Sinthujan SIVANATHAN 16/09/2016
Test de développement Vert.X

Énoncé

Implémenter et exposer une API REST avec Vert.x 3 (en JavaScript ou en Java) pour afficher et consulter les livres et les journaux d'une bibliothèque

Aide

http://vertx.io/docs/vertx-web/js/
http://vertx.io/docs/vertx-web/java/

À modéliser

- une bibliothèque qui contient des livres et des journaux
- le livre et le journal sont composés de pages
- la page d'un livre peut faire partie d'un chapitre
- la page d'un journal peut contenir un ou plusieurs articles

note : Il n'est pas nécessaire de gérer de stockage de données. Vous pouvez simuler ("mocker") les données directement dans le code.

Exposer le contrat fonctionnel suivant via une API REST:

OK - voir toute la bibliothèque
OK - voir un livre donné
OK - voir un journal donné
OK - voir directement une page d'un livre
OK - voir directement une page d'un journal
OK - voir tous les articles d'une page de journal
OK - [OPTION] proposer une gestion simple de marque-page (enregistrement et récupération)

note : il n'est pas nécessaire d'implémenter une IHM. Les donnée peuvent être présentées simplement en JSON ou en XML

Livrables

- une implémentation de l'API REST avec Vert.x en utilisant JavaScript ou Java selon votre préférence
- un texte d'explication de vos choix de conception et d'implémentation

note : Le code doit être livré sous forme d'une archive zip ou via un dépôt git (github, bitbucket, serveur personnel …)

Pour les différentes entités de la bibliothèque j'ai établi le schéma suivent :

C : contient

Bibliothèque C Livre et Journaux

Livre C Chapitre
Chapitre C PageLivre
PageLivre C Text

Journaux C PageJournal
PageJournal C Text et/ou Article

Dans le cas des livres j'ai choisi de mettre le chapitre avant les pages pour ne pas avoir une redondance de la même information dans les pages.

Pour la Base de données j'ai utilisé un Sandbox sur MongoLab,
Avec mongoDB j'avais le choix de faire une base plus déstructuré avec des collections distinct pour chaque entité et des identifiants qui les relient entre eux.

Cela aurait été pertinent s'il y avait la recherche d'un mot sur tous les éléments de la bibliothèque.

Mais pour les fonctionnalité demandé cette solution ma paru assez lourde donc j'ai opter pour des collections imbriqué.

Ce choix nécessitait des traitements côté serveur pour extraire des données, mais il n'y a plus le besoin de faire plusieurs requêtes côté base de données.

J'ai fait aussi le choix de réaliser la base de données pour mieux analyser le problème et surtout pour utiliser au maximum Vert.X.

L'extraction des données (pages, articles) se fait dans les handlers,

La gestion du marque-page via les cookies, c'est l'option la plus simple, mais si le cookie est effacé l'utilisateur perd son marque-page.

La meilleure solution serait de mettre en place un système d'authentification via des tokens afin de garder des information en base de données , de plus ça respect le critère de Statless de REST.

Implementation

Une Class Java par entité, avec une notion de Chapitre abstrait qui suppose qu'il y ait des livres sans chapitres.
Un convertisseur Json et MongoClient de Vert.X permet la sauvegarder en base.

L'exécution du serveur s'est faite avec éclipse et les tests via le navigateur en saisissant les différentes URL

```
//voir toute la bibliothèque
/bibliotheque/list

//voir un livre donné
bibiliotheque/livres/:IDlivre

//voir un journal donné
/bibiliotheque/journaux/:IDJournal

//voir directement une page d'un livre
/bibiliotheque/livres/:IDlivre/:numPage

//voir directement une page d'un journal
/bibiliotheque/journaux/:IDJournal/:numPage;

//voir tous les articles d'une page de journal
/bibiliotheque/journaux/:IDJournal/:numPage/articles

//engistrement marque page
/bibiliotheque/*/*/*/marquepage

//recuperation marque page
/marquepage
```