symbolic Link.
stats.isFIFO()	Trả về true nếu đó là một kiểu FIFO.
stats.isSocket()	Trả về true nếu đó là một kiểu Socket.

Ví dụ

Sau đây là ví dụ minh họa cách lấy thông tin về một file nào đó. Tạo **main.js** và sử dụng phương thức stat() của **fs** Module đã trình bày ở trên:

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

console.log("Chuan bi lay thong tin File hien tai!");
fs.stat('input.txt', function (err, stats) {
   if (err) {
       return console.error(err);
   }
   console.log(stats);
   console.log("Lay thong tin File thanh cong!");

   // Kiem tra kieu file
   console.log("isFile ? " + stats.isFile());
   console.log("isDirectory ? " + stats.isDirectory());
});
```

Chạy main.js để xem kết quả:

```
$ node main.js
```

Kiểm tra kết quả:

```
Chuan bi lay thong tin File hien tai!
{ dev: 1792,
  mode: 33188,
  nlink: 1,
  uid: 48,
  gid: 48,
  rdev: 0,
  blksize: 4096,
  ino: 4318127,
  size: 97,
  blocks: 8,
  atime: Sun Mar 22 2015 13:40:00 GMT-0500 (CDT),
  mtime: Sun Mar 22 2015 13:40:57 GMT-0500 (CDT),
  ctime: Sun Mar 22 2015 13:40:57 GMT-0500 (CDT) }
Lay thong tin File thanh cong!
isFile ? true
isDirectory ? false
```

Ghi dữ liệu vào File trong Node.js

Cú pháp

Để ghi dữ liệu vào File trong Node.js, bạn có thể sử dụng phương thức writeFile() của **fs** Module như sau:

Phương thức này sẽ ghi đè nếu file đã tồn tại.

Chi tiết về tham số

path - Đây là một chuỗi biểu diễn tên file cũng như đường dẫn tới file đó.

data - Dữ liệu dạng String hoặc Buffer để ghi vào File.

options - Tham số này là một đối tượng giữ {encoding, mode, flag}. Theo mặc định, mã hóa là utf8, mode là giá trị 0666 và flag là 'w'

callback - Hàm callback nhận một tham số là err và được sử dụng để trả về một lỗi nếu xảy ra bất kỳ lỗi nào trong hoạt động ghi

Ví dụ

Ví dụ sau minh họa cách ghi dữ liệu tới một file. Tạo **main.js** có nội dung như sau:

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

console.log("Chuan bi ghi du lieu vao file hien tai");
fs.writeFile('input.txt', 'Hoc Node.js co ban tai VietJack!',  function(err) {
   if (err) {
       return console.error(err);
   }
   console.log("Ghi du lieu vao file thanh cong!");
   console.log("Doc du lieu vua duoc ghi");
   fs.readFile('input.txt', function (err, data) {
      if (err) {
         return console.error(err);
      }
      console.log("Phuong thuc doc file khong dong bo: " + data.toString());
   });
});
```

Chạy main.js để xem kết quả:

```
$ node main.js
```

Kiểm tra kết quả:

```
Chuan bi ghi du lieu vao file hien tai
Ghi du lieu vao file thanh cong!
Doc du lieu vua duoc ghi
Phuong thuc doc file khong dong bo: Hoc Node.js co ban tai VietJack!
```

Đọc dữ liệu từ File trong Node.js

Cú pháp

Để đọc dữ liệu từ một File, bạn sử dụng phương thức read() có cú pháp sau:

```
fs.read(fd, buffer, offset, length, position, callback)
```

Phương thức này sẽ sử dụng tham số fd (viết tắt của File Descriptor) để đọc file. Nếu bạn muốn đọc file bởi sử dụng trực tiếp tên file thì bạn nên sử dụng phương thức khác.

Chi tiết về tham số

fd - Là viết tắt của file descriptor được trả về bởi phương thức fs.open().

buffer - Đây là Buffer, là nơi dữ liệu được ghi vào.

offset - Đây là offset trong Buffer để dữ liệu bắt đầu ghi từ vị trí đó.

length - Một số nguyên xác định số byte để đọc.

position - Một số nguyên xác định nơi bắt đầu đọc từ trong file. Nếu vị trí là null, dữ liệu sẽ được đọc từ vị trí hiện tại của file.

callback - Một hàm callback nhận ba tham số, có dạng (err, bytesRead, buffer).

Ví dụ

Tạo **main.js** có nội dung sau:

```
console.log("Chuan bi mo mot File dang ton tai");
fs.open('input.txt', 'r+', function(err, fd) {
   if (err) {
       return console.error(err);
   }
   console.log("File duoc mo thanh cong!");
   console.log("Chuan bi doc du lieu tu File da mo");
   fs.read(fd, buf, 0, buf.length, 0, function(err, bytes){
      if (err){
         console.log(err);
      }
      console.log(bytes + " bytes read");

      // In so luong byte da doc.
      if(bytes > 0){
         console.log(buf.slice(0, bytes).toString());
      }
   });
});
```

Chạy main.js để xem kết quả:

```
$ node main.js
```

Kiểm tra kết quả:

```
Chuan bi mo mot File dang ton tai
File duoc mo thanh cong!
Chuan bi doc du lieu tu File da mo
97 bytes read
VietJack la trang Web huong dan cac bai lap trinh
hoan toan mien phi cho tat ca moi nguoi!!!!!
```

Đóng File trong Node.js

Cú pháp

Để đóng một file sau khi đã mở, bạn sử dụng phương thức close() có cú pháp:

```
fs.close(fd, callback)
```

Chi tiết về tham số

fd - Là viết tắt của file descriptor được trả về bởi phương thức fs.open().

callback - Hàm callback nhận một tham số để xử lý trường hợp nếu có exception.

Ví dụ

Tạo **main.js** có nội dung:

```
var fs = require("fs");
var buf = new Buffer(1024);

console.log("Chuan bi mo mot File dang ton tai");
fs.open('input.txt', 'r+', function(err, fd) {
   if (err) {
       return console.error(err);
   }
   console.log("File duoc mo thanh cong!");
   console.log("Chuan bi doc du lieu tu File da mo");
   fs.read(fd, buf, 0, buf.length, 0, function(err, bytes){
      if (err){
         console.log(err);
      }

      // In so luong byte da doc.
      if(bytes > 0){
         console.log(buf.slice(0, bytes).toString());
      }

      // Dong mot File vua duoc mo.
      fs.close(fd, function(err){
         if (err){
            console.log(err);
         }
         console.log("File duoc dong thanh cong.");
      });
   });
});
```

```
ap trinh
!!!
```

de.js

ợng thúc ftruncate() có cú pháp:

ề bởi phương thức fs.open().

ıcate.

để xử lý trường hợp nếu có exception.

Ví dụ

Tạo **main.js** có nội dung sau:

```
var fs = require("fs");
var buf = new Buffer(1024);

console.log("Chuan bi mo mot File dang ton tai");
fs.open('input.txt', 'r+', function(err, fd) {
   if (err) {
       return console.error(err);
   }
   console.log("File duoc mo thanh cong!");
   console.log("Chuan bi truncate file");

   // Truncate mot File da duoc mo.
   fs.ftruncate(fd, 10, function(err){
      if (err){
         console.log(err);
      }
      console.log("File duoc truncate thanh cong.");
      console.log("Chuan bi doc du lieu tu File");
      fs.read(fd, buf, 0, buf.length, 0, function(err, bytes){
         if (err){
            console.log(err);
         }

         // In so luong byte da doc.
         if(bytes > 0){
            console.log(buf.slice(0, bytes).toString());
         }

         // Dong File vua mo.
         fs.close(fd, function(err){
            if (err){
               console.log(err);
            }
            console.log("File duoc dong thanh cong.");
         });
      });
   });
});
```

Chạy main.js để xem kết quả:

```
$ node main.js
```

Kiểm tra kết quả:

```
Chuan bi mo mot File dang ton tai
File duoc mo thanh cong!
Chuan bi truncate file
File duoc truncate thanh cong.
Chuan bi doc du lieu tu File
VietNamVo
File duoc dong thanh cong.
```

Xóa File trong Node.js

Để xóa một file trong Node.js, bạn sử dụng phương thức unlink() có cú pháp:

```
fs.unlink(path, callback)
```

Chi tiết về tham số

path - Là tên file hoặc tên đường dẫn trỏ đến file.

callback - Hàm callback nhận một tham số để xử lý trường hợp nếu có exception.

Ví dụ

Tạo **main.js** có nội dung sau:

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

console.log("Chuan bi xoa mot File dang ton tai");
fs.unlink('input.txt', function(err) {
   if (err) {
       return console.error(err);
   }
   console.log("Xoa File thanh cong!");
});
```

Chạy main.js để xem kết quả:

```
$ node main.js
```

Kiểm tra kết quả:

```
Chuan bi xoa mot File dang ton tai
Xoa File thanh cong!
```

Tạo thư mục trong Node.js

Cú pháp

Để tạo một thư mục trong Node.js, bạn sử dụng phương thức mkdir() có cú pháp:

```
fs.mkdir(path[, mode], callback)
```

Chi tiết về tham số

path - Là tên thư mục bao gồm đường dẫn trỏ tới thư mục đó.

mode - Chế độ xác định các quyền cho phép khi truy cập thư mục. Giá trị mặc định là 0777.

callback - Hàm callback nhận một tham số để xử lý trường hợp nếu có exception.

Ví dụ

Tạo **main.js** có nội dung sau:

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

console.log("Chuan bi tao thu muc /tmp/test");
fs.mkdir('/tmp/test',function(err){
   if (err) {
       return console.error(err);
   }
   console.log("Thu muc duoc tao thanh cong!");
});
```

Chạy main.js để xem kết quả:

```
$ node main.js
```

Kiểm tra kết quả:

```
Chuan bi tao thu muc /tmp/test
Thu muc duoc tao thanh cong!
```

Đọc thư mục trong Node.js

Cú pháp

Để đọc thư mục trong Node.js, bạn sử dụng phương thức readdir() có cú pháp:

Chi tiết về tham số

path - Là tên thư mục bao gồm đường dẫn trỏ tới thư mục đó.

callback - Hàm callback nhận hai tham số, dạng (err, files) trong đó files là một mảng chứa các tên file trong thư mục.

Ví dụ

Tạo **main.js** có nội dung sau:

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

console.log("Chuan bi doc thon tin tu thu muc /tmp");
fs.readdir("/tmp/",function(err, files){
   if (err) {
       return console.error(err);
   }
   files.forEach( function (file){
       console.log( file );
   });
});
```

Chạy main.js để xem kết quả:

```
$ node main.js
```

Kiểm tra kết quả:

```
Chuan bi doc thon tin tu thu muc /tmp
ccmzx99o.out
ccyCSbkF.out
employee.ser
hsperfdata_apache
test
test.txt
```

Xóa thư mục trong Node.js

Cú pháp

Để xóa một thư mục trong Node.js, bạn sử dụng phương thức rmdir() có cú pháp:

```
fs.rmdir(path, callback)
```

Chi tiết về tham số

path - Là tên thư mục bao gồm đường dẫn trỏ tới thư mục đó.

callback - Hàm callback nhận một tham số để xử lý trường hợp nếu có exception.

Ví dụ

Tạo **main.js** có nội dung sau:

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

console.log("Chuan bi xoa thu muc /tmp/test");
fs.rmdir("/tmp/test",function(err){
   if (err) {
       return console.error(err);
   }
   console.log("Chuan bi doc thon tin tu thu muc /tmp");
   fs.readdir("/tmp/",function(err, files){
      if (err) {
          return console.error(err);
      }
      files.forEach( function (file){
          console.log( file );
      });
   });
});
```

Chạy main.js để xem kết quả:

```
$ node main.js
```

Kiểm tra kết quả:

```
ccyCSbkF.out
employee.ser
hsperfdata_apache
test.txt
```

Loạt bài hướng dẫn **học NodeJS cơ bản và nâng cao** của chúng tôi dựa trên nguồn tài liệu của: Tutorialspoint và W3Schools

Follow fanpage của team **https://www.facebook.com/vietjackteam/ (https://www.facebook.com/vietjackteam/)** hoặc facebook cá nhân Nguyễn Thanh Tuyền **https://www.facebook.com/tuyen.vietjack (https://www.facebook.com/tuyen.vietjack)** để tiếp tục theo dõi các loạt bài mới nhất về Java,C,C++,Javascript,HTML,Python,Database,Mobile.... mới nhất của chúng tôi.

Các bài học NodeJS phổ biến khác tại VietJack:

Giới thiệu qua về Module (http://vietjack.com/nodejs/module_trong_nodejs.jsp)

Ứng dụng Hello World (http://vietjack.com/nodejs/ung_dung_hello_world_trong_nodejs.jsp)

REPL Terminal (http://vietjack.com/nodejs/repl_terminal_trong_nodejs.jsp)

Node.js NPM (http://vietjack.com/nodejs/npm_trong_nodejs.jsp)

Khái niệm Callback (http://vietjack.com/nodejs/khai_niem_callback_trong_nodejs.jsp)

Trang trước (../nodejs/stream_trong_nodejs.jsp) Trang sau (../nodejs/doi_tuong_toan_cuc_trong_nodejs.jsp)

Bài viết liên quan

Các loạt bài khác:

160 bài học ngữ pháp tiếng Anh hay nhất (http://vietjack.com/ngu-phap-tieng-anh/index.jsp)

155 bài học Java tiếng Việt hay nhất (http://vietjack.com/java/index.jsp)

100 bài học Android tiếng Việt hay nhất (http://vietjack.com/android/index.jsp)

247 bài học CSS tiếng Việt hay nhất (http://vietjack.com/css/index.jsp)

197 thẻ HTML cơ bản (http://vietjack.com/html/index.jsp)

297 bài học PHP (http://vietjack.com/php/index.jsp)

85 bài học C# hay nhất (http://vietjack.com/csharp/index.jsp)

101 bài học C++ hay nhất (http://vietjack.com/cplusplus/index.jsp)

97 bài tập C++ có giải hay nhất (http://vietjack.com/bai_tap_cplusplus_co_giai/index.jsp)

208 bài học Javascript có giải hay nhất (http://vietjack.com/javascript/index.jsp)

Và còn rất nhiều loạt bài học khác tại trang web của chúng tôi....

Học tiếng Anh tại vietjack.com:

- 160 bài học ngữ pháp tiếng Anh hay nhất (http://vietjack.com/ngu-phap-tieng-anh/index.jsp)
- 160 bài tập ngữ pháp tiếng Anh hay nhất (http://vietjack.com/bai-tap-ngu-phap-tieng-anh/index.jsp)
- 72 bài ngữ pháp thực hành (http://vietjack.com/ngu-phap-tieng-anh-co-ban/index.jsp)
- 50 tình huống tiếng Anh thông dụng (http://vietjack.com/tinh-huong-tieng-anh-thong-dung/index.jsp)
- 120 bí kíp luyện phần V TOEIC (http://vietjack.com/part-5-toeic/index.jsp)

Liên hệ với chúng tôi

66 Kim Hoa, Đống Đa, Hà Nội

Phone: 01689933602

Email: **vietjackteam@gmail.com (mailto:vietjackteam@gmail.com)**