**Comment utiliser EJS pour modéliser votre application Node**

### [Introduction](https://www.digitalocean.com/community/tutorials/how-to-use-ejs-to-template-your-node-application-fr#introduction)

Lors de la création rapide d’applications de nœuds en temps réel, il est parfois nécessaire de disposer d’un moyen simple et rapide de modéliser notre application.

[Jade](http://jade-lang.com/) est le moteur de visualisation d’Express par défaut, mais la syntaxe de Jade peut être trop complexe pour de nombreux cas d’utilisation. [EJS](https://ejs.co/) est une alternative qui fait bien ce travail et qui est très facile à mettre en place. Voyons comment nous pouvons créer une application simple et utiliser EJS pour **inclure des parties répétables de notre site (partiels)** et **transmettre des données à nos vues**.

## [Mettre en place l’application de démonstration](https://www.digitalocean.com/community/tutorials/how-to-use-ejs-to-template-your-node-application-fr#mettre-en-place-l-application-de-demonstration)

Nous allons faire deux pages pour notre application : une page avec **une page pleine** et l’autre avec une **barre latérale**.

**Obtenez le code :** [Vous pouvez trouver un git repo du code complet de la démo sur GitHub ici](https://github.com/do-community/ejs-demo)

### [Structure de fichier](https://www.digitalocean.com/community/tutorials/how-to-use-ejs-to-template-your-node-application-fr#structure-de-fichier)

Voici les dossiers dont nous aurons besoin pour notre application. Nous allons faire notre modèle dans le dossier « views » et le reste est assez standard dans les pratiques de nœuds.

- views

----- partials

---------- footer.ejs

---------- head.ejs

---------- header.ejs

----- pages

---------- index.ejs

---------- about.ejs

- package.json

- server.js

package.json contiendra nos informations sur les applications de nœuds et les dépendances dont nous avons besoin ([express](http://expressjs.com/) et [EJS](https://ejs.co/)). server.js contiendra la configuration de notre serveur Express.  Nous allons définir nos itinéraires vers nos pages ici.

### [Configuration des nœuds](https://www.digitalocean.com/community/tutorials/how-to-use-ejs-to-template-your-node-application-fr#configuration-des-n-uds)

Allons dans notre package.json et nous y installerons notre projet.

package.json

{

 "name": "node-ejs",

 "main": "server.js",

 "dependencies": {

 "ejs": "^3.1.5",

 "express": "^4.17.1"

 }

}

Tout ce dont nous aurons besoin, c’est d’Express et EJS. Il nous faut maintenant installer les dépendances que nous venons de définir. Allez-y et exécutez :

1. npm install

Une fois toutes nos dépendances installées, configurons notre application pour utiliser EJS et définissons nos itinéraires pour les deux pages dont nous avons besoin : la page index (pleine largeur) et la page about (barre latérale).  Nous ferons tout cela à l’intérieur de notre fichier server.js.

server.js

// load the things we need

var express = require('express');

var app = express();

// set the view engine to ejs

app.set('view engine', 'ejs');

// use res.render to load up an ejs view file

// index page

app.get('/', function(req, res) {

 res.render('pages/index');

});

// about page

app.get('/about', function(req, res) {

 res.render('pages/about');

});

app.listen(8080);

console.log('8080 is the magic port');

Ici, nous définissons notre application et nous la définissons pour qu’elle s’affiche sur le port 8080. Nous devons également définir EJS comme le moteur de visualisation de notre application Express en utilisant app.set('view engineer', 'ejs') ;. Remarquez comment nous envoyons une vue à l’utilisateur en utilisant res.render(). Il est important de noter que **res.render() va chercher dans un dossier de vues** pour le visualiser. Il nous suffit donc de définir pages/index puisque le chemin complet est views/pages/index.

### [Démarrez notre serveur](https://www.digitalocean.com/community/tutorials/how-to-use-ejs-to-template-your-node-application-fr#demarrez-notre-serveur)

Allez-y et démarrez le serveur en utilisant :

1. node server.js

Nous pouvons maintenant voir notre application dans le navigateur à l’adresse http://localhost:8080 et http://localhost:8080/about. Notre application est mise en place et nous devons définir nos fichiers de consultation et voir comment EJS fonctionne avec ceux-là.

## [Créez les fichiers partiels EJS](https://www.digitalocean.com/community/tutorials/how-to-use-ejs-to-template-your-node-application-fr#creez-les-fichiers-partiels-ejs)

Comme beaucoup d’applications que nous construisons, il y aura beaucoup de code qui sera réutilisé. Nous appellerons ces **partiels** et définirons trois fichiers que nous utiliserons sur l’ensemble de notre site : head.ejs, header.ejs, et footer.ejs. Faisons maintenant ces fichiers.

views/partials/head.ejs

<meta charset="UTF-8">

<title>EJS Is Fun</title>

<!-- CSS (load bootstrap from a CDN) -->

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.5.2/css/bootstrap.min.css">

<style>

 body { padding-top:50px; }

</style>

views/partials/header.ejs

<nav class="navbar navbar-expand-lg navbar-light bg-light">

 <a class="navbar-brand" href="/">EJS Is Fun</a>

 <ul class="navbar-nav mr-auto">

 <li class="nav-item">

 <a class="nav-link" href="/">Home</a>

 </li>

 <li class="nav-item">

 <a class="nav-link" href="/about">About</a>

 </li>

 </ul>

</nav>

views/partials/footer.ejs

<p class="text-center text-muted">© Copyright 2020 The Awesome People</p>

## [Ajoutez les partiels EJS à Views](https://www.digitalocean.com/community/tutorials/how-to-use-ejs-to-template-your-node-application-fr#ajoutez-les-partiels-ejs-a-views)

Nous avons maintenant défini nos partiels.  Tout ce que nous avons à faire, c’est de les inclure dans nos vues. Allons dans index.ejs et about.ejs et utilisons la syntaxe include pour ajouter les partiels.

### [Syntaxe pour l’inclusion d’un partiel EJS](https://www.digitalocean.com/community/tutorials/how-to-use-ejs-to-template-your-node-application-fr#syntaxe-pour-l-inclusion-d-un-partiel-ejs)

Utilisez <%- include('RELATIVE/PATH/TO/FILE') %> pour intégrer une partie de EJS dans un autre fichier.

* Le trait d’union <%- au lieu de <% pour dire à EJS de rendre le **HTML brut.**
* Le chemin vers le partiel est **relatif** au fichier actuel.

views/pages/index.ejs

<!DOCTYPE html>

<html lang="en">

<head>

 <%- include('../partials/head'); %>

</head>

<body class="container">

<header>

 <%- include('../partials/header'); %>

</header>

<main>

 <div class="jumbotron">

 <h1>This is great</h1>

 <p>Welcome to templating using EJS</p>

 </div>

</main>

<footer>

 <%- include('../partials/footer'); %>

</footer>

</body>

</html>

Nous pouvons maintenant voir notre vue définie dans le navigateur à l’adresse <http://localhost:8080>.



Pour la page about, nous ajoutons également une barre latérale bootstrap pour montrer comment les partiels peuvent être structurés pour être réutilisés dans différents modèles et pages.

views/pages/about.ejs

<!DOCTYPE html>

<html lang="en">

<head>

 <%- include('../partials/head'); %>

</head>

<body class="container">

<header>

 <%- include('../partials/header'); %>

</header>

<main>

<div class="row">

 <div class="col-sm-8">

 <div class="jumbotron">

 <h1>This is great</h1>

 <p>Welcome to templating using EJS</p>

 </div>

 </div>

 <div class="col-sm-4">

 <div class="well">

 <h3>Look I'm A Sidebar!</h3>

 </div>

 </div>

</div>

</main>

<footer>

 <%- include('../partials/footer'); %>

</footer>

</body>

</html>

Si nous visitons http://localhost:8080/about nous pouvons consulter notre page about avec une barre latérale !



Nous pouvons maintenant commencer à utiliser EJS pour transmettre les données de notre application de noeuds à nos vues.

## [Transmettre les données aux vues et aux partiels](https://www.digitalocean.com/community/tutorials/how-to-use-ejs-to-template-your-node-application-fr#transmettre-les-donnees-aux-vues-et-aux-partiels)

Définissons quelques variables de base et une liste à transmettre à notre page d’accueil. Retournez dans votre fichier server.js et ajoutez ce qui suit dans votre itinéraireapp.get('/').

server.js

// index page

app.get('/', function(req, res) {

 var mascots = [

 { name: 'Sammy', organization: "DigitalOcean", birth\_year: 2012},

 { name: 'Tux', organization: "Linux", birth\_year: 1996},

 { name: 'Moby Dock', organization: "Docker", birth\_year: 2013}

 ];

 var tagline = "No programming concept is complete without a cute animal mascot.";

 res.render('pages/index', {

 mascots: mascots,

 tagline: tagline

 });

});

Nous avons créé une liste appelée mascots et une chaîne simple appelée tagline. Allons dans notre fichier index.ejs et utilisons-les.

### [Rendre une variable unique dans EJS](https://www.digitalocean.com/community/tutorials/how-to-use-ejs-to-template-your-node-application-fr#rendre-une-variable-unique-dans-ejs)

Pour faire écho à une seule variable, nous utilisons simplement <%= tagline %>. Ajoutons ceci à notre fichier index.ejs :

views/pages/index.ejs

...

<h2>Variable</h2>

<p><%= tagline %></p>

...

### [Boucler les données dans EJS](https://www.digitalocean.com/community/tutorials/how-to-use-ejs-to-template-your-node-application-fr#boucler-les-donnees-dans-ejs)

Pour boucler nos données, nous utiliserons .forEach. Ajoutons ceci à notre nouveau fichier :

views/pages/index.ejs

...

<ul>

 <% mascots.forEach(function(mascot) { %>

 <li>

 <strong><%= mascot.name %></strong>

 representing <%= mascot.organization %>, born <%= mascot.birth\_year %>

 </li>

 <% }); %>

</ul>

...

Nous pouvons maintenant voir dans notre navigateur les nouvelles informations que nous avons ajoutées !



### [Transmettre les données à un partiel dans EJS](https://www.digitalocean.com/community/tutorials/how-to-use-ejs-to-template-your-node-application-fr#transmettre-les-donnees-a-un-partiel-dans-ejs)

Le partiel EJS a accès à toutes les données identiques à celles de la vue parentale. Mais attention : Si vous faites référence à une variable dans un partiel, elle doit être définie dans chaque vue qui utilise le partiel ou elle lancera une erreur.

Vous pouvez également définir et transmettre des variables à un partiel EJS dans la syntaxe include, comme ceci :

views/pages/about.ejs

...

<header>

 <%- include('../partials/header', {variant:'compact'}); %>

</header>

...

Mais vous devez encore une fois faire attention à ne pas supposer qu’une variable a été définie.

Si vous souhaitez référencer une variable dans un partiel qui n’est pas toujours défini, et lui donner une valeur par défaut, vous pouvez le faire de cette manière :

views/partials/header.ejs

...

<em>Variant: <%= typeof variant != 'undefined' ? variant : 'default' %></em>

...

Dans la ligne ci-dessus, le code EJS rend la valeur de variant si elle est définie, et par default si ce n’est pas le cas.

## [Conclusion](https://www.digitalocean.com/community/tutorials/how-to-use-ejs-to-template-your-node-application-fr#conclusion)

EJS nous permet de créer des applications rapides lorsque nous n’avons pas besoin de quelque chose de trop complexe. En utilisant des partiels et en ayant la possibilité de passer facilement des variables à nos vues, nous pouvons construire rapidement d’excellentes applications.

Pour plus de références sur EJS, voir le les [documents officiels ici](https://ejs.co/#docs).