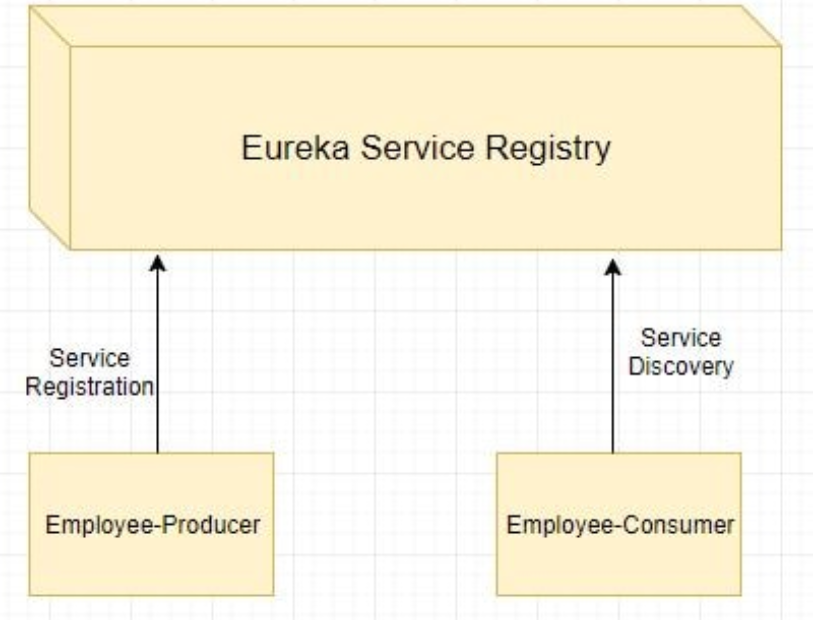
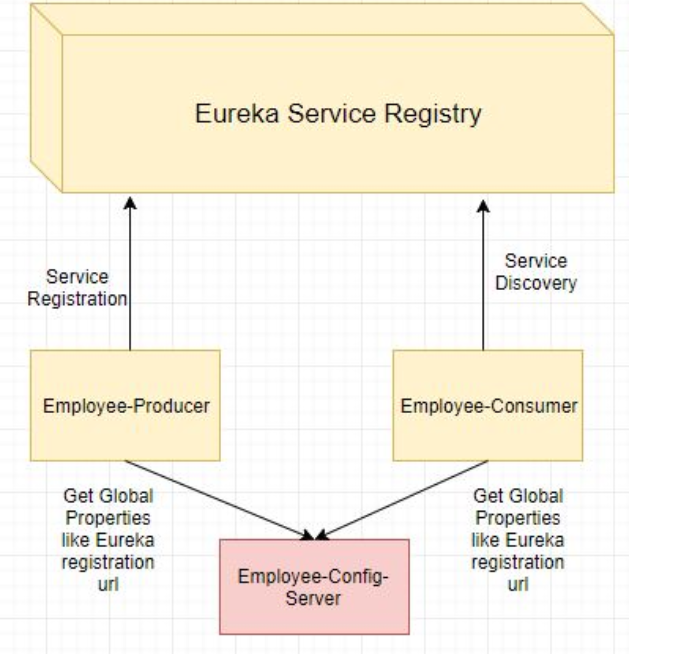
What is Spring Cloud Config ?Need for it?

Spring Cloud Config provides **server** and **client-side** support for **externalized** **configuration** in a distributed system.

With the **Config Server** you have a **central place** to manage **external properties** for applications **across all environments**.



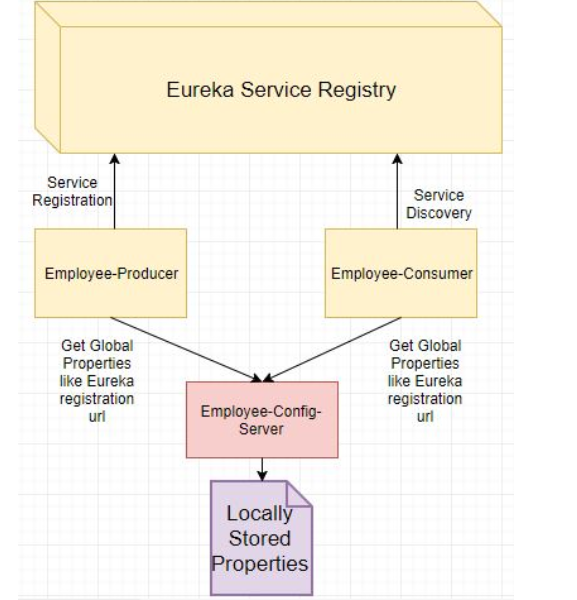
Modules can have common **global properties** which are repeated in all the modules. For example we have properties related to Database etc. For example in our employee-consumer and employee-producer we are having the following property for registering to Eureka Server.



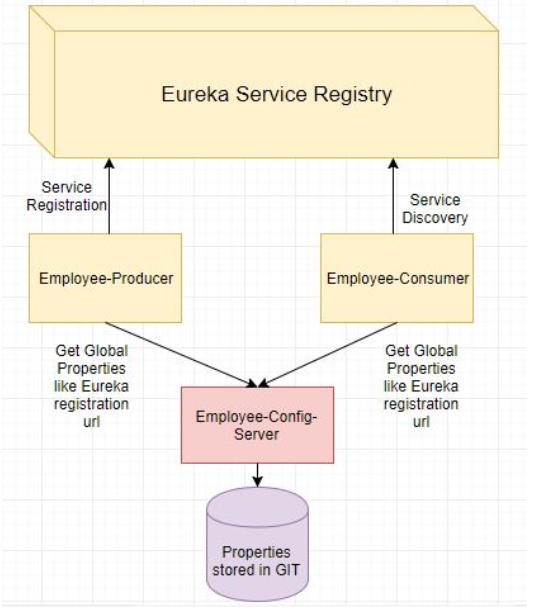
How Spring Cloud Config Works?

Spring Cloud Config Server can be either configured in following 2 ways-

**Using Local File System -**Properties to be externalized are stored in the local file system of the Spring Cloud Config Server.

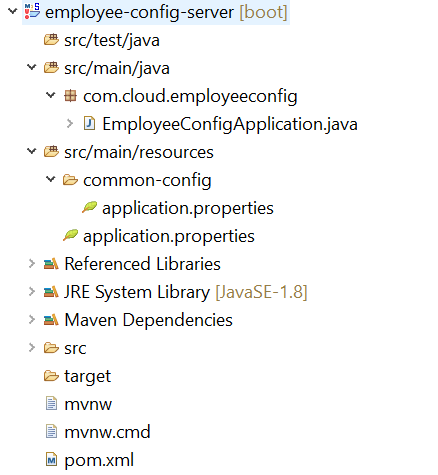


**Using GIT Repo -**Properties to be externalized are stored in the GIT Repo.



Let’s Begin

1. We will be creating a new module name employee-config-server having the externalized Eureka registration property.
2. Next we will the make code changes to the employee-producer module to get the required Eureka registration property from the employee-config-server.



The pom.xml will be as follows with the **spring-cloud-config-server**.

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cloud</groupId>

<artifactId>employee-config-server</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>employee-config</name>

<description>Demo project for Spring Boot</description>

<parent>

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

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

<version>1.4.1.RELEASE</version>

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

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

**<dependency>**

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

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

**</dependency>**

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

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

<artifactId>spring-cloud-dependencies</artifactId>

<version>Camden.SR6</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

**<resources>**

**<resource>**

**<filtering>true</filtering>**

**<directory>src/main/resources</directory>**

**<includes>**

**<include>\*.properties</include>**

**<include>common-config/</include>**

**</includes>**

**</resource>**

**</resources>**

</build>

</project>

Next define the properties in application.properties. The **spring.profiles.active=native** property tells the config module to look for the externalized properties locally.

spring.profiles.active=native

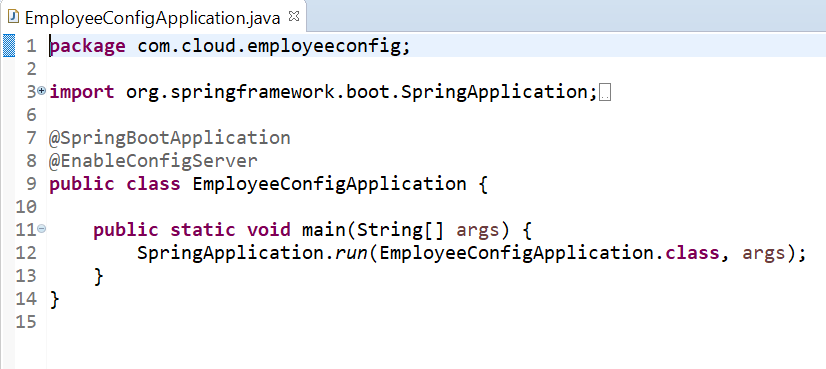
server.port=8888

spring.cloud.config.server.native.search-locations=classpath:/common-config

Next create a folder named **common-config**, and inside it create the **application.properties.** This is where we will store the **global common properties** to be used by other microservices.

eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka

Finally we annotate the Spring Boot Main class with **@EnableConfigServer**.With this the module will act as a config server.



Code changes for employee-producer

The changes we make for the consumer module are

* Add the Spring Cloud Server Config dependency in the pom.xml  
  The pom.xml will be as follows-
* <?xml version="1.0" encoding="UTF-8"?>
* <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
* xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
* <modelVersion>4.0.0</modelVersion>
* <groupId>com.cloud</groupId>
* <artifactId>employee-producer</artifactId>
* <version>0.0.1-SNAPSHOT</version>
* <packaging>jar</packaging>
* <name>SpringBootHelloWorld</name>
* <description>Demo project for Spring Boot</description>
* <parent>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-starter-parent</artifactId>
* <version>1.4.1.RELEASE</version>
* <relativePath /> <!-- lookup parent from repository -->
* </parent>
* <properties>
* <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
* <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
* <java.version>1.8</java.version>
* </properties>
* <dependencies>
* **<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.cloud</groupId>
* <artifactId>spring-cloud-starter-eureka</artifactId>
* </dependency>
* </dependencies>
* <dependencyManagement>
* <dependencies>
* <dependency>
* <groupId>org.springframework.cloud</groupId>
* <artifactId>spring-cloud-dependencies</artifactId>
* <version>Camden.SR6</version>
* <type>pom</type>
* <scope>import</scope>
* </dependency>
* </dependencies>
* </dependencyManagement>
* <build>
* <plugins>
* <plugin>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-maven-plugin</artifactId>
* </plugin>
* </plugins>
* <resources>
* <resource>
* <filtering>true</filtering>
* <directory>src/main/resources</directory>
* <includes>
* <include>\*.properties</include>
* </includes>
* </resource>
* </resources>
* </build>
* </project>

Remove the Eureka property from the employee producer module

Next start the modules in following sequence-

* employee-config-server

Type the url : <http://localhost:8888/application/default>

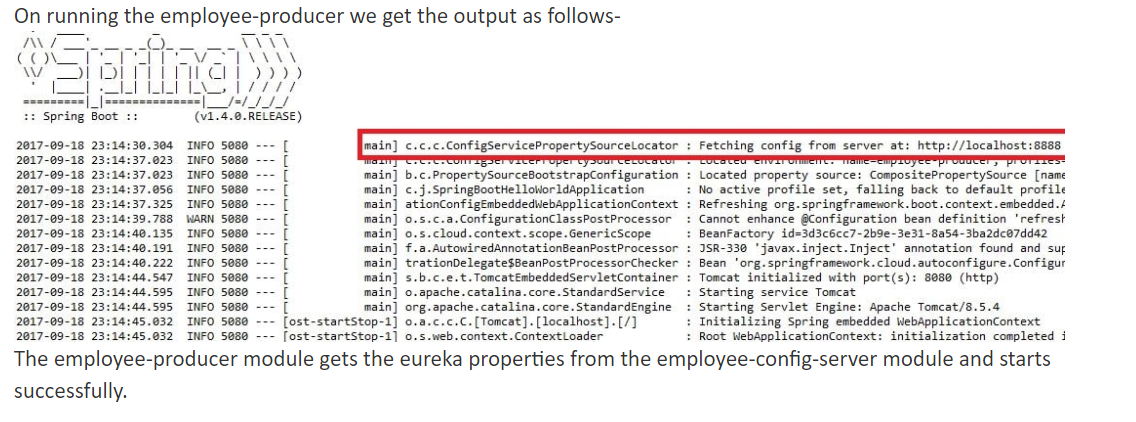
You will get the details of properties of profiles.

Sources":[{"name":"classpath:/common-config/application.properties","source":{"eureka.client.serviceUrl.defaultZone":"http://localhost:8761/eureka"}}]}

This informs us the program runs perfectly.

* eureka-server
* Start the eureka server
* employee-producer
* Start the employee producer and see the details like below screen

On running the employee-producer we get the output as follows-



This shows that employee-producer is trying to fetch details from config server.

Finally run the eureka server to see the employee-producer instance registered or not

@@@@@@@@@@@@@@@@ Happy Learning @@@@@@@@@@@@@@@@@@@@