Bài 5
Template và Static file

Module: Web backend development with NodeJS

Mục tiêu

- Trình bày được khái niệm và ý nghĩa template trong ứng dụng web
- Trình bày được cơ chế template trong ứng dụng
- Triển khai được cơ chế template trong ứng dụng
- Trình bày được khái niệm static file trong ứng dụng web
- Triển khai được cơ chế phục vụ static file trong ứng dụng web

Mục tiêu

- Trình bày được các bước upload file lên server
- Trình bày được các bước xử lý dữ liệu người dùng nhập từ form trên server
- Triển khai được chức năng nhận dữ liệu đầu vào của người dùng nhập vào form
- Triển khai được server nhận file upload từ người dùng

Template

Thảo luận

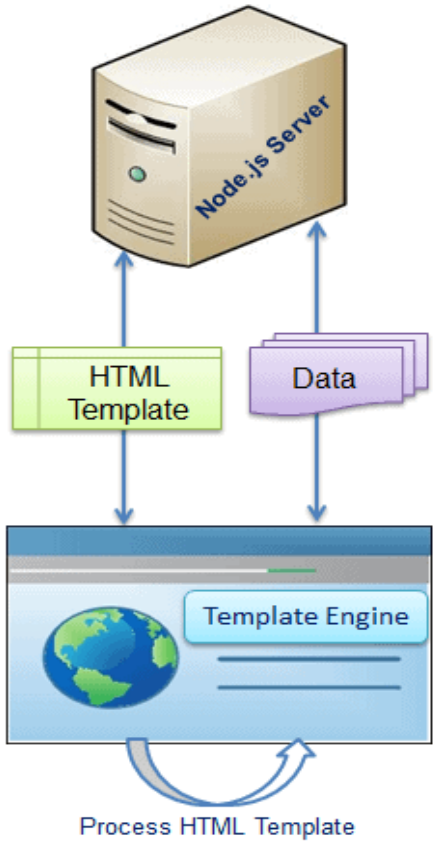
Template

- Template engines là công cụ giúp chúng ta tạo ra các HTML template bằng những đoạn mã được tối giản. Ngoài ra nó có thể đưa dữ liệu vào HTML template ở phía máy khách và tạo ra các đoạn mã HTML.
- Các thư viện template engines trong NodeJS:
  - Jade
  - Vash
  - EJS
  - Mustache
  - Dust.js
  - v.v..

Template

Trình duyệt phía máy khách tải HTML template, dữ liệu JSON / XML và thư viện Template Engines từ máy chủ

Template Engines tạo ra HTML cuối cùng bằng cách sử dụng template và dữ liệu trong trình duyệt của máy khách.



Static file

Thảo luận

Khái niệm Static file

- Là tệp không thay đổi khi ứng dụng bạn đang chạy.
- Các tệp này giúp cải thiện ứng dụng của bạn rất nhiều, nhưng chúng không được tạo động bởi máy chủ web của bạn.
- Các tệp tĩnh phổ biến nhất của bạn sẽ là các loại sau:
  - Cascading Style Sheets - CSS
  - JavaScript
  - Images

Làm việc với static file

- Trình duyệt không dựa vào mở rộng của file để hiển thị nội dung file.
- Nó dựa vào thông tin của Content-type ở header trong response
- Ví dụ:
  - Với Content-type: text/html – trình duyệt hiển thị mã HTML
  - Với Content-type: text/css – trình duyệt hiểu mã css

Làm việc với static file – Ví dụ

- Tạo file index.html

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

Làm việc với static file – Ví dụ

- Tạo file server.js: Sử dung module fs trong nodejs để làm việc với file

```
const http = require('http’);
const fs = require('fs');
http.createServer(function (req, res) {
  fs.readFile(index.html', function(err, data) {
    res.writeHead(200, {'ContentType': 'text/html'});
    res.write(data);
    return res.end();
  });
}).listen(8080);
```

Làm việc với static file – Ví dụ

- Chạy server: node server.js
- Truy cập vào http://localhost:8000 sẽ thấy nội dung file index.html hiển thị dưới dạng HTML và trình duyệt sẽ đọc nội dung đó rồi hiển thị

Upload file lên server

Thảo luận

Các bước upload file lên server

- Thiết kế form upload.
- Lấy thông tin file upload.
- Di chuyển tập tin từ thư mục tạm sang thư mục upload

Upload file trong Nodejs

- Bước 1: Tạo form upload. Tạo file server.js

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

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write('<form action="fileupload" method="post" enctype="multipart/form-data">');
  res.write('<input type="file" name="filetoupload"><br>');
  res.write('<input type="submit">');
  res.write('</form>');
  return res.end();
}).listen(8080);
```

Upload file trong Nodejs

- Bước 2: Phân tích cú pháp file tải lên

```
const http = require('http');
const formidable = require('formidable'); // module giúp phân tích cú pháp file tải lên

http.createServer(function (req, res) {
  if (req.url == '/fileupload') {
    var form = new formidable.IncomingForm();
    form.parse(req, function (err, fields, files) {
      // code xử lý upload file
      res.write('File uploaded');
      res.end();
    });
  } else {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write('<form action="fileupload" method="post" enctype="multipart/form-data">');
    res.write('<input type="file" name="filetoupload"><br>');
    res.write('<input type="submit">');
    res.write('</form>');
    return res.end();
  }
}).listen(8080);
```

Upload file trong Nodejs

- Bước 3: Lưu file. Chỉnh sửa phần code upload file như sau

```
form.parse(req, function (err, fields, files) {
  var oldpath = files.filetoupload.filepath;
  var newpath = ’upload/' + files.filetoupload.originalFilename;
  fs.rename(oldpath, newpath, function (err) {
    if (err) throw err;
    res.write('File uploaded and moved!');
    res.end();
  });
```

Tóm tắt bài học

- Template định nghĩa giao diện ứng dụng web
- Static file là tệp không thay đổi khi ứng dụng bạn đang chạy. Nó bao gồm như: css, html, image ...
- Upload file là chức năng phổ biến trong các ứng dụng web

Hướng dẫn

- Hướng dẫn làm bài thực hành và bài tập
- Chuẩn bị bài tiếp: Routing & Xử lý HTTP request