TP : Web Services REST

MBDS

09/10/2017

Présentation Générale

L'objectif de ce TP est de vous faire réaliser une API REST en passant de l'analyse à la conception sans oublier la documentation.

Nous réaliserons l'API REST à l'aide du framework Grails, pour des raisons de simplicité à la mise en place et aussi pour faciliter la correction, vous travaillerez sur le SGBD H2 définit initialement dans le projet que vous créerez.

Ce TP est à faire seul à l'aide de tous les documents à votre disposition. De nombreuses ressources en ligne traitent du sujet, **soyez prudent quant à la pertinence des sources que vous utiliserez**, REST n'est pas un sujet très vaste, mais c'est par contre un sujet largement repris par des « non techniques » dans des versions édulcorées, dans le doute suivez les indications du cours.

Objectifs

Vous devez construire une API REST complète couvrant tous les besoins possibles sur un modèle de donnée simple, composé de deux entités liées par une **composition**.

Nous travaillerons sur l'exemple classique « Bibliothèque – Livre ».

La bibliothèque aura pour propriétés :

Un nom (String), une adresse (String), une année de construction (Int), une collection de livres (Livre)

Le livre aura pour propriétés :

Un nom (String), une date de parution (Date), un ISBN (String) et un auteur (String)

Vous devrez traiter les opérations GET / POST / PUT / DELETE sur les deux entités.

Vous devrez aussi ajouter la gestion de ressources liées, dans notre cas, donner la possibilité d'accéder à un livre en passant par la bibliothèque, ce qui pourrait donner une requête du genre :

GET /biblio/1/books pour récupérer la liste de tous les livres de la bibliothèque ayant l'id 1.

PUT /biblio/2/book/1 pour modifier le livre ayant l'id 1 référencé dans la bibliothèque ayant l'id 2.

Il va de soi que si le livre avec l'id 1 ne se trouve pas dans la bibliothèque ayant l'id 2, il faudra retourner un code d'état approprié.

Spécifications et Documentation (8 points)

Vous commencerez par réaliser un document de spécification complet sur l'API que vous allez réaliser, ce document devra contenir l'ensemble des méthodes mises à disposition par votre API.

Pour chaque méthode vous devrez produire :

- Une description fonctionnelle incluant les contraintes éventuelles
- Une description technique du service incluant
    - L'URL exposée pour le service en question
    - La structure des données en entrée du service
    - La structure des données en réponse du service
    - Un listing des cas d'erreurs gérés accompagnés d'une brève description

Le document devra se présenter comme un réel document de spécification, n'oubliez pas de présenter le contexte et d'être très clair dans votre rédaction.

Bien entendu, lors du rendu final, le document devra coller à 100% à l'implémentation que vous rendrez en même temps.

Implémentation (6 points)

Vous devez implémenter ladite API REST en Grails. La version à utiliser est la 3.3 et afin de limiter les problèmes de compatibilité et simplifier l'ensemble, merci de respecter cette consigne.

Vous devrez coder tout la partie métier sans utiliser les annotations (@Resources) ou les déclarations équivalentes dans le fichier UrlMappings (qui génèrent l'API REST à votre place).

Votre API devra permettre d'utiliser des méthodes GET / POST / PUT / DELETE dans le respect des principes de REST sur toutes les entités présentes.

Votre API devra donc au minimum être capable de répondre à 10 requêtes (5 pour chaque entité).

N'oubliez pas de gérer les cas d'erreur, utilisez les codes HTTP appropriés.

Procédure de Test (6 points)

Vous avez le choix de produire :

- Une collection Postman
- Un document listant des appels CURL

Cette procédure / ces tests devront – pour chaque méthode de l'API – exécuter et tracer un appel pour chacun des cas gérés

Les tests devront tracer à minima, les **appels** et **retours du serveur (incluant les headers),** vous pouvez aussi ajouter pour chaque appel, le résultat attendu.

Bonus – Sécurité (5 points)

Implémentez une couche de sécurité pour votre API.

Pour ce faire vous avez deux approches :

1) Faire une implémentation « manuelle » basée sur l'un des concepts décrits dans le cours
2) Utiliser Spring Security et son extension (autre plugin) pour la mise en place dans le cadre d'une API REST (Spring Security seul ne fera qu'une partie du travail, vous devrez impérativement utiliser l'extension « Spring Security Rest » en plus)

Dans tous les cas, vous devrez décrire dans le rendu une description de ce que vous avez mis en place et comment vous l'avez fait

Rendu

Ce projet sera noté à la fin des séances et constituera une part importante de votre notation dans ce module (moitié de votre note)

Tout projet rendu en retard se verra appliquer une pénalité de 2 points par jour de retard.

Vous devrez envoyer un mail à mon adresse (greg.galli@tokidev.fr), le **sujet du mail** sera formaté de la manière suivante :

Le rendu de ce projet devra se faire par mail au plus tard le 5 Novembre 2017 à minuit. Le sujet du mail devra comporter votre nom ainsi que la matière : Sujet : [WS REST] Votre Nom - Rendu de projet.

La soumission du projet se fera obligatoirement via un **git** sur lequel vous aurez **mis à jour régulièrement votre projet**.

Votre git devra contenir un **readme** précisant certains éléments que vous souhaiteriez éclaircir et **pour chaque bonus fait, vous expliquerez brièvement comment vous avez procédé et où je peux trouver les éléments relatifs à ces derniers.**

Si vous avez des questions, adressez-moi un mail à l'adresse greg.galli@tokidev.fr, je ferai de mon mieux pour revenir vers vous le plus rapidement possible.

Les projets copiés se verront systématiquement attribuer une note de 0.