**Spring Cloud Config – 2022**

Create a config folder and keep all configuration files specific to environments like dev, prod etc and push to git(github or gitlab). Clone it in local directory as shown in the below image.

Graphical user interface, text, application

Description automatically generated

**Spring Cloud Config Server**

Create a small microservice called Cloud Config Server or your application name config server.

**pom.xml**

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.7.1</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<java.version>11</java.version>

<spring-cloud.version>2021.0.3</spring-cloud.version>

</properties>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>**spring-cloud-config-server**</artifactId>

</dependency>

**application.properties (src/main/resources)**

server.port=8888

spring.cloud.config.server.git.uri: <file://E:/CQRS/spring-cloud-configs> 🡸 Git local location

spring.cloud.config.server.git.default-label=**master** 🡸 git branch name

#Disable security of the Management endpoint

management.security.enabled=false

**Spring Boot Main Application**

**@EnableConfigServer 🡸 Important to enable Config Server**

@SpringBootApplication

**public** **class** CloudConfigServerApp {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(CloudConfigServerApp.**class**, args);

}

}

**Create Spring Cloud Config Client**

This is a normal microservice which uses Spring Cloud Config Server to get the properties values. It can be any microservice.

**pom.xml**

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.7.1</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<java.version>11</java.version>

<spring-cloud.version>2021.0.3</spring-cloud.version>

</properties>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>**spring-cloud-starter-config**</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId> 🡸 It used for **@RefreshScope**

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

**application.properties (src/main/resources)**

server.port=8080

**spring.config.import=optional:configserver:http://localhost:8888** **🡸 Important**

#spring.profiles.active=dev

spring.profiles.active=prod 🡸 Environment Name

spring.application.name=app1 🡸 Name of your application

#you call the below method to get the refreshed value

#localhost:8080/actuator/refresh

**management.endpoints.web.exposure.include=\* 🡸 For actuator refresh**

**Spring Boot Main Application**

@SpringBootApplication

**public** **class** CloudConfigClientApp {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(CloudConfigClientApp.**class**, args);

}

}

**Controller**

**@RefreshScope 🡸**

@RestController

**public** **class** TestController {

@Value("${user.role:hello}")

**private** String role;

@Value("${db.name}")

**private** String message;

@GetMapping("/what/message")

**public** ResponseEntity<String> whatIsMessage() {

**return** **new** ResponseEntity<>(message, HttpStatus.***OK***);

}

}

To get the refreshed value, make a post call to the actuator URL: **localhost:8080/actuator/refresh**

Just make a call and you will get the data.

GET http://localhost:8080/what/message

**Github location**

<https://github.com/debjava/spring-cloud-config-2022> 🡸 Config Server

<https://github.com/debjava/spring-cloud-config-client-2022> 🡸 Config Client or any other service

<https://github.com/debjava/spring-cloud-configs> 🡸 All configuration files