**Spring Cloud Service Discovery and Client along with Feign Client – 2024**

Spring Eureka Service Discovery is a client-side service discovery mechanism for microservices, using Netflix Eureka. It allows services to dynamically discover and communicate with each other without hard-coding hostnames and ports.

**How to create Spring Discovery Server (aka Eureka Server)**

**Pom.xml relevant portion**

<parent>

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

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

<version>3.3.6</version>

<relativePath />

</parent>

<properties>

<java.version>21</java.version>

<spring-cloud.version>2023.0.3</spring-cloud.version>

</properties>

<dependencies>

<dependency>

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

<artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

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

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

<version>${spring-cloud.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

**Main Class**

@SpringBootApplication

@EnableEurekaServer 🡸 Important

**public** **class** SpringDiscoveryEurekaServerApp {

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

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

}

}

Application.properties

spring.application.name=eureka-server **🡨 Just a name**

server.port=8761 **🡨 Default Port**

**eureka.client.register-with-eureka=false 🡨 Important**

**eureka.client.fetch-registry=false 🡨 Important**

How to Access Discovery URL: [**http://localhost:8761/**](http://localhost:8761/)

**How to Use Discovery Client**

We will create the below ordinary microcservices with different port along eureka discovery client.

|  |  |  |
| --- | --- | --- |
| Name | Port | Description |
| sample-app1-service | 8081 | Ordinary Microservice |
| sample-app2-service | 8082 | Ordinary Microservice |
| sample-app3-service | 8083 | Microservice with Feign Client |

**For sample-app1-service and sample-app2-service microservices, code given below.**

**Pom.xml**

<parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.3.6</version>  
 <relativePath />  
</parent>  
  
<properties>  
 <java.version>17</java.version>  
 <spring-cloud.version>2023.0.0</spring-cloud.version>  
</properties>  
  
<dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  
 </dependency>  
</dependencies>  
<dependencyManagement>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-dependencies</artifactId>  
 <version>${spring-cloud.version}</version>  
 <type>pom</type>  
 <scope>import</scope>  
 </dependency>  
 </dependencies>  
</dependencyManagement>

**application-dev.properties for sample-app1-service**

spring.application.name=sample-app1-service  
server.port=8081  
server.servlet.contextPath=/sample-app1-service  
  
spring.cloud.compatibility-verifier.enabled=false

**application-dev.properties for sample-app2-service**

spring.application.name=sample-app2-service  
server.port=8082  
server.servlet.contextPath=/sample-app2-service  
  
spring.cloud.compatibility-verifier.enabled=false

**There will be a simple Main class annotated with @SpringBootApplication**, no other annotations required.

**For sample-app3-service**

**Pom.xml relevant portion**

<parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.3.6</version>  
 <relativePath />  
</parent>  
  
<properties>  
 <java.version>17</java.version>  
 <spring-cloud.version>**2023.0.0**</spring-cloud.version>  
</properties>  
  
<dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 **<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-openfeign</artifactId>  
 </dependency>**  
 **<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  
 </dependency>**  
</dependencies>  
<dependencyManagement>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-dependencies</artifactId>  
 <version>${spring-cloud.version}</version>  
 <type>pom</type>  
 <scope>import</scope>  
 </dependency>  
 </dependencies>  
</dependencyManagement>

**application-dev.properties**

spring.application.name=sample-app3-service  
server.port=8083  
server.servlet.contextPath=/sample-app3-service  
spring.cloud.compatibility-verifier.enabled=false 🡸 To avoid conflict with Spring Cloud verion

**SpringBoot Main Application Class**

**@SpringBootApplication  
@EnableFeignClients 🡸 For FeignClient  
@EnableDiscoveryClient 🡸 For Discovery client to be used in FeignClient Interface**public class SampleApp3Service {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(SampleApp3Service.class, args);  
 }  
}

**FeignClient Interface**

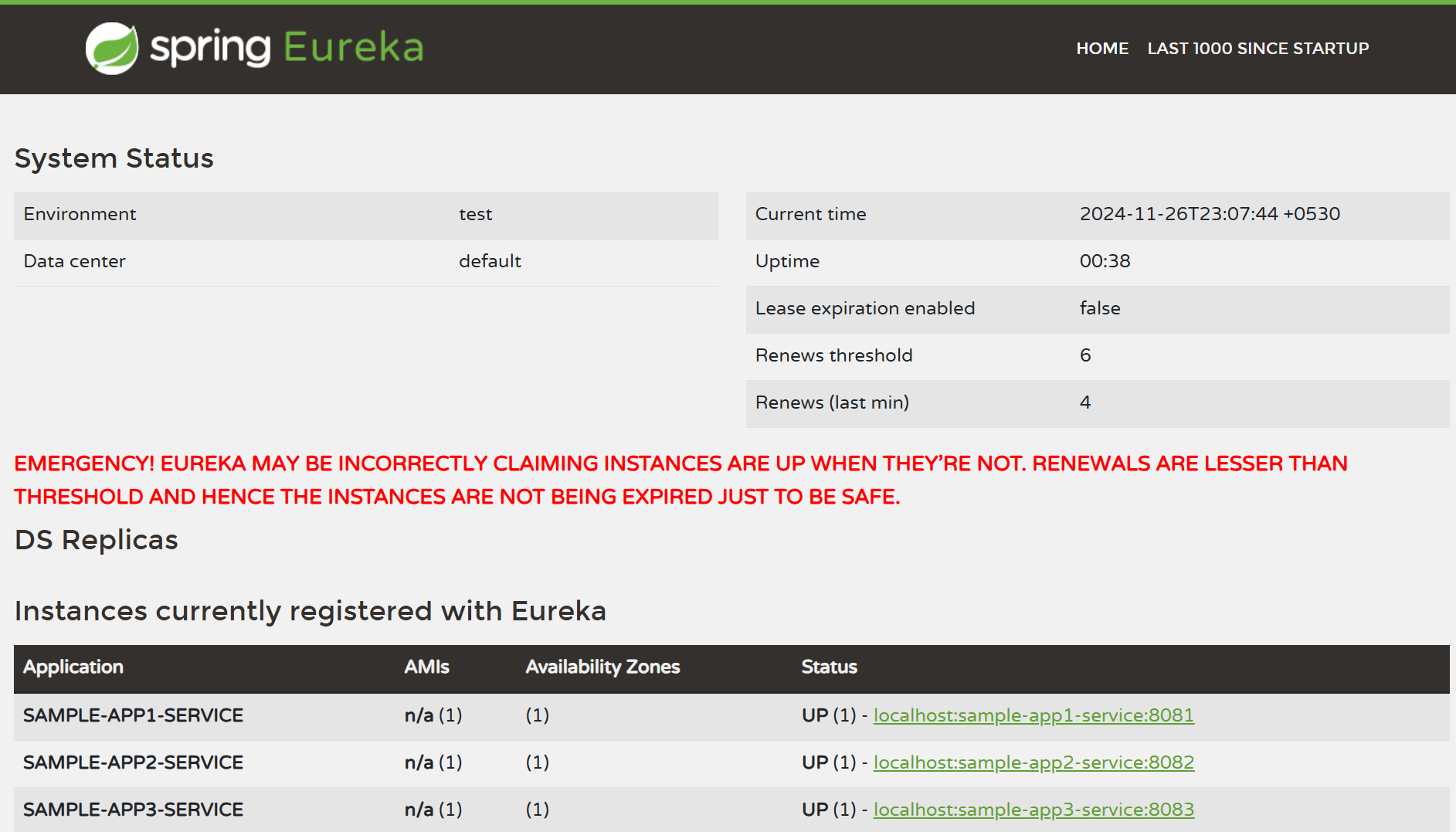
@FeignClient(name = "DISCOVERYFEIGNCLIENT",

url = "https://this-is-a-dummy-url-or-placeholder.com") 🡨 Dummy url  
public interface AppFeignClient {  
   
 @GetMapping(path = "/v1/info")  
 String getInfo(URI baseUri);  
}

**AutoRun class is given below.**

@Component  
public class AutoRun {  
  
 **@Autowired  
 private AppFeignClient feignClient;  
  
 @Autowired  
 private DiscoveryClient discoveryClient;**  
 public void invokeFeign() {  
 System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* RESPONSE USING FEIGN CLIENT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
 **URI baseURI = URI.*create*("http://localhost:8081/sample-app1-service");  
 String infoText = feignClient.getInfo(baseURI);** System.*out*.println("Info Text: "+infoText);  
 }  
  
 public void invokeFeignUsingDiscovery() {  
 System.*out*.println("discoveryClient----->"+discoveryClient);  
 **discoveryClient.getServices().forEach(value -> System.*out*.println("Service: "+value));**  
  
 System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* RESPONSE USING FEIGN CLIENT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
 String serviceName = "sample-app1-service";  
 **List<ServiceInstance> instances = discoveryClient.getInstances(serviceName);** if(!instances.isEmpty()) {  
 **ServiceInstance serviceInstance = instances.get(0);** **String serviceId = serviceInstance.getServiceId();** System.*out*.println("Service Id: "+serviceId); *//SAMPLE-APP1-SERVICE* URI rootURI = serviceInstance.getUri();  
 System.*out*.println("ROOT URI: "+rootURI); *// http://localhost:8081* **URI baseURI = URI.*create*(rootURI+"/"+serviceName);**  
 System.*out*.println("Base URI: "+baseURI);  
 **String infoText = feignClient.getInfo(baseURI);** System.*out*.println("Info Text: "+infoText);  
 }  
 }  
  
 @EventListener(ApplicationReadyEvent.class)  
 public void run() {  
 System.*out*.println("Sample App3 Application started running ...");  
 invokeFeign();  
 invokeFeignUsingDiscovery();  
 }  
}

**Sample Image of Discovery using localhost:8761**

****