COMPTE RENDU DE

Développement

WEB

TP1 : MEAN Stack

Installation et configuration de NodeJS

Réalisé par : Nejmeddine Jaafar

(Remarque : les commentaires et les "voirs" sont écrits en couleur verte)

## Introduction

Nodejs est un logiciel permettant d'exécuter du JavaScript côté serveur, contrairement à ce qu'on a l'habitude de voir avec le Javascript côté client.

L'avantage d'utiliser Nodejs est que Javascript permet l'exécution de tâches asynchrones, ce qui peut être pratique dans certaines situations. C'est de plus en plus souvent le cas avec le « nouveau » web qui arrive (html5/css3, etc.).

De plus, NodeJs permet de créer des applications « serveur » facilement grâce à des applications tierces qu'il prend en charge via un logiciel similaire à un gestionnaire de paquets.

## Pré-requis

1. Disposer des droits d'administration sur votre machine (de préférence Linux).
2. Disposer d’un éditeur de texte Atom ou sublimetext
3. Disposer d'une connexion à Internet configurée et activée.
4. Avoir un background sur les notions et les techniques de base en Javascript et HTML
5. Avoir suivi un cours\_démo\_tuto sur les services Web REST
6. Consulter la documentation **npmjs.com** : de la documentation sur NodeJS, le détail de tous ses modules.

## Installation de l’éditeur ATOM

Atom ubuntu

sudo add-apt-repository ppa:webupd8team/atom

sudo apt-get update

sudo apt-get install atom

Atom fedora

Telecharger Atom.rpm depuis le site : <https://atom.io/>

then

sudo dnf install ./atom.x86\_64.rpm

ou sudo rpm -ivh ./atom.x86\_64.rpm

enfin pour lancer atom taper

atom

pour windows (-\_-)

<https://github.com/atom/atom/releases/download/v1.12.9/AtomSetup.exe>

## Installation de NodeJS sur Linux

# Ubuntu

Pour installer nodejs sur ubuntu

curl -sL https://deb.nodesource.com/setup\_6.x | sudo -E bash - sudo apt-get install -y nodejs

# Fedora et Centos

Pour installer nodejs sur Fedora ou Centos ( Red Hat)

curl --silent --location https://rpm.nodesource.com/setup\_6.x | bash -

yum -y install nodejs

# Windows

Si vous avez Windows il suffit de télécharger le package à partir du lien suivant :

https://nodejs.org/en/download/

Vérifier que NodeJS est installé en tapant :node -v

Résultat attendu: v6.x.y

## Commencer à utiliser NodeJS

# 1ère application : Hello world

# Nous pouvons maintenant nous lancer dans le grand bain et créer notre première application Node.js, la traditionnelle « Hello World ».

Ouvrez votre éditeur favori et créez un fichier ***helloworld.js***. Nous souhaitons afficher le message «Hello World » sur la sortie standard. Voici le code permettant de le faire :

*console.log('Hello World');*

*Imprimer Hello World sur la console (et log si on est sur le navigateur et le serveur est démarré)*

Sauvegardez le fichier et exécutez-le avec Node.js :

*node helloworld.js*

*Executer le fichier helloworld*

Ceci devrait afficher le texte « Hello World » dans la console.

# Afficher hello world dans une page web

Nous voulons servir des pages Web, nous avons donc besoin d'un serveur HTTP.

Notre serveur devra répondre aux requêtes issues en affichant hello world.

Nous allons donc créer un fichier principal pour lancer notre serveur ***server.js*** contenant le code suivant :

*var http = require("http");*

*importation du module http*

*http.createServer(function (request, response) {*

*création du serveur nodejs*

*response.writeHead(200, {'Content-Type': 'text/plain'});*

*création d'un texte*

*response.end('Hello World\n');*

*le texte est Hello World*

*}).listen(8081);*

*ecouter sur el port 8081*

C'est tout ! Vous venez juste d'écrire un serveur HTTP fonctionnel. Pour le vérifier, nous allons le lancer et le tester. D'abord, exécutez votre script avec Node.js :

*node server.js*

Maintenant, ouvrez votre navigateur à l'adresse *http://localhost:8081/*. Ce qui devrait afficher une page avec le message « Hello World ».

Si nous détaillons un peu notre code :

La première ligne indique que nous avons besoin du module http inclus dans Node.js et qu'il sera disponible à travers la variable http.

Nous appelons ensuite la fonction createServer() présente dans le module http. Cette fonction retourne un objet, dont une des méthodes, *listen()*, prend une valeur numérique en paramètre correspondant au port que doit écouter le serveur(8081).

la méthode createServer() prend une fonction comme argument (callback function qui vaêtre exécutée chaque fois notre serveur reçue une requête GET/).

## Faire la somme de Deux nombres

Pour faire la somme de deux nombres :

(server.js)

*var http = require("http");*

*http.createServer(function (request, response) {*

*response.writeHead(200, {'Content-Type': 'text/plain'});*

*var a = 100;*

*var b = 200;*

*var result = a+b;*

*création et calcul des variables*

*response.end(result.toString());*

*exportation du résultat*

*}).listen(8081);*

On doit convertir result en un string pour l'inclure dans le payload du protocole http et l'envoyer à l'utilisateur

## Créer sa page HTML

## Envoyer un fichier html à l'utilisateur

On suppose qu'au lieu d'afficher 'hello world” dans le navigateur du client on va afficher une page html complète : (page d'accueil d'une bibliothèque en ligne):

home.html:

<ul>

<li>

<hr>

<h3><strong> Book Title : </strong>Crime and Punishment</h3>

<ul>

<li><strong>Author :</strong>Fyodor Dostoyevsky</li>

<li><strong>Type :</strong>russian litterature</li>

<li><strong>Pages :</strong>361</li>

<li><strong>Avalability :</strong>true</li>

<li><strong>Book id :</strong>x144YHEZ454GC4CTZ</li>

</ul>

<hr>

</li>

</ul>

code de la page html (évident)

on créer maintenant notre serveur http ;

var http = require('http');

var fs = require('fs');

importation du module file system

var server = http.createServer(function (req, res) {

displayHome(res);

appel de la fonction displayhome

}).listen(3000);

console.log("server listening on 3000");

Rien de nouveau, un simple serveur http qui va utiliser le port 3000 et le module fs (file system) pour lire des fichier à partir du disque dur

**la fonction displayHome(res)**

*function displayHome(res) {*

*fs.readFile('home.html', function (err, data) {*

*importation (lecture) du fichier home.html*

*res.writeHead(200, {*

*ecriture sur la page parente*

*'Content-Type': 'text/html',*

*un texte de type html*

*'Content-Length': data.length*

*de longueur la taille du fichier*

*});*

*res.write(data);*

*écriture*

*res.end();*

*fin de l'opération et de l'exécution du script*

*});*

*}*

La fonction displayHome va utiliser le module fs pour lire le contenu du fichier home.html

La fonction writeHead va spécifier la nature du payload du protocoloe http (text, json, file, mpeg ...)

Ensuite, on va utiliser la fonction write qui va charger le contenu du fichier dans le payload du protocole http (pour être techniquement correct)

Envoyer un formulaire

form.html

*<html>*

*<body>*

*<hr>*

*<form action="" method="post" enctype="multipart/form-data">*

*<fieldset>*

*<label for="name">Name:</label>*

*<input type="text" placeholder="Enter your full name" />*

*<br />*

*<label for="email">Email:</label>*

*<input type="email" placeholder="Enter your email address" />*

*<br />*

*<label for="password">password:</label>*

*<input type="password" placeholder="password" />*

*<br />*

*<input type="submit" value="Create Profile" />*

*</fieldset>*

*</form>*

*<hr>*

*</body>*

*</html>*

*une page html contenant des champs de saisi texte (évidente)*

de la même manière on affiche la form html à l'utilisateur

## Commencer à utiliser NPM

NPM, le package manager de Node.js, a grandement participé au succès de ce dernier. Il facilite la création, le partage et l'installation de modules. Il permet aux développeursde télécharger des modules pour les utiliser dans son propre projet comme express, crypto ...

NPM joue un rôle similaire à *apt-get* pour ubuntu et *yum* pour fedora et centos

Pour installer un module il suffit de taper :

*npm install nom\_du\_module*

**Exemple**

On veut hasher une chaîne de caractères en utilisant l'algorithme md5:

Où lieu de développer une fonction pour achever cette tâche on peut utiliser le module md5:

Installation de md5

Tout d'abord créer un dossier pour notre projet qui va contenir tous nos fichier

*mkdir project*

*cd project*

Installer md5

*npm install md5*

Créer un fichier test.js contenant le code :

*var md5 = require('md5');*

*//text à crypter*

*var clairText = "i like nodejs"*

*//invoquer la fonction md5 pour hasher le text*

*var hashedText = md5(clairText);*

*application du md5 sur le text à crypter*

*//afficher le hash*

*console.log(hashedText)*

Sauvegardez le fichier et exécutez-le avec Node.js :

*node test.js*

Remarque

Vous pouvez consulter le site [https://www.npmjs.com](https://www.npmjs.com/) pour télécharger des modules.

# **To do 1**

Utiliser le même module (md5) pour hasher un fichier (penser au module fs)

Voir fichiers commençants par todo1

**To do 2**

Chercher un module dans npmjs.com pour hasher un fichier avec sha256

**Exercice (envoyer un mail avec nodejs):**

Créer une adresse mail (gmail)

Activer

(<https://www.google.com/settings/security/lesssecureapps>) (turn on)

Utiliser nodemailer pour envoyer un mail

<https://www.npmjs.com/package/nodemailer>

Indication :

On suppose que vous avez créé l'adresse suivante : [mynode@gmail.com](mailto:mynode@gmail.com) et le mot de passe c'est : mypassword

remplacer var transporter **=** nodemailer.createTransport('smtps://user%40gmail.com:[pass@smtp.gmail.com](mailto:pass@smtp.gmail.com)');

par :

var transporter **=** nodemailer.createTransport('smtps://mynode%40gmail.com:[mypassword@smtp.gmail.com](mailto:mypassword@smtp.gmail.com)');

Remarque

Vous pouvez consulter le site [https //www.npmjs.com](https://www.npmjs.com/) pour télécharger des modules.

## voir fichier todo2 mailer

## Créer vos propres modules

Vous pouvez créer vos propres modules et les utiliser dans vos projets.

Exemple

On veut créer un module sum qui calcule la somme d'une suite arithmétique :

Créer un nouveau dossier Sum , et dans ce dossier, Créer un fichier sum.js

*touch sum.js*

# Le fichier sum.js va contenir ce code

*var sum={};*

*//la fonction calculate est un attribue de l'objet sum*

*sum.calculate = function (n) {*

*return (n\*(n+1))/2;*

*}*

*//pour qu'on puisse utiliser cette fonction on doit exporter ce module*

*module.exports = sum;*

Créer le fichier index.js : colonne vertébrale de votre projet

*var sum = require('./sum');*

*var resultat = sum.calculate(100);*

*console.log('la somme de la suite 1+2+..+100 est :'+resultat)*

Exécuter le fichier index.js

node index.js

resultat: la somme de la suite 1+2+..+100 est :5050

# Travail à faire

Développer une application avec NodeJS qui demande à l'utilisateur d'introduire son mot de passe dans un champ Text, appliquer un Hashage du mot de passe introduit et le comparer avec le hash d'un mot de passe statique.

Indications

Installer et consulter la documentation du module md5 disponible en suivant ce lien

<https://www.npmjs.com/package/md5>

Créer une page html qui va contenir une form html pour la saisie des données

<http://www.w3schools.com/html/html_forms.asp>

Envoyer la page html au client lorsqu'il se connecte à votre serveur sur le port 8080

Vous pouvez installer et utiliser le module formidable

Vous devez utiliser le module natif fs pour lire le fichier form.html et l'envoyer au client

https://www.sitepoint.com/creating-and-handling-forms-in-node-js/

Voir taf