Louis ROUSSEAU
Antoine LEFEVRE

IMT Lille Douai
École Mines-Télécom
IMT-Université de Lille

Documentation développeur

1/ Outils à installer

Installation de node.js → https://nodejs.org/en/

Installation de MongoDB → https://www.mongodb.com/download-center/community

**Installation de Studio 3T for MongoDB → https://studio3t.com/download/

Installation d'un éditeur de texte

2/ Configuration environnement de développement

L'environnement de développement est déjà configuré avec les outils de la section 1.

3/ Procédure de build

Création et remplissage de la base de données :

Depuis l'outil Studio 3T, créer une nouvelle connexion sur le port 27017 qui correspond au port par défaut de MongoDB. Une fois la connexion établi, il faudra créer une base de données et y ajouter des collections.

Les deux collections à créer sont « chapters » et « characters ».

Les documents de chapters comporteront les objets suivants :

- ObjectId : _id
- Array : characters (fera office de « clé étrangère » d'un point de vue SQL)
- String : name

Les documents de characters comporteront les objets suivants :

- ObjectId : _id
- String : chineseName
- String : chinesePhonetic
- String : frenchName

4/ Présentation de l'architecture, des concepts

- Le concept même d'utiliser du Javascript coté serveur (node.js) est de pouvoir générer des pages web rapidement (grâce à son modèle non bloquant) tout en se passant de langages serveur comme PHP, Java EE, etc.
- Node.js étant bas niveau, il n'utilise pas de serveur web HTTP comme Apache. C'est à nous de créer le serveur.
- Node.js va être lié à un système de gestion de base de données NoSQL orientée documents (MongoDB).
- L'application adopte une architecture Modèle Vue Contrôleur (MVC).
  Le modèle comportera les fichiers d'initialisation de nos schémas de données liées à la MongoDB.
  La vue comportera les fichiers HTML que le client recevra.
  Le contrôleur comportera la logique du code et les routes possibles que le client requêtera, appelant ainsi les fichiers HTML lié à cette route.
- Le concept de notre application est de mettre à disposition un dictionnaire Français/Chinois découpé en chapitre. Un chapitre comportera plusieurs caractères et dont un caractère sera visible sous 3 formes possibles (chinois, phonétique chinoise, français).
- La manière dont nous avons pensé cette application engendre donc une relation entre les chapitres et les caractères. En effet, la collection chapitre de la MongoDB comportera un tableau de caractères qui pointera directement sur les documents de la collection caractère qui elle-même aura un champ qui pointera sur un document de la collection chapitre. Ce concept permet de lier les collections entre elles puisque MongoDB étant NoSQL, il n'est pas possible d'associer des clés étrangères aux documents. Cela engendre donc une relation one-to-many qui est la base de notre modèle.

5/ Documentation du code

- Fichier principal app.js :
  Va rendre les modèles et contrôleurs accessibles grâce à des « require »
  Initialiser le Framework express pour le routage ainsi que la création du server http

- Dossier models/ Fichiers model.js (Modèle) :
  Vont permettre d'initialiser les schémas des chapitres et des mots de la MongoDB pour pouvoir créer des objets utilisables portant ce schéma dans les contrôleurs.
  Les schémas reprennent les champs des documents des collections de la MongoDB pour que les objets ainsi générés puissent modifier les collections.

- Dossier views (Vue) :
  Ce dossier comportera l'ensemble des fichiers HTML que requêtera le client. Ces fichiers font appel à du code JavaScript pour pouvoir traiter les variables passées en paramètre par le contrôleur lors de l'appel d'une route et générer ainsi des pages dynamiques.
  Egalement, du Css a été directement intégré au code HTML mais le plus gros de la mise en page a été réalisée à partir de classe Bootstrap.

- Dossier controllers (Contrôleur) :
  Initialisation des constantes utilisées en globales dans les fonctions du fichier.
  Initialisation des routes (URL) ainsi que du comportement des fonctions de ces dernières pour faire un rendu des fichiers HTML paramétrés par la fonction.
  Les fonctions async() permettent de limiter le comportement non bloquant de Node.js en forçant l'ordre de compilation des fonctions avec l'attribut await devant la fonction prioritaire.

6/ Axes d'amélioration

- La mise en forme des vues n'a pas été autant travaillée que nous le souhaitions. De ce fait, nous n'avons pas pu perfectionner le rendu des pages html. Le premier point d'amélioration pourra donc se tourner vers la mise en forme des vues du fait que le code des contrôleurs ne comporte aucune erreur et ne demande pas une révision d'urgence.

- Pour des questions de sécurité, la vérification des inputs réalisés par l'utilisateur devra être faite pour filtrer les entrées pouvant affecter la MongoDB.

- Pour l'utilisation de l'application en dehors de l'environnement de développement (local), la base de données MongoDB devra être installée sur un server. La configuration restera la même pour cette dernière.

Axes d'amélioration :

- Ajout d'une authentification (couple login/mdp) qui sera stockée dans une nouvelle collection de la MongoDB. Cette authentification pourra permettre de restreindre l'accès à la vue admin.
- Possibilité de mise en place de session d'utilisation pour garder en cookie des résultats d'actions que l'utilisateur aura effectué.
- Cette session pourra également permettre à l'utilisateur d'enregistrer des mots dans un tableau. Ce tableau sera différent pour chaque utilisateur du fait de la mise en place des sessions.