



# **LẬP TRÌNH SERVER CHO ANDROID**

## **BÀI 3: XÂY DỰNG SERVER VỚI NODE JS**

- ① Node JS server
- ① Thao tác tập tin trong Node js



## Phần I: Node JS server

-  Web server

-  Node js server

## Phần II: Thao tác tập tin trong Node js

-  Hệ thống tập tin

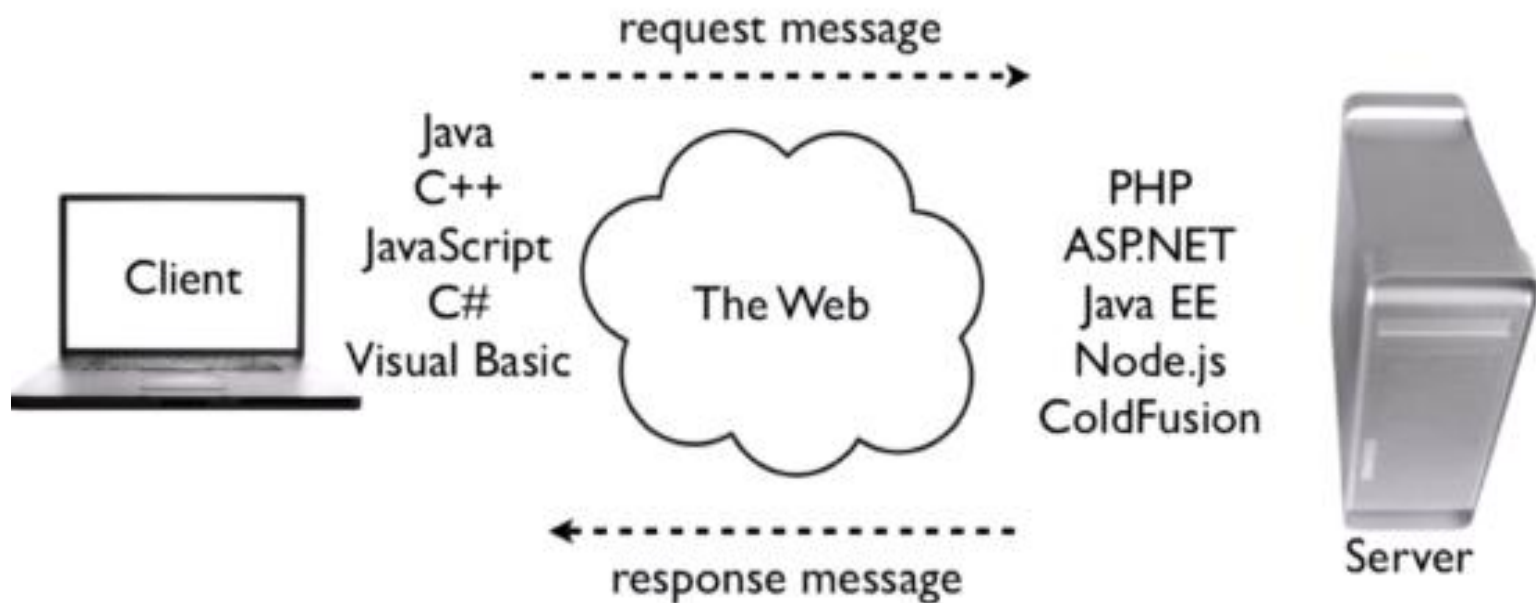
-  Thao tác tập tin



- ❑ Web server là một máy tính lưu trữ các file thành phần của một website (ví dụ: các tài liệu HTML, các file ảnh, CSS và các file JavaScript) và có thể phân phát chúng tới thiết bị của người dùng cuối (end-user). Nó kết nối tới mạng Internet và có thể truy cập tới thông qua một tên miền.
- ❑ Một số web server thông dụng: IIS, Apache, Tomcat...

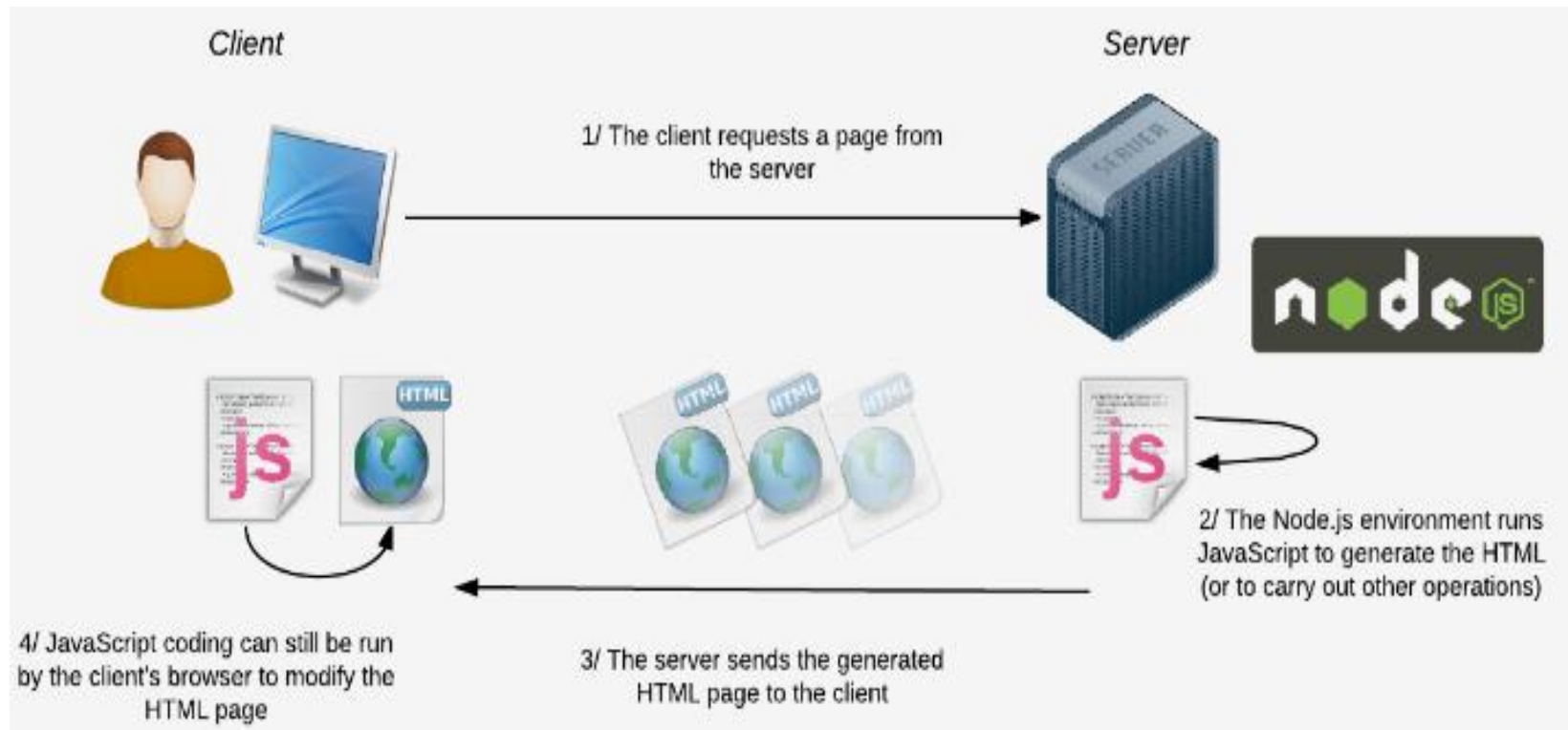
## ❑ Mô hình client – server

### Client and Server Languages



# TỔNG QUAN NODE.JS WEB SERVER

- ❑ Node.js cung cấp các khả năng để tạo ra máy chủ web của riêng bạn, nó sẽ xử lý các yêu cầu HTTP một cách không đồng bộ



- ❑ Bước 1: thêm module 'http' vào tập tin js

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

- ❑ Bước 2: sử dụng phương thức 'createServer' để tạo server lắng nghe tại port 9000

```
// include http module in the file
var http = require('http');

// create a server
http.createServer(function (req, res) {
    // code to feed or request and prepare response
}).listen(9000); //the server object listens on port 9000
```

## ❑ Bước 3: chuẩn bị nội dung hồi đáp từ server

httpWebServer.js

```
// include http module in the file
var http = require('http');

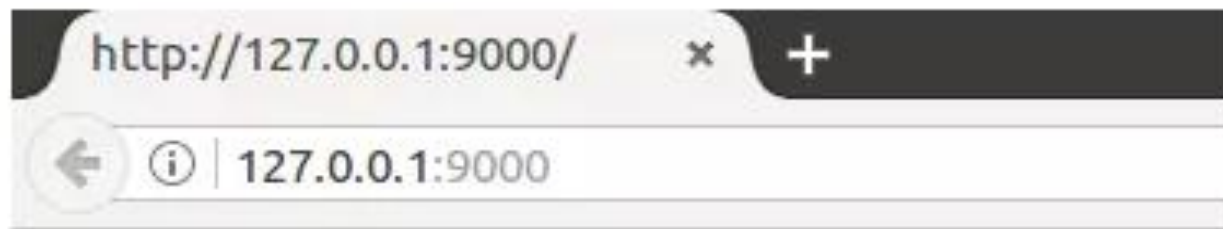
// create a server
http.createServer(function (req, res) {
  // http header
  // 200 - is the OK message
  // to respond with html content, 'Content-Type' should be 'text/html'
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write('Node.js says hello!'); //write a response to the client
  res.end(); //end the response
}).listen(9000); //the server object listens on port 9000
```



- ❑ Bước 4: khởi chạy web server tại port xác định

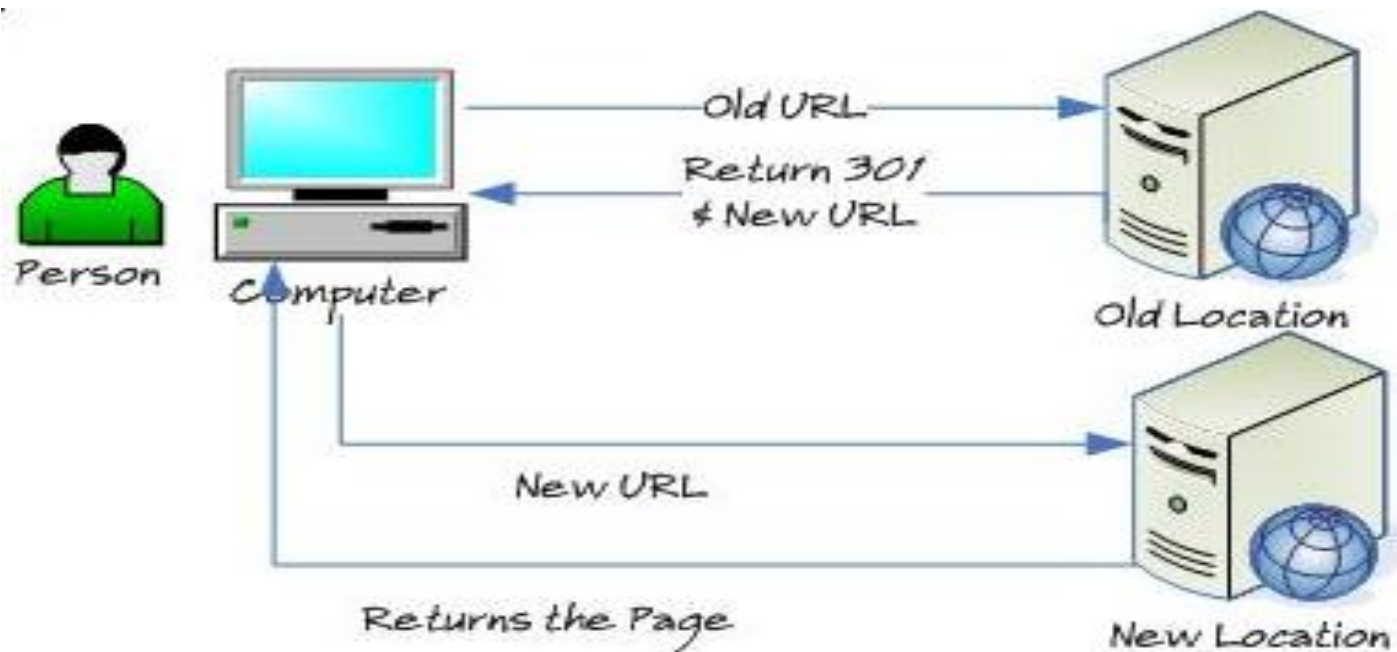
```
$ node httpWebServer.js
```

- ❑ Mở trình duyệt, gửi yêu cầu từ client tới server, nhận kết quả trả về từ server



Node.js says hello!

- ❑ Định hướng người dùng đến địa chỉ web xác định trước.
- ❑ Xác định các địa chỉ web cố định được thực thi khi có sự cố xảy ra



- ❑ Ví dụ người dùng truy cập 'page-c.html' sẽ được định hướng đến 'page-b.html'

node-js-http-redirect.js

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

// create a http server
http.createServer(function (req, res) {

  if (req.url == '/page-c.html') {
    // redirect to page-b.html with 301 (Moved Permanently) HTTP code in the response
    res.writeHead(301, { "Location": "http://" + req.headers['host'] + '/page-b.html' });
    return res.end();
  } else {
    // for other URLs, try responding with the page
    console.log(req.url)
    // read requested file
    fs.readFile(req.url.substring(1),
      function(err, data) {
        if (err) throw err;
        res.writeHead(200);
        res.write(data.toString('utf8'));
        return res.end();
      });
  }
}).listen(8085);
```

## ❑ Khởi chạy server

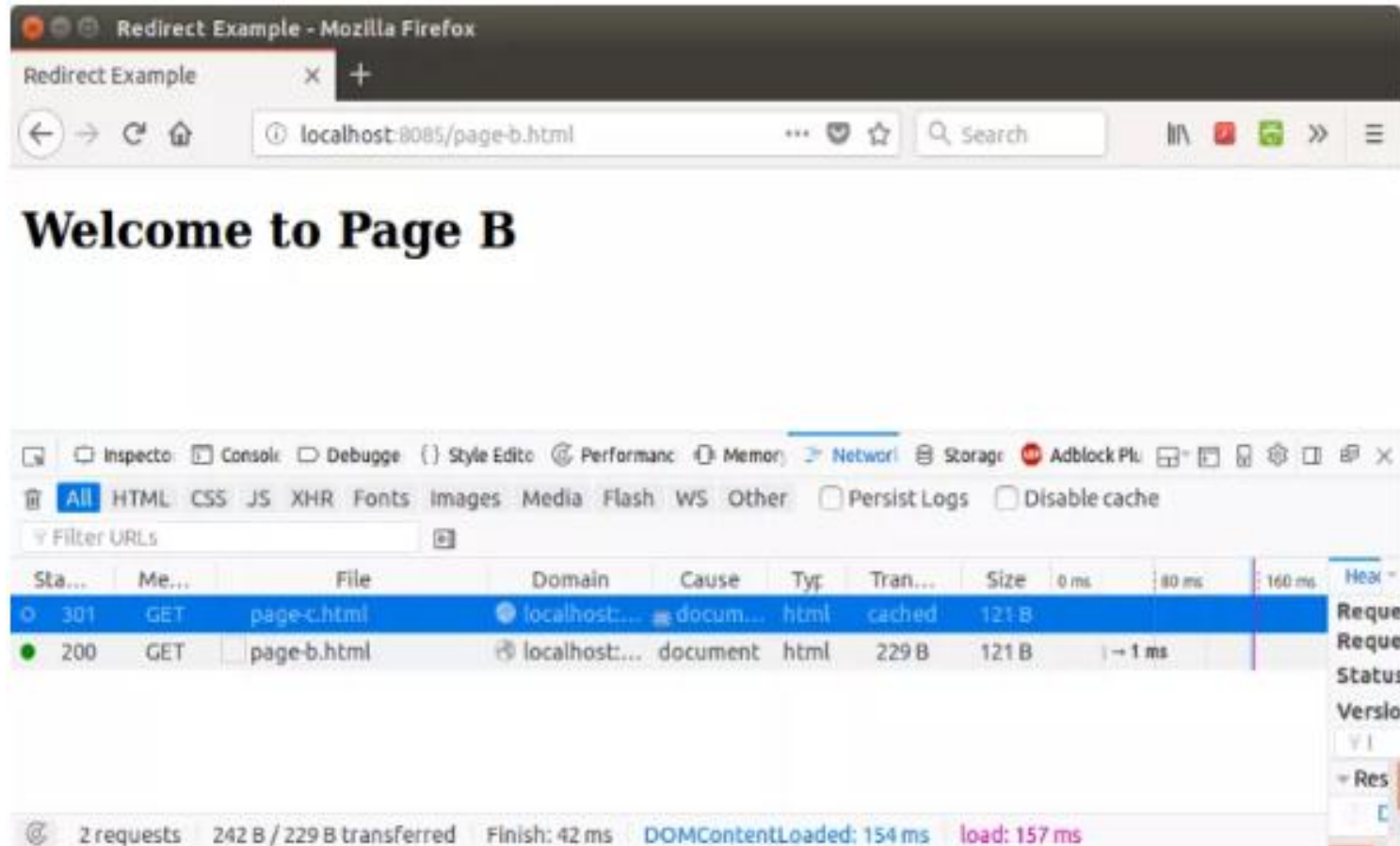
Terminal Output

```
$ node node-js-http-redirect.js
```

## ❑ Yêu cầu server thực thi trang 'page-c.html'

"http://localhost:8085/page-c.html".

❑ Kết quả sẽ được redirect về 'page-b.html'



❑ Ví dụ có sự cố website thì sẽ redirect về 'page-b.html' thay vì page 404

node-js-http-redirect-file-not-found.js

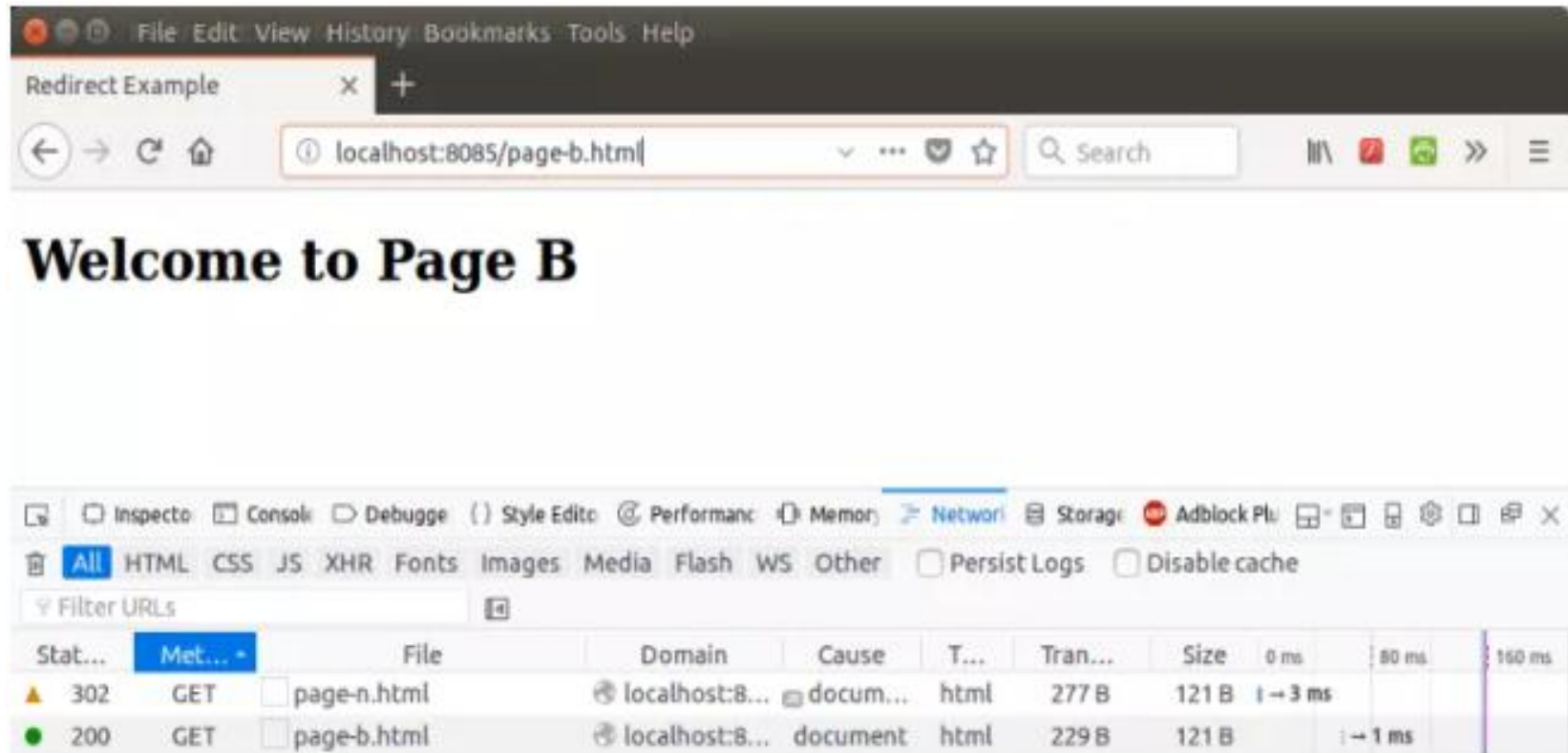
```
var http = require('http');
var fs = require('fs');

// create a http server
http.createServer(function (req, res) {
  var filePath = req.url.substring(1);
  fs.readFile(filePath,
    function(err, data) {
      // if there is an error reading the file, redirect it to page-b.html
      if (err){
        // redirect to page-b.html with 302 HTTP code in response
        res.writeHead(302, { "Location": "http://" + req.headers['host'] + '/page-b.html' });
        return res.end();
      }
      res.writeHead(200);
      res.write(data.toString('utf8'));
      return res.end();
    });
}).listen(8085);
```

❑ Khởi chạy server, kiểm tra trên trình duyệt

Terminal Output

```
$ node node-js-http-redirect-file-not-found.js
```





DEMO

- Tạo server node
- Redirect url

Lập trình Server cho Android



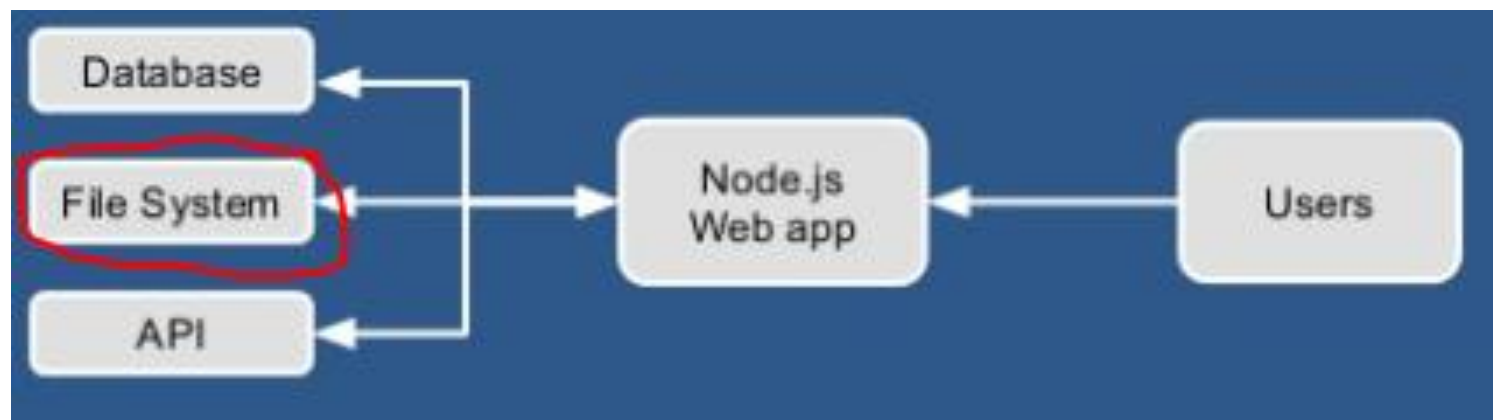


# **LẬP TRÌNH SERVER CHO ANDROID**

## **BÀI 2: XÂY DỰNG SERVER VỚI NODE JS (P2)**

- ❑ Node.js đóng vai trò là 1 File Server cho phép tương tác với các tập tin trên máy tính.
- ❑ Để nhúng module File System dùng phương thức `require()` nhúng module "fs"

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



## ❑ Read Files

- ❖ Thêm module File System vào tập tin node js

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

- ❖ Đọc file bằng phương thức 'readFile()'

```
fs.readFile('<fileName>', <callbackFunction>)
```

- filename: đường dẫn đầy đủ đến tập tin
- Options: tham số xác định encoding, mặc định utf8
- Callback: hàm được gọi khi hoàn thành quá trình đọc tập tin

## ❑ Read Files

❖ Ví dụ đọc tập tin 'demofile1.html'

```
<html>  
<body>  
<h1>My Header</h1>  
<p>My paragraph.</p>  
</body>  
</html>
```

## ❑ Read Files

❖ Tạo tập tin 'demo\_readfile.js' xử lý việc đọc file

```
var http = require('http');
var fs = require('fs');
http.createServer(function (req, res) {
    //Open a file on the server and return it's content:
    fs.readFile('demofile1.html', function(err, data) {
        res.writeHead(200, {'Content-Type': 'text/html'});
        res.write(data);
        return res.end();
    });
}).listen(8080);
```

## ❑ Read Files

### ❖ Khởi chạy server

```
C:\Users\Your Name>node demo_readfile.js
```

### ❖ Kiểm tra kết quả trên trình duyệt (<http://localhost:8080>)

`http://localhost:8080`

**My Header**

My paragraph.

## ❑ Writing File

### ❖ Sử dụng phương thức: fs.writeFile()

```
fs.writeFile(filename, data[, options], callback)
```

- filename: Đường dẫn và tên tập tin
- Data: Dữ liệu được ghi vào tập tin
- options: tham số quy định encoding và cờ đọc hoặc ghi
- callback: phương thức callback

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

fs.writeFile('test.txt', 'Hello World!', function (err) {
    if (err)
        console.log(err);
    else
        console.log('Write operation complete.');
```

```
});
```

## □ Append File

❖ Sử dụng phương thức: `fs.appendFile()`

**`fs.appendFile(path, data[, options], callback)`**

```
nodejs-append-to-file-example.js
// Example Node.js program to append data to file
var fs = require('fs');

var data = "\nLearn Node.js with the help of well built Node.js Tutorial.";

// append data to file
fs.appendFile('sample.txt', data, 'utf8',
  // callback function
  function(err) {
    if (err) throw err;
    // if no error
    console.log("Data is appended to file successfully.");
  });
```



## ❑ Append File

❖ Sử dụng phương thức: `fs.appendFileSync()`

**`fs.appendFileSync(path, data[, options])`**

```
nodejs-append-to-file-example-2.js
```

```
// Example Node.js program to append data to file
```

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

```
var data = "\nLearn Node.js with the help of well built Node.js Tutorial.";
```

```
// append data to file
```

```
fs.appendFileSync('sample.txt',data, 'utf8');
```

```
console.log("Data is appended to file successfully.")
```

## ❑ Open File

### ❖ Sử dụng phương thức: `fs.open()`

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

- Flags: xác định thao tác cần thực hiện sau khi mở tập tin
- Mode: kiểu read, write hoặc readwrite. Mặc định là 0666 readwrite

Flag	Description
r	Open file for reading. An exception occurs if the file does not exist.
r+	Open file for reading and writing. An exception occurs if the file does not exist.
rs	Open file for reading in synchronous mode.
rs+	Open file for reading and writing, telling the OS to open it synchronously. See notes for 'rs' about using this with caution.
w	Open file for writing. The file is created (if it does not exist) or truncated (if it exists).

## □ Open File

Example: File open and read

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

fs.open('TestFile.txt', 'r', function (err, fd) {

    if (err) {
        return console.error(err);
    }

    var buffr = new Buffer(1024);

    fs.read(fd, buffr, 0, buffr.length, 0, function (err, bytes) {

        // Print only read bytes to avoid junk.
        if (bytes > 0) {
            console.log(buffr.slice(0, bytes).toString());
        }

        // Close the opened file.
        fs.close(fd, function (err) {
            if (err) throw err;
        });
    });
});
```

## ❑ Delete File

- ❖ Sử dụng phương thức: `fs.unlink(path, callback);`

```
deleteFile.js
// include node fs module
var fs = require('fs');

// delete file named 'sample.txt'
fs.unlink('sample11.txt', function (err) {
  if (err) throw err;
  // if no error, file has been deleted successfully
  console.log('File deleted!');
});
```

- ❖ Sử dụng phương thức: `fs.unlinkSync(filePath)`

```
deleteFile.js
// include node fs module
var fs = require('fs');

// delete file named 'sample.txt'
fs.unlink('sample.txt', function (err) {
  if (err) throw err;
  // if no error, file has been deleted successfully
  console.log('File deleted!');
});
```

## ❑ Rename File

```
fs.rename(new_file_path, old_file_path, callback_function)
```

```
nodejs-rename-file.js
```

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

fs.rename('sample.txt', 'sample_old.txt', function (err) {
  if (err) throw err;
  console.log('File Renamed.');
```


```
});
```

```
fs.renameSync(new_file_path, old_file_path)
```

```
nodejs-rename-file.js
```


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

fs.renameSync('sample.txt', 'sample_old.txt');
console.log('File Renamed.');
```



**DEMO**

- Các thao tác với tập tin



Lập trình Server cho Android

# Tổng kết bài học

## Phần I: Node JS server

 Web server

 Node js server

## Phần II: Thao tác tập tin trong Node js

 Hệ thống tập tin

 Thao tác tập tin





**KẾT THÚC**