

LAB 02- SỬ DỤNG TEMPLATE EJS TRONG NODEJS

Sử dụng template EJS trong NodeJS giúp bạn code với các views một cách nhanh chóng. EJS là một template được dùng phổ biến trong nodejs.

Cài đặt EJS

```
npm install ejs
```

Khai báo sử dụng EJS

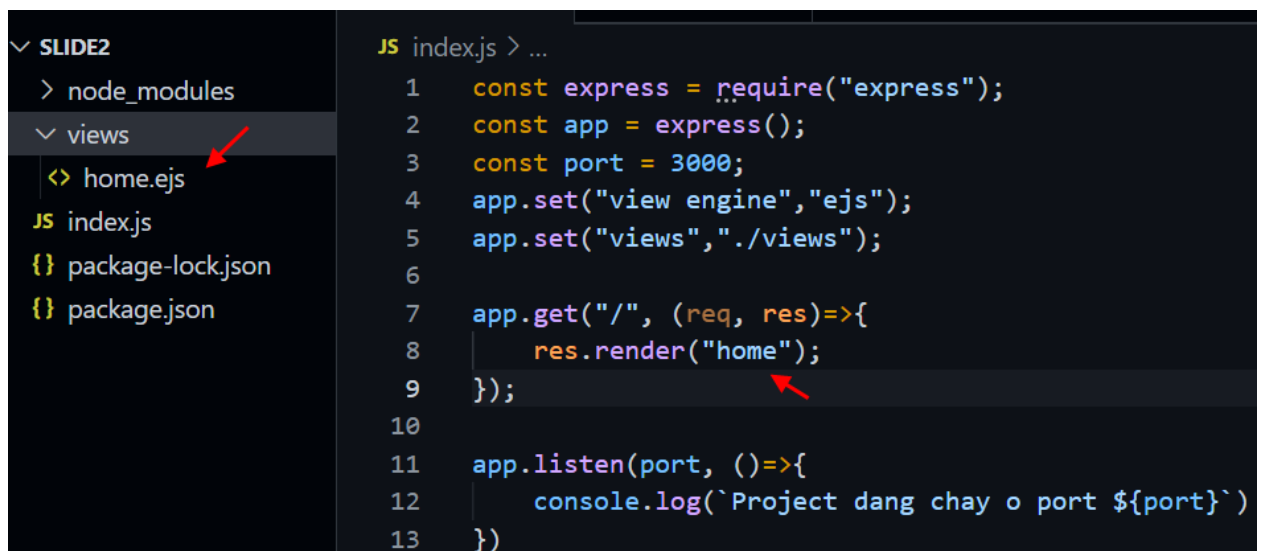
Để khai báo sử dụng ejs với NodeJS, dùng hàm set của node server với tham số **view engine** là ejs, đồng thời khai báo thông số **views** là folder chứa các file view. Xem code sau:

```
//index.js
const express = require("express");
const app = express();
const port = 3000;

app.set("view engine","ejs");
app.set("views","./views");

app.listen(port, ()=>{ console.log(`Project đang chạy ở port ${port}`)})
```

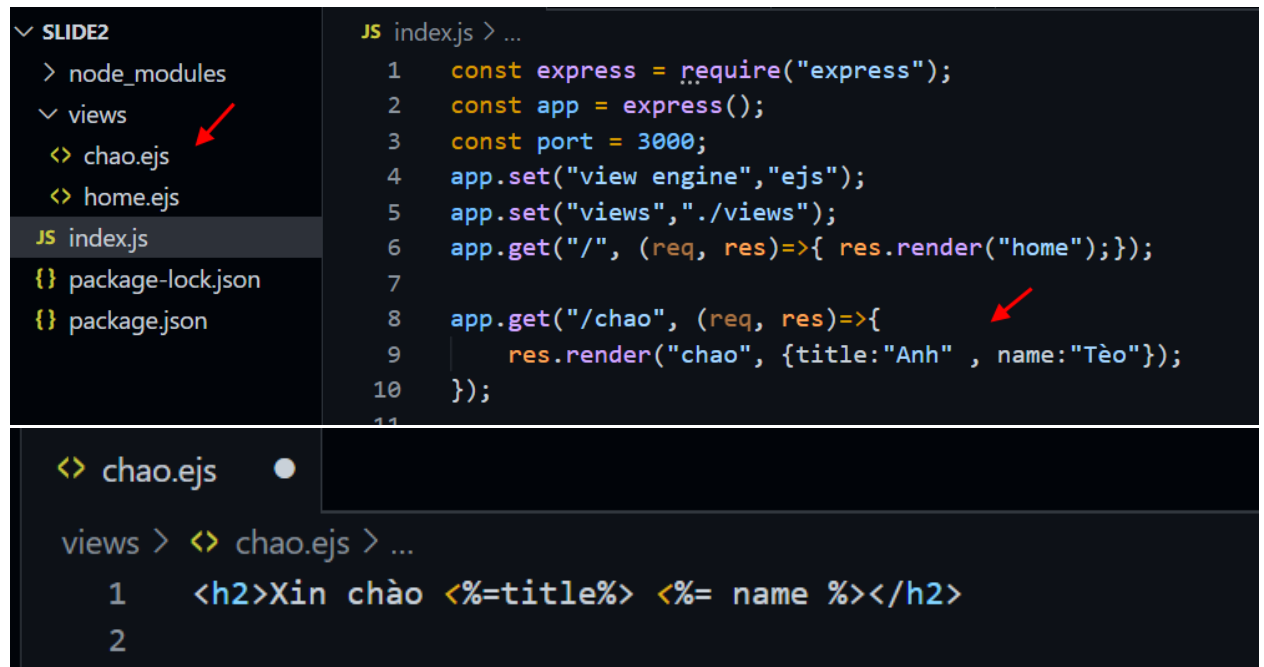
Nạp view: Để nạp view, bạn dùng hàm render trong đối tượng response. File view phải đặt trong folder views đã khai báo và có tên mở rộng là .ejs



Khi nạp view , bạn có thể truyền biến cho view để view hiển thị. Các biến truyền trong 1 đối tượng là tham số thứ hai của hàm render. Code sau truyền 2 biến **title** và **name** cho view chao

```
res.render("chao", {title:"Anh" , name:"Tèo"});
```

Trong view, hiện giá trị của biến bằng cú pháp `<%= tên biến %>`. Ví dụ:



```
JS index.js > ...
1  const express = require("express");
2  const app = express();
3  const port = 3000;
4  app.set("view engine","ejs");
5  app.set("views","./views");
6  app.get("/", (req, res)=>{ res.render("home");});
7
8  app.get("/chao", (req, res)=>{
9      res.render("chao", {title:"Anh" , name:"Tèo"});
10 });
11
```

```
<> chao.ejs
views > <> chao.ejs > ...
1  <h2>Xin chào <%=title%> <%= name %></h2>
2
```

CODE TRONG VIEW EJS

Để viết code trong view ejs, các em dùng cú pháp `<% code js %>` .

Sử dụng lệnh if trong ejs view

```
Chào bạn!  
<% if (day=='Thứ bảy' || day=='Chủ nhật') { %>  
<h3>Hôm nay là cuối tuần</h3>  
<% } else { %>  
<h3>Hôm nay là ngày <%= day%></h3>  
<% }%>
```

Vòng lặp for trong ejs view

Trong ejs view, các em có thể lặp qua mảng rất dễ dàng

```
app.get("/sp", (req, res)=>{  
  var sp= [  
    {name:'HTC M9', price:6000000},  
    {name:'Samsung S8', price:750000},  
  ]  
  res.render("sp", {sp:sp});  
});
```

Tại view

```
<% for (let p of sp) {%>  
  <p>  
    Tên SP: <b> <%=p.name%> </b> <br>  
    Giá <%=p.price %>  
  </p>  
  <hr>  
<% } %>
```

Sử dụng CSS trong view ejs

Để nhúng file css vào view, trước tiên (bước 1) bạn dùng lệnh use của node server với hàm static để chỉ định folder gốc chứa các tài nguyên tĩnh (css, js, images...). Code như sau là chỉ định folder **public** chứa các tài nguyên tĩnh

```
//index.js
const express = require("express");
const app = express();
const port = 3000;
app.set("view engine", "ejs");
app.set("views", "./views");
app.use(express.static('public'));
```

Sau đó (bước 2) là tạo file css trong folder **public**. Và cuối cùng (bước 3) nhúng file css vào trong ejs view

```
<link href="css/c1.css" rel="stylesheet">
<% for (let p of sp) {%>
  <p>
    Tên SP: <b> <%=p.name%> </b> <br>
    Giá <%=p.price %>
  </p>
  <hr>
<% } %>
```

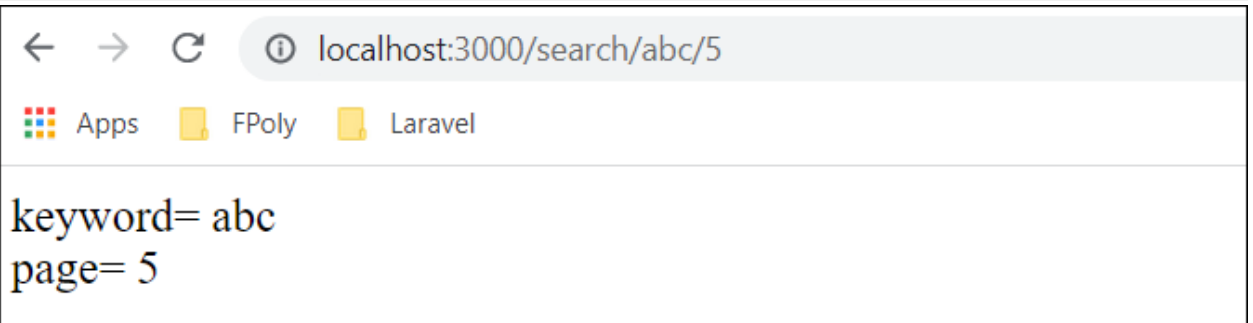
NHẬN THAM SỐ TỪ URL

Nhận tham số gửi lên từ trình duyệt là rất quan trọng. Nhờ các tham số mà trên server chúng ta mới biết phía client muốn gì. Ví dụ client muốn tìm kiếm với từ khóa gì, xem trang mấy... Các tham số thường nằm trong url và các em có thể nhận bằng 2 cách.

Cách 1: Các giá trị của tham số nằm trực tiếp trong thành phần của url.

Ví dụ <https://localhost:3000/search/abc/3>. Lúc này bạn dùng hàm **params** trong đối tượng request để lấy các giá trị để dùng.

```
app.get('/search/:keyword/:page', (req, res) => {  
    let str = `keyword= ${req.params.keyword}<br>`;   
    str+= `page= ${req.params.page}` ;  
    res.send(str);  
})
```



Cách 2: là các giá trị truyền đến như là query string.

Ví dụ <http://localhost:3000/cat?idcat=10&page=3>. Lúc này các em dùng hàm **query** của đối tượng request để nhận tham số.

```
app.get('/cat', (req, res) => {  
    str=`idcat= ${req.query.idcat} <br>`;   
    str+= `page= ${req.query.page}` ;  
    res.send(str);  
});
```



SỬ DỤNG FORM TRONG NODEJS

1. NodeJS chạy ở phía server, do đó sẽ có lúc các em cần code để nhận data gửi lên từ các form. Muốn vậy thì dùng module body-parser và chỉ định encode như sau:

```
//index.js
const bodyParser = require("body-parser");
app.use(bodyParser.urlencoded({extended:true}));
```

2. Sau khi đã nạp body-parser như trên, tiếp theo bạn tạo 1 file view chứa form gì đó. Tag form phải có thuộc tính action trỏ lên 1 route sẽ nhận dữ liệu khi user submit. Code tham khảo như sau:

```
<!-- addEmail.ejs -->
<form action="/addEmail" method="post">
  <input name="email">
  <button type="submit">Add Email</button>
</form>
```

3. Việc tiếp theo cần tạo route để nạp file view mới tạo

```
app.get('/addEmail', (req, res)=>{
  res.render("addEmail.ejs");
})
```

4. Cuối cùng tạo route để nhận dữ liệu. Code trong đây bạn dùng **req.body** để lấy dữ liệu trong form gửi lên, từ đó xử lý thế nào là tùy ý.

```
app.post('/addEmail', (req, res)=>{
  let email = req.body.email;
  res.send(email);
})
```

Upload hình trong form

Để có thể upload hình từ form, các em cần dùng external module. Có nhiều module giúp các em việc này. Sau đây là hướng dẫn cài và sử dụng module formidable.

- Cài đặt module **formidable**

```
npm install formidable
```

- Nhúng module vào trang, nên nhúng luôn module file system (fs) để tiện các thao tác trên file đã upload

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

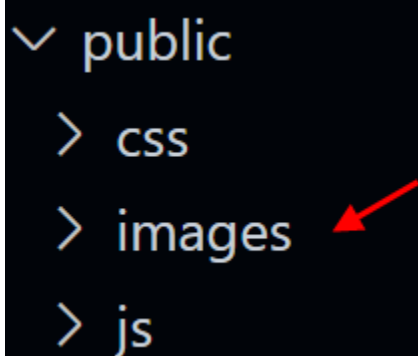
- Tạo 1 view chứa form. Tag form phải có thuộc tính **enctype="multipart/form-data"**, trong form có tag `<input type="file">` và form phải có thuộc tính action trỏ lên 1 route sẽ nhận dữ liệu

```
<!-- addproduct.ejs -->  
<form action="/addnewprod" method="post" class="form"  
  enctype="multipart/form-data" >  
  <p>Tên SP <input type="text" name="productName"> </p>  
  <p>HìnhSP <input type="file" name="productImage"> </p>  
  <p> <input type="submit" value="Thêm sản phẩm"> </p>  
</form>
```

- Tiếp theo cần tạo route để nạp view chứa form vừa tạo ở trên:

```
app.get('/addprod', (req, res)=>{  
  res.render("addproduct.ejs");  
})
```

- Việc tiếp theo là tạo 1 folder để chứa hình sẽ upload



```
✓ public  
  > css  
  > images  
  > js
```

- Cuối cùng là tạo route để nhận dữ liệu từ form , lưu hình

```
app.post("/addnewprod",(req, res) =>{
  var form = new formidable.IncomingForm();
  form.parse(req, function (err, fields, files) {
    let name = fields.productName;
    let tmpPath = files.productImage.filepath;
    let tenFile = files.productImage.originalFilename;
    let destPath = __dirname + '\\public\\images\\' +
tenFile;
    fs.rename(tmpPath, destPath, function (err) {
//copyFile
      if (err) throw err;
      res.end('File đã upload');
    });
    //res.end(JSON.stringify({ fields, files }, null, 2));
    res.send("Name=" + name);
  });
});
```

Biến **fields** trong hàm **parse** chứa các form field bình thường trong form, còn biến **files** trong hàm parse thì chứa các file được upload.

Xem thêm: https://www.w3schools.com/nodejs/nodejs_uploadfiles.asp

XOÁ ẢNH

Để xóa 1 file ảnh, các em sử dụng module fs : **var fs=require('fs');** và dùng hàm unlink để thực hiện xóa

```
app.get("/delete",(req,res)=>{
  pathfile="/images/a.png";
  fs.unlink(pathfile, function (err) {
    if (err) throw err;
    console.log('File deleted!');
    res.send("File đã xóa");
  });
});
```

Đọc thêm tài liệu ej: <https://www.npmjs.com/package/ejs>