CFA Institut National Supérieur des Technologies Avancées

Compte rendu

Introduction à l'événementiel sur une architecture multi-serveur

Gianni-Alessandro NGAMY
Ashanth CHANDRAMOHAN

Mai 2022

I) Introduction

Il nous a été demandé d'enrichir une architecture multi-serveur en Spring Boot, un framework Java. Nous avons en notre possession un domaine, un job-master et un worker. Le domaine nous servira à créer l'entité principale de notre application ainsi que sa base de données, et devra être exposé par une API. Le job-master nous servira à recevoir les ordres des utilisateurs et les traiter ou les déléguer. Et le worker nous servira à exécuter des tâches donner par le job-master.

Nous avons choisi d'implémenter le remote chunk, alors le reader sera dans le job-master et le writer dans le worker.

II) domaine-sqlite

Dans notre domaine, nous avons créé l'entité Letter, qui est le cœur de notre application, la transmission de Letter est l'objet de notre application. Un Letter est un message daté. Etant donné que nous créons une table stockant les Letter, nous avons créé un repository associé aux Letters (LetterRepository) qui nous permettra d'accéder au Letter qu'on recherche dans la base de données. Et pour finir, nous avons créé un controller qui va associer des routes à des actions (voir la/les Letter ou en écrire une nouvelle en base).

III) job-master-server

Dans notre job-master, nous avons créé le reader (LetterReader) qui lire le contenu d'un fichier csv contenant des Letter. De plus, nous avons créé 4 jobs (simpleJob, jobS, letterJob, letterCloudJob) qui sont des actions associées aux différentes routes incluses dans le controller du job-master, le LetterCloudJob est celui qu'on utilise dans notre démonstration. La configuration du job est contenue dans le fichier LetterJobCloudConfig, il lie le job-master au worker en remote chunk.

IV) worker-server

Dans notre worker, nous avons créé des services associés aux Letter, notamment le Processor, qui sert à créer un Letter à partir d'un message (Objet contenant une chaîne de caractères) et le Writer qui va écrire le Letter en base, par le le controller du Letter. Et pour finir, Nous lui avons associé une config (LetterChunkingConfig) qui va recevoir la requête du job-master et renvoyer une réponse.

Conclusion :

Nous avons réussi à créer une application multi-serveur de messagerie très simple, en utilisant le framework Java Spring et en implémentant la méthode remote chunk. Cette méthode est intéressante car elle permet d'avoir accès aux Letters sans dépendre du worker et permet d'alléger le travail du worker qui est le serveur qui permettra d'alimenter notre application.