Eureka Server

Eureka Server Provides service for registration and discovery it can register other client (Microservices) and client applications (Microservices) can discover other client through this server.

Step 1: Goto Window Menu -> Change Perspective ‘Spring’

Step 2: Goto File Menu -> New -> Spring Starter Project -> Type Project Name -> Next -> Select EurekaDiscovery (Service discovery using spring-cloud-netflix and Eureka) and Ureka Server (spring-cloud-netflix Eureka Server) -> Click on Finish

Step 3: Annotate @EnableEurekaServer at Spring runner class to enable this microservice as Eureka server to discover and register other client (Microservices).

Step 4: Goto application properties/yml file and specify following:

#Default port is 8761 for Eureka Server

server.port=8761

# To specify hostname

eureka.instance.hostname=localhost

#these two properties disable this server to resister himself as client set this property false because it will act as server. It will look for other client to register

eureka.client.register-with-eureka=false

eureka.client.fetch-registry=false

Eureka Client

Step 1: Goto Window Menu -> Change Perspective ‘Spring’

Step 2: Goto File Menu -> New -> Spring Starter Project -> Type Project Name -> Next -> Select EurekaDiscovery (Service discovery using spring-cloud-netflix and Eureka) and Ureka Server (spring-cloud-netflix Eureka Server) -> Click on Finish

Step 3: Annotate @EnableEurekaClient/@EnableDiscoveryClient (It will provide metadata information and every time it will send heartbeat to Eureka Server.) at Spring runner class to enable this microservice as Eureka Client to register itself with Eureka Server so that other client (Microservices) can discover it.

Step 4: Goto application properties/yml file and specify following:

#Port to run client

server.port=8080

#Service name for lockup for other client

spring.application.name=student-service

#Eureka Server fallback URL where this client will register itself. Note: if you are running multiple Eureka Server Instance then you can give multiple Instance URL of Eureka Server.

eureka.client.service-url.defaultZone=http://localhost:8761/eureka

#Eureka Client instance ID. Note: Default Instance Id will be as ‘SERVICE-NAME/PHISICAL-MACHIME-NAME:PORT’

eureka.instance.instance-id=${spring.application.name}:${random.port}

Reference: <https://docs.pivotal.io/spring-cloud-services/1-4/common/index.html>

Config Server (File System)

Step 1: Goto Window Menu -> Change Perspective ‘Spring’

Step 2: Goto File Menu -> New -> Spring Starter Project -> Type Project Name -> Next -> Select Config Server -> Click on Finish

Step 3: Annotate @EnableConfigServer (It will add capability for Config Server.) at Spring runner class to enable this microservice as Config Server to provide central configuration details (like db connection details, SMTP details) for other Client (Microservices) for consumtion.

Step 4: Goto application.properties/yml file and specify following:

#Port to run Config Server

server.port=8888

Step 5: Goto resources folder and create bootstrap.properties file.

Note: bootstrap.property file get loaded before all resources so that those configuration which required get loaded before application.property must be placed inside it.

#NATIVE enables file system property setup. By default it enabled for GIT

spring.profiles.active=native

#By default CONFIG is default search location when you enable native profile

spring.cloud.config.server.native.search-locations=classpath:/config, classpath:/app1, classpath:/app2

**Step 6:** Goto resources folder and create project Wise folder: Ex. app1, app2 etc. And create different projectwise property files like myapp1-production.properties, myapp2-production.properties etc

And place all configuration details like this

portal.jdbc.url=jdbc:mysql://localhost:3306/prodApp1

portal.jdbc.user=rootProd

portal.jdbc.password=mysqlProd

**Step 6:** Now your Config server is ready to provide configuration details from file system, Click on run button and type following URL to get configuration details.

<http://localhost:8888/myapp1/production>

Config Client

**Step 1:** Goto Window Menu -> Change Perspective ‘Spring’

**Step 2:** Goto File Menu -> New -> Spring Starter Project -> Type Project Name -> Next -> Select Config Client [You can select MVC and other required dependencies for your module] -> Click on Finish

**Step 3:** Goto application.properties/yml file and specify following:

#Port to run Config Server

server.port=8080

**Step 4:** Goto resources folder and create bootstrap.properties file.

Note: bootstrap.property file get loaded before all resources so that those configuration which required get loaded before application.property must be placed inside it.

#application name play very important role it is basically property file name in your Config Server

spring.application.name=myapp1-production

spring.profiles.active=default

#Fallback URI of Config Server

spring.cloud.config.uri=http://localhost:8888

**Step 5:** Create ReST endpoint to check whether your config client is able to get configuration details from Config Server or Not

@RestController

**public** **class** CommonController {

@Value("${portal.jdbc.url:default url}")

String url;

@Value("${portal.jdbc.user:default user}")

String userName;

@Value("${portal.jdbc.password:default password}")

String password;

@RequestMapping(value="/getProperty", method=RequestMethod.***GET***)

**public** String displayProperty() {

Map<String,String> m=**new** HashMap<String,String>();

m.put("url", url);

m.put("userName", userName);

m.put("password", password);

**return** m.toString();

}

}

Config Server Security

**Step 1:** Add following dependency in Config Server POM

<dependency>

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

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

</dependency>

**Step 2:** Set following property in bootstrap.properties file of Config Server

security.basic.enabled=true

security.user.name=rootUser

security.user.password=1234

**Step 3:** Set following property in bootstrap.property file of Config Client

spring.cloud.config.username=rootUser

spring.cloud.config.password=1234