Projet Microservice

AMRANI Said

RAVOAVY HARIMAMPIANINA Christian

Architecture de notre projet

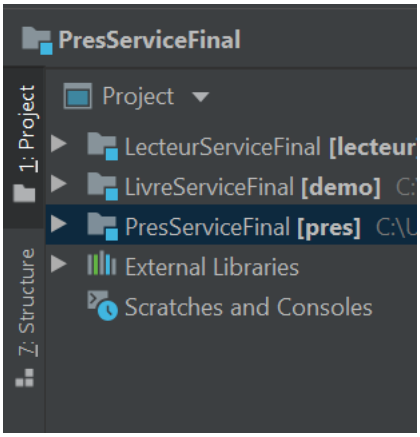
Notre projet se constitue de 3 microservice qui sont :

Service de livre : il fournit l'ajout, la suppression, la modification et le listing des livre en se basant sur la recherche par titre, par id, par édition …

Service lecteur : il fournit l'ajout, la suppression, la modification et le listing des lecteur

Service Pres : il permet l'ajout d'un emprunt, sa suppression, sa modification, l'historique d'emprunt, les livres emprunter par un lecteur …

La figure suivante montre la construction du projet dans intellij :



Structure du projet

Le projet est réalisé avec Maven

1.LivreServiceFinal

Pour le service livre voici le diagramme de classe

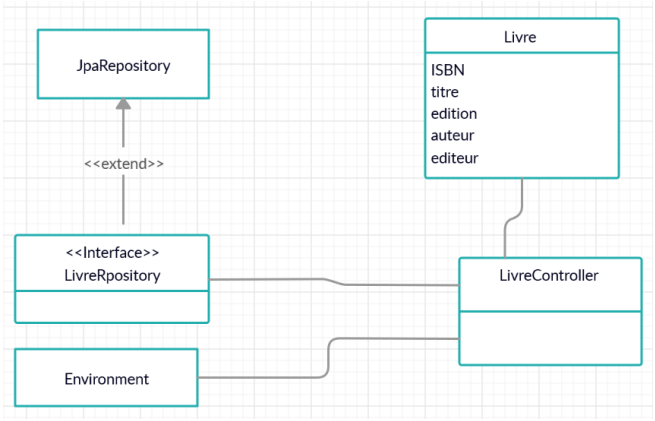


Diagramme de classe pour le service livre

Pour exécuter les différents microservice liés aux livres (ajout, suppression …) il faut se servir d'un client.

Dans notre cas on a utilisé Postman qui nous fournit toutes les méthode de l'architecture REST (POST, GET, DELETE, PUT …)

Voici le lien pour télécharger Postman : https://www.postman.com/downloads/

Le livreServiceFinal s'exécute sur le port 7000 et a une base de données H2 qui contient la table Livre avec comme attribut les attributs de la classe Livre.

Voici les services liés aux livres et comment les exécuté

Avec la méthode GET

Dans Postman, on choisit la méthode GET pour exécuter les différents services en utilisant ces URLs :

http://localhost:7000//Livre/{id} : récupère un livre en se basant sur son id

http://localhost:7000//LivreParTitre/{s} : récupère un livre par titre

http://localhost:7000/LivreParAuteur/{s} : récupère un livre par auteur

http://localhost:7000/allLivres : liste tout les livres

http://localhost:7000/ /Test/{inf}/{sup} : liste les livre qui sont entre deux dates d'édition différentes

Avec la méthode Put

Dans Postman, on choisit la méthode PUT on renseigne les chemins suivant :

http://localhost:7000//ModifierLivre : pour modifier un livre

Avec la méthode Post

http://localhost:7000/AjoutLivre : pour ajouter un livre

Avec la méthode Delete

http://localhost:7000/SupLivre/{id} : supprime un livre en se basant sur son id

2.LecteurServiceFinal

Le LecteurServiceFinal s'exécute sur le port 7001 et a une base de données H2 qui contient la table Lecteur avec comme attribut les attributs de la classe Lecteur.

Le diagramme de classe de ce service à la même architecture que celui de livreServiceFinal

Voici les services liés aux lecteurs et comment les exécuter

Avec la méthode GET

Dans Postman, on choisit la méthode GET on renseigne les chemins suivant :

http://localhost:7000/Livre/{id}: récupère un livre en se basant sur son id

http://localhost:7000/LivreParTitre/{s} : récupère un livre par titre

http://localhost:7000/LivreParAuteur/{s} : récupère un livre par auteur

http://localhost:7000/allLivres : liste tout les livres

http://localhost:7000/Test/{inf}/{sup} : liste les livre qui sont entre deux dates d'Edition différentes

Avec la méthode Put

Dans Postman, on choisit la méthode PUT on renseigne les chemins suivant :

http://localhost:7000//ModifierLivre : pour modifier un livre

Avec la méthode Post

http://localhost:7000/AjoutLivre : pour ajouter un livre

Avec la méthode Delete

http://localhost:7000/SupLivre/{id} : supprime un livre en se basant sur son id

3.SercicePresFinal

Le SercicePresFinal s'exécute sur le port 7002 et a une base de données H2 qui contient la table Pres avec comme attribut les attributs de la classe Pres.

Le diagramme de classe de ce service à la même architecture que celui de livreServiceFinal

Voici les services liés aux lecteurs et comment les exécuté

Avec la méthode GET

Dans Postman, on choisit la méthode GET on renseigne les chemins suivant :

http://localhost:7002/Pres/{date}: récupère un Pres en se basant sur son id

http://localhost:7002/HistoriqueDePres : récupère l'historique des Pres

http://localhost:7002/between/{inf}/{sup} : récupère tous les près qui sont compris entre deux dates différente par rapport à la date de près du livre

http://localhost:7002/TestByDate/{inf} : récupère tous les près qui enregistré dans une date donnée

http://localhost:7002/LivreEnCourEmprunt: récupère tous les livres qui sont en cours d'emprunt

http://localhost:7002/ LivreEmprunteParLecteur/{lecteur}: récupère tous les livres qui sont emprunté par un lecteur

Avec la méthode Post

http://localhost:7002/ajoutPres : pour ajouter ou modifier un Près

Avec la méthode Delete

http://localhost:7002/supPres/{id}: Pour supprimer un près en se basant par son id

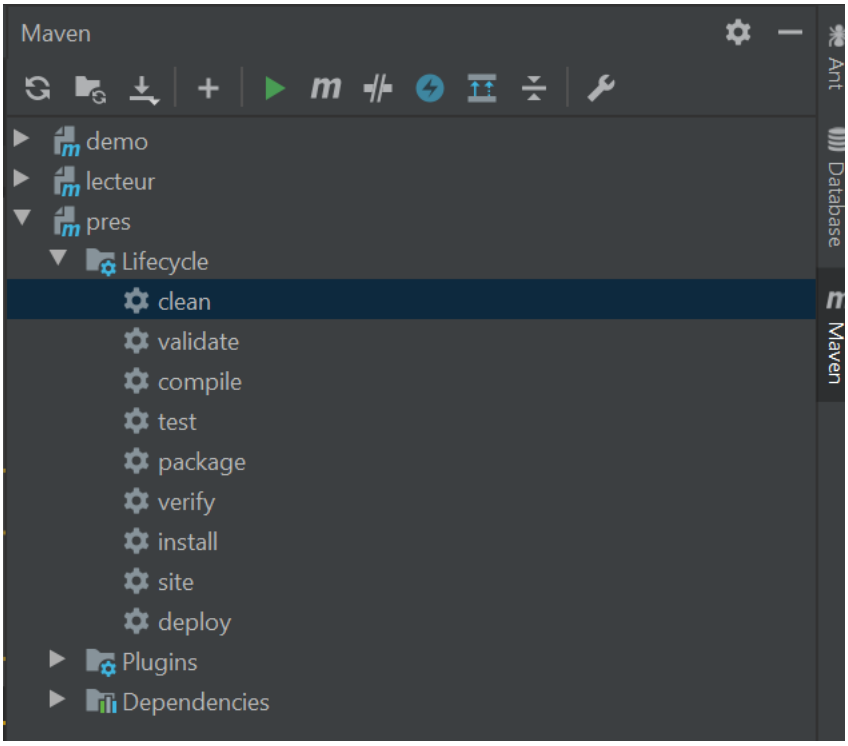
Docker

Pour exécuter le docker sur Windows il faut installer l'application Docker

Voici le lien de téléchargement : https://www.docker.com/products/docker-desktop

Pour exécuter le docker dans intellij il faut :

En haut a droite dans Intellij on a l'onglet Maven et nos 3 microservice comme le montre la figure :



Pour chaque microservice :

il faut exécuter un clean et un install.

Dans terminal il faut exécuter ses commandes :

```
C:\Users\Easyfront\IdeaProjects\LecteurServiceFinal>Docker build -f Dockerfile -t docker-spring-boot .
```

```
C:\Users\Easyfront\IdeaProjects\LecteurServiceFinal>docker run -p 7210:7001 docker-spring-boot
```

Remarque :

Pour chaque microservice il faut naviguer vers son répertoire comme entouré en rouge dans la première commande

Pour les port : le premier c'est celui dont on désire que l'image soit exécuter(on le choisit comme on veut) le deuxième c'est le port sur lequel notre application actuel s'exécute (7001 dans le cadre de LecteurServiceFinal°

Bilan

Ce que on a aimé :

- Une gestion de dépendances simplifié
- Un déploiement facilité
- Indépendance des services entre eux, par exemple on peut mettre en place un service avec une base de données SQL et un autre avec une base de données NoSQL sans aucun problème
- Maintenabilité : si un service tombe, le reste de l'application marchera normalement.
- Il contient un ensemble de fonctionnalités prêtes à l'emploi comme Spring Security.

Ce qu'on voulait mettre en place :

- Mettre en place la communication entre les microservice, par exemple le service près qui demande au service livre le titre d'un livre on lui transférant son identifiant.
- Mettre en place l'intégrité de données, c'est-à-dire on ne peut pas prêter un livre s'il n'est pas dans la base de données.