Ecosystème javascript

Pré-installations

- Nodejs

http://www.nodejs.org

Implantation serveur de la machine v8 de google Chrome. https://code.google.com/p/v8/

```
node
\> i = 8
\> console.log("coucou", i++)
```

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

http.createServer(
  function (request, response) {
    response.writeHead(200, {'Content-Type': 'text/plain'});
    response.end('Hello World\n');
  }
).listen(8000);

console.log('Server running at http://localhost:8000/');
```

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

net.createServer(
  function (stream) {
    stream.write('hello\r\n');

    stream.on('end',
      function () {
        stream.end('goodbye\r\n');
      }
    );

    stream.pipe(stream);
  }
).listen(8000);
```

- Mongodb
  http://www.mongodb.org/
  Base de données nosql, approche documentaire.
- NPM - Node package manager
- Bower - Web package manager

```
> npm install -g bower
```

- grunt
  chaine de compilation `npm install -g grunt-cli`

Outils installés par la chaine :

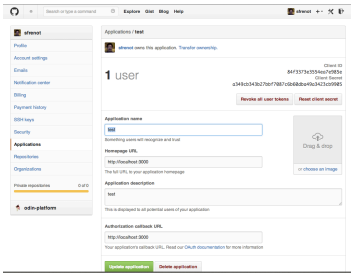
- Express : server web au dessus de nodejs
- Mongoose : description javascript des documents mongodb
- Passport : Outil de gestion des connexions sur réseaux sociaux
- AngularJS : Framework graphique client
- Twitter Bootstrap : Interface graphique CSS / HTML5
- UI Bootstrap : composants graphiques génériques

Lancement rapide MEAN

```
$ [sudo] npm install -g meanio@latest
$ mean init <myApp>
$ cd <myApp> && npm install
$ grunt
```

==> Consultation de la page

Ajout d'un accès réseau social



https://github.com/settings/applications

Tester avec github et autres accès réseaux sociaux.

Ajout d'un nouvel affichage

Injecter une variable du scope disponible

Fichier de modèle mongoose : server/models/user.js

Essayez d'afficher différents paramètres du scope

public/system/views/index.html {{global.user.username}}

Essayez d'afficher le mail de l'utilisateur

Attaquons des nouvelles données

Installation de redis

http://redis.io

Base noSql orientée haute performance en mémoire.

```
$ tar xxx
$ make
```

Récupérer le dump.rdb Lancer redis-server, attendre 20s avant validation de la base.

Tester en vérifiant une collection

```
$ redis-client
$ smembers "hal:coauthors:stephane,frenot"
```

Injectons les co-auteurs

Le but ici est d'injecter les co-auteurs d'un utilisateurs dans le système.

1. Injection des co-auteurs à la création du compte Utiliser le module redis pour s'interfacer avec la base redis. Injecter les coauteurs dans la base juste avant la sauvegarde de l'objet user.

On passe par le controlleur AngularJs. server/controllers/users.js

Testez : redisModule = require('redis')
Puis chargez une référence redis = redisModule.createClient()

```
redis.smembers('hal:coauthors:'+user.name, function (err, response) {
   response.forEach(function (reply) {
     user.coauteurs.push(reply);
   });
   user.save(function(err) {
      ...
```

--> Bug message

Ajoutez les coauteurs dans le modele.

1. Afficher les coauteurs
   Valider les coauteurs dans la vue (cf. controlleur)
   server/controllers/index.js
2. Injecter le resultat dans la vue principale `<h2> Vos CoAuteurs </h2> <ul class="nav nav-list"> <li class="navli" ng-repeat="coauteur in global.user.coauteurs" ui-animate> {{coauteur}} </li> </ul>`

La dernière ligne droite

Créer le crawler de hal J'utilise coffeescript. (http://coffeescript.org)

Récupération des centres de recherches

Insérer une clé de test dans redis en utilisant coffeescript

```
#!/opt/local/bin/coffee
"use strict"
redis = require("redis")
client = redis.createClient()
client.on 'error', (err) ->
  console.log("Error "+ err)
client.set("test", 2)
console.log("Valeur 'test' injectée'")
client.quit()
```

Testez que votre clé est bien insérée dans redis.

Tester la requête de base sur hal

`http://hal.archives-ouvertes.fr/index.php?halsid=p48p8bgnm6huuhp2jh595uad56&lance_recherche=1&action_todo=search_known_labo&rechercher=Recherche&text_to_search=`

Ecrire le code coffeescript permettant d'interroger hal.

(On peut passer par le module npm request.)

Parsez le résultat de la requête

Parsez le résultat pour extraire l'identifiant et le descripteur d'un laboratoire.

Pour parser des chaines, j'utilise deux fonctions coffeescript. test et exec.
http://coffeescriptcookbook.com/chapters/regular_expressions/searching-for-substrings

Insérez les clés dans redis

key :: hal:centre: -> description zset :: hal:centres -> id, description

Testez qu'un des centres de recherche de hal est bien inséré

Récupération des auteurs / coauteurs

Lancer et analysez le code du fichier loop.coffee

```
#!/opt/local/bin/coffee
"use strict"
redis = require("redis")
request = require("request")
client = redis.createClient()

client.on 'error', (err) ->
  console.log("Error" + err)

quit = () ->
  client.quit()

getPapers = (labId, idx) ->
  console.log("Ajout Labo #{labId} index : #{idx}")
  request 'http://hal.archives-ouvertes.fr/Public/afficheRequetePubli.php?
labos_exp='+labId+'&CB_auteur=oui&CB_identifiant=oui&langue=Francais&tri_exp=annee_publi&tri_exp2=typdoc&tri_exp3=date_pub
li&ordre_aff=TA&Fen=Rech&lang=', (error, response, body) ->
    console.log 'Recupération Lab : '+labId
    if (!error && response.statusCode == 200)
      lignes = body.split("\n")
      while lignes.length > 0
        start = lignes.shift()
        if /<dl class=\'NoticeRes\'>/.test(start)
          auteurs = (/<dt class="ChampRes">Auteurs<\/dt><dd class=\"ValeurRes Auteurs\">(.*)<\/dd>/.exec(lignes.shift())
[1]).split("; ")
          identifiant = /<dt class=\"ChampRes\">Identifiant<\/dt><dd class=\"ValeurRes Identifiant\"><a
href=\"http:\/\/hal.archives-ouvertes.fr\/index.php\?action_todo=search&amp;view_this_doc=(.*) target=\"_blank\" >(.*)
<\/a><\/dd>/.exec(lignes.shift())[2]
          for auteur in auteurs when auteurs.length < 20
            console.log("Ajout #{auteur} / #{auteurs.length}")
            client.sadd "hal:authors", auteur.trim()
            client.lpush "hal:id:"+identifiant, auteur.trim()
            client.sadd "hal:lab:"+labId, auteur.trim()
            for coauteur in auteurs
              if coauteur isnt auteur
                client.sadd "hal:coauthors:"+auteur.trim(), coauteur.trim()
      client.set "hal:doneId", idx
    else
      console.log "Error "+error
      quit()

getLabsPapers = (labList, idx) ->
  lab = labList.shift()
  if lab isnt undefined
    client.zscore ["hal:centres", lab], (err, idLab) ->
      idx++
      getPapers(idLab, idx)
      setTimeout(getLabsPapers, 10000, labList, idx)
  else
    quit()

# hal:doneId indique le rang du labo. cf numéro de ligne dans le fichier liste.centre.sort.all
client.zcard "hal:centres" , (err, taille) ->
  client.get "hal:doneId", (err, start) ->
    client.zrange ["hal:centres", start, taille], (err, response) ->
      getLabsPapers(response, start)
```

Refs : ngbook_angularjs.pdf Pro_AngularJs.pdf

http://modulecounts.com/

redmonk...

http://strongloop.com/developers/node-js-infographic/

http://delicious-insights.com/mixit-node/#/1

https://github.com/search?q=stars%3A%3E%3D0&type=Repositories&ref=searchresults

http://techcrunch.com/2012/09/12/javascript-tops-latest-programming-language-popularity-ranking-from-redmonk/

http://www.bennadel.com/blog/2439-My-Experience-With-AngularJS-The-Super-heroic-JavaScript-MVW-Framework.htm