Spring Documentation personnelle

Dans ce document, j’expose la construction d’un site internet avec une architecture de type micro service. L’objectif de ce document est faire la démonstration d’une construction et d’un fonctionnement de Micro service avec la technologie Spring.

ce type d’architecture qui géré avec un service passerelle, un service d’enregistrement des micro services et des micro services permet la gestion du site.

Ce dit site sera composé en Front end du Framework Angulars (technologie Google) et de back end du Framework Spring et plus particulier de la partie bootable du Framework (Spring Boot).

Architecture du projet

Liste des dépendances

Liste des dépendances

La sélection des dépendances peut être réaliser depuis le site : <https://start.spring.io/>

* Sélection des dépendances pour la création d’un Micro service Spring classique avec une architecture Cloud.

**Spring Web :** Utilise un conteneur Apache tomcat par défaut, permet de construire une application Web de type MVC ou Micro Service.

**Spring Data JPA :** Couche d’accès aux données (JPA => Java Persistance application) qui est un ORM (Object Relation Mapping)

**Postgresql Divers :** Driver pour la connection a la base de données (le driver reste un choix de conception).

**Rest Repositories :** Permet d’exposer les données d’un service web via Spring Data Rest.

**Eureka Discovery Client :** Et un Micro service permettant d’enregistrer les micro services dans un registre. Ce registre permet dans une architecture distribuer l’équilibrage de charge client et dissocier les fournisseurs de service des consommateurs sans avoir besoin de DNS (Domain Name Server). Voir aussi : <https://spring.io/guides/gs/service-registration-and-discovery/>

**Spring Boot Actuator :** Permet de faire le monitoring des web Service présent dans l’architecture du projet.

**Lombok :** librairie d’annotation Java qui permet de générer pour le développeur en respectant les bonnes pratiques ce qu’on appelle du « boiler plate » c’est-à-dire :

* Getter/setter
* Equals/ hashcode
* toString
* Constructeurs
* Modificateur d’accès (private, protected)

**Spring Boot DevTools :** Outil de développement qui permet le redémarrage rapide et autre configuration voir : <https://www.baeldung.com/spring-boot-devtools>

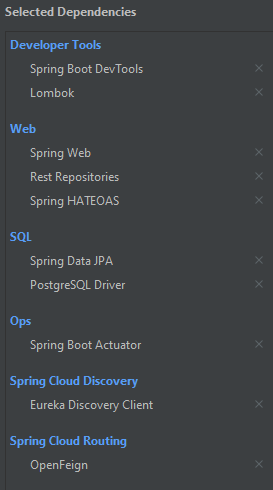
* Sélection des dépendances pour la création du Gateway (la passerelle)

1. **Spring Cloud Routing : Gateway** Qui est la passerelle entre les micro services
2. **Spring Cloud Discovery : Eureka Discovery Client**
3. **Ops : Spring Boot Actuator (Ops => operating system)**

* Sélection des dépendances pour la création du service de registre

**Spring Cloud Discovery :** Eureka server

* Sélection des dépendances pour la création du service de gestion de jonction entre deux micro service. Ce service aura pleinement sa fonction de micro service mais aura besoin de données provenant d’autre micro service. C’est pourquoi OpenFeign a était ajouté.

Voici une capture d’écran sur la liste des dépendances sélectionné dans l’IDE intellij IDEA

**Spring Web :** Utilise un conteneur Apache tomcat par défaut, permet de construire une application Web de type MVC ou Micro Service.

**Spring Data JPA :** Couche d’accès aux données (JPA => Java Persistance application) qui est un ORM (Object Relation Mapping)

**Postgresql Divers :** Driver pour la connection a la base de données (le driver reste un choix de conception).

**Rest Repositories :** Permet d’exposer les données d’un service web via Spring Data Rest.

**Eureka Discovery Client :** Et un Micro service permettant d’enregistrer les micro services dans un registre. Ce registre permet dans une architecture distribuer l’équilibrage de charge client et dissocier les fournisseurs de service des consommateurs sans avoir besoin de DNS (Domain Name Server). Voir aussi : <https://spring.io/guides/gs/service-registration-and-discovery/>

**Spring Boot Actuator :** Permet de faire le monitoring des web Service présent dans l’architecture du projet.

**Lombok :** librairie d’annotation Java qui permet de générer pour le développeur en respectant les bonnes pratiques ce qu’on appelle du « boiler plate » c’est-à-dire :

* Getter/setter
* Equals/ hashcode
* toString
* Constructeurs
* Modificateur d’accès (private, protected)

**Spring Boot DevTools :** Outil de développement qui permet le redémarrage rapide et autre configuration voir : <https://www.baeldung.com/spring-boot-devtools>

**OpenFeign :** Permet la communication entre micro service

**Spring HATEAOS :** HATEAOS => Hypermedia As The Engine of Application State. Ce format permet la sérialisation / désérialisation des objets Javade manière simplifier en un format JSON (JavaScript Object Notation).

Architecture Micro Service de Spring