esprit Se former autrement HONORIS UNITED UNIVERSITIES	Année Universitaire : 2024-2025 **Applications Web Distribuées** **UP_WEB**

Serveur de configuration

Mise en situation

Les microservices sont composés de nombreux petits services autonomes, chacun avec sa propre configuration. Plutôt qu'un fichier de propriétés centralisé (comme le monolithe), la configuration est dispersée sur plusieurs services, s'exécutant sur plusieurs serveurs. Dans un environnement de production, où il y'a probablement plusieurs instances de chaque service, la gestion de la configuration peut devenir une tâche lourde.

La configuration centralisée est un pattern dans lequel la configuration de tous les services est gérée dans un référentiel central plutôt que d'être dispersée sur des services individuels. Chaque service extrait sa configuration du référentiel central au démarrage.

Spring Cloud Config fournit un support côté serveur et côté client pour la configuration externalisée dans un système distribué. Avec le Config Server, on peut définir un emplacement central pour gérer les propriétés externes des applications dans tous les environnements.

Objectifs

- Créer un serveur de configuration
- Consulter les propriétés centralisées

Etapes

Etape 1: Créer le serveur de configuration - Config Server

1- Créer un projet spring avec les starters **Server config** et **Eureka discovery client.**

2- Dans le main, ajouter les 2 annotations :

@enableConfigServer : pour le définir comme un service de configuration
@EnableDiscoveryClient : pour que ce serveur apparaisse dans la liste Eureka

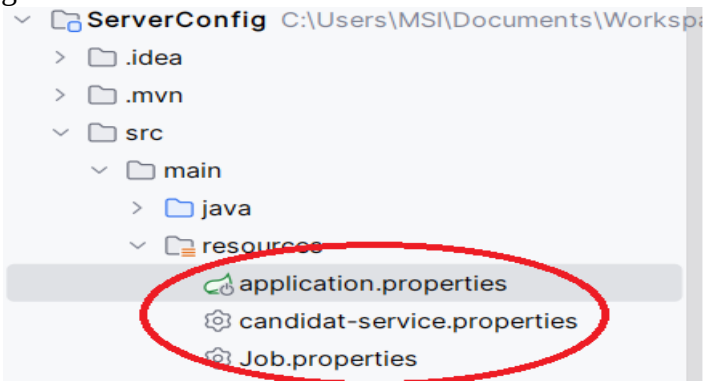
3- Ajouter les propriétés suivantes dans le fichier src/main/resources/application.properties du serveur de config

```
#eureka registration
spring.application.name=config-server
server.port=8888
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka
eureka.instance.prefer-ip-address=true
eureka.server.wait-time-in-ms-when-sync-empty=5
#eureka.client.register-with-eureka=true

spring.profiles.active=native
#spring.cloud.config.server.native.searchLocations=file://${user.home}

 spring.cloud.config.server.native.searchLocations=./src/main/resources
```

Etape 2: Préparer les configurations centralisées

Les configurations centralisées des autres microservices seront stockées, tel que mentionné dans le fichier précédent, sous le répertoire ./src/main/resources

Les fichiers de configurations centralisées seront structurés ainsi :



- **Application.properties** stockera les configurations de tous les microservices clients

- **candidat-service** stockera les configurations du microservice avec le nom candidat-service (le nom de l'application correspondant au MS candidat)

- **Job** stockera les configurations du microservice avec le nom Job (Le nom correspondant au MS Job)

Ci-après les propriétés avec lesquelles on va tester :

application.properties

```
spring.application.name=config-server

server.port=8888
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/
eureka.client.register-with-eureka=true

eureka.instance.prefer-ip-address=true
eureka.server.wait-time-in-ms-when-sync-empty=5

spring.profiles.active=native
#spring.cloud.config.server.native.searchLocations=file://${user.home}
spring.cloud.config.server.native.searchLocations=./src/main/resources
#spring.cloud.config.server.git.uri=https://github.com/my/servercloudconfig.git

welcome.messageGlobal= Welcome to spring cloud config-server. This is a global config to all client-config microservices
```

candidat-service.properties

```
welcome.message= Welcome to spring cloud config-server. This is a specific server-config to candidat
```

Job.properties

```
welcome.message= Welcome to spring cloud config-server. This is a specific server-config to Job
```

Etape 3: Configurer les clients du Config Server

Les microservices qui seront clients du config server (candidat-service et job) doivent être configurés comme suit, dans leurs fichiers **application.properties** :

```
spring.cloud.config.enabled=true
spring.cloud.config.uri=http://localhost:8888
spring.config.import=optional:configserver:http://localhost:8888/
```

Etape 4: Consulter les configurations

- Veuillez exécuter les projets suivants :

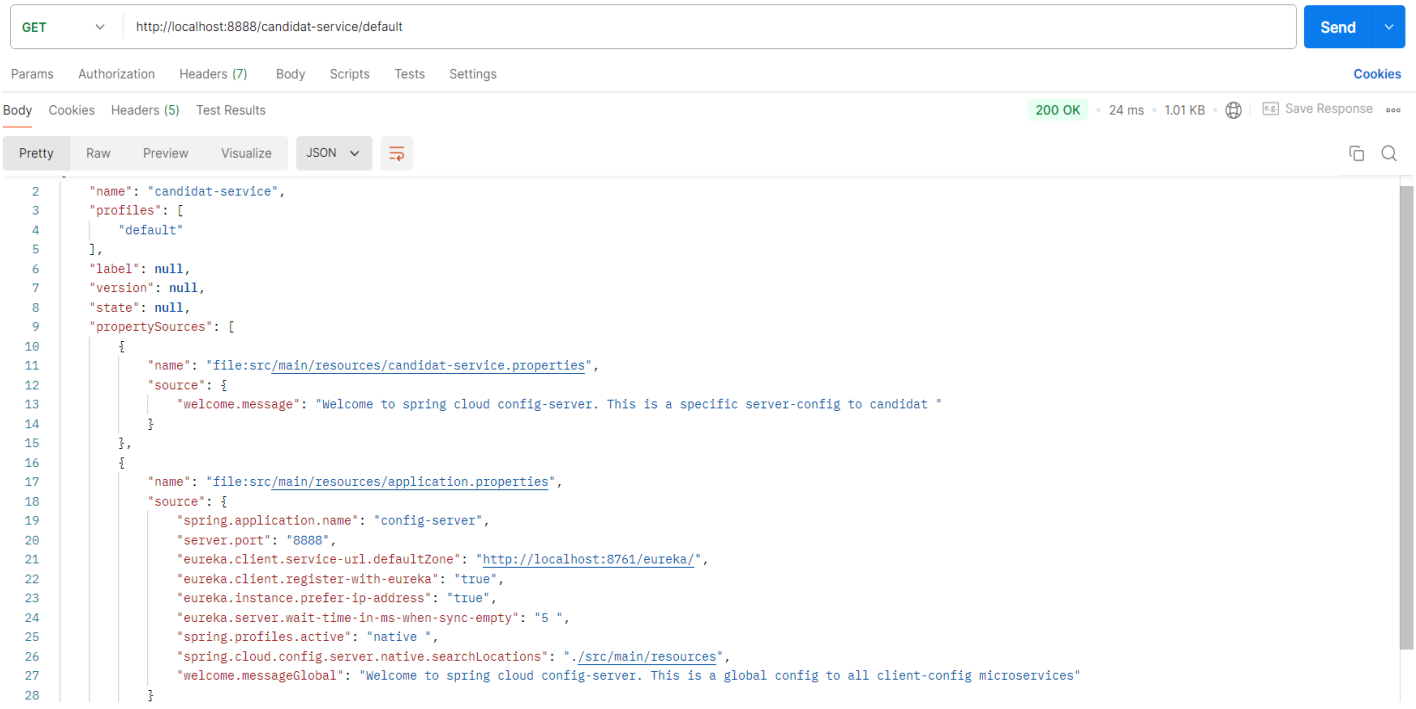
Eureka
Server config
Candidat

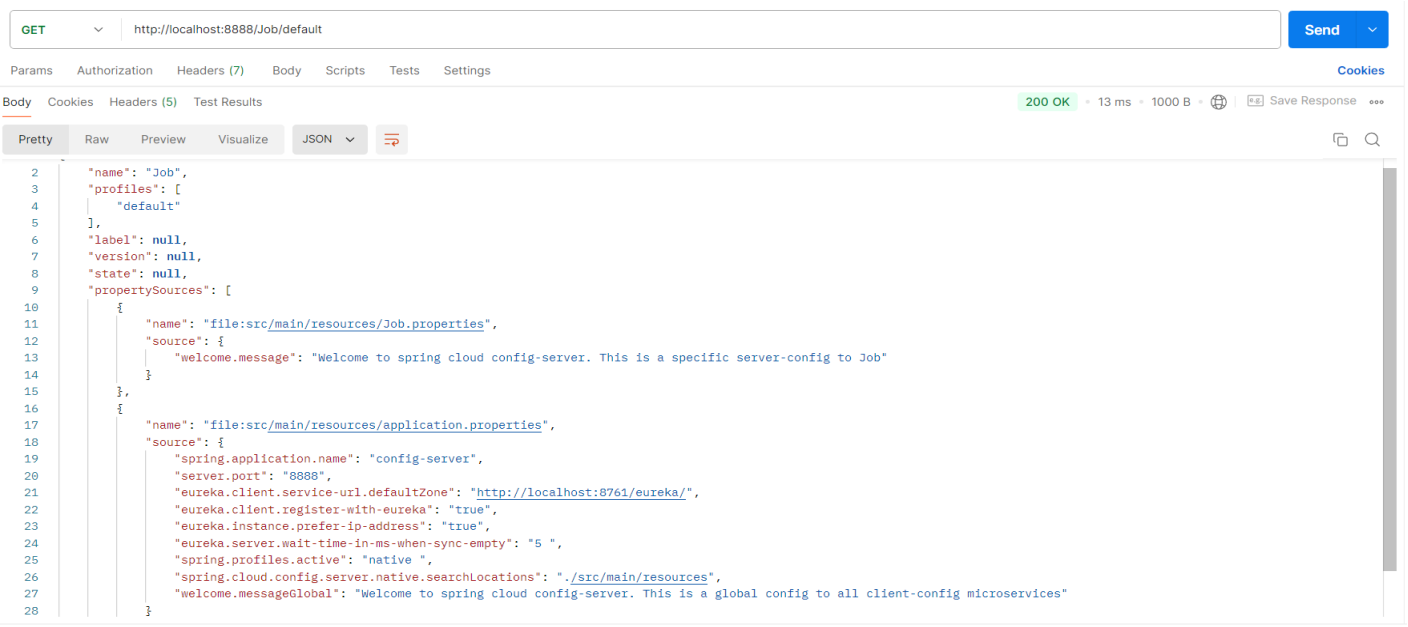
➔ Par la suite stopper Candidat et exécuter Job **tout en gardant les projets euereka et server config en exécution**

- Pour consulter les configurations prises du server config, par exemple celles du candidat-service, il suffit d'exécuter l'application du config-server sur le port 8888 et d'aller ensuite

sur le lien : http://localhost:8888/candidat-service/default

Les résultats attendus sont comme suit :





Plus de références

Ces liens sont utiles pour récupérer les propriétés des microservices en utilisant le **Git**
https://cloud.spring.io/spring-cloud-config/reference/html/

https://dzone.com/articles/microservices-and-spring-cloud-config-server