**Spring Cloud Service Discovery With Eureka and Feign Client - 2025**

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>

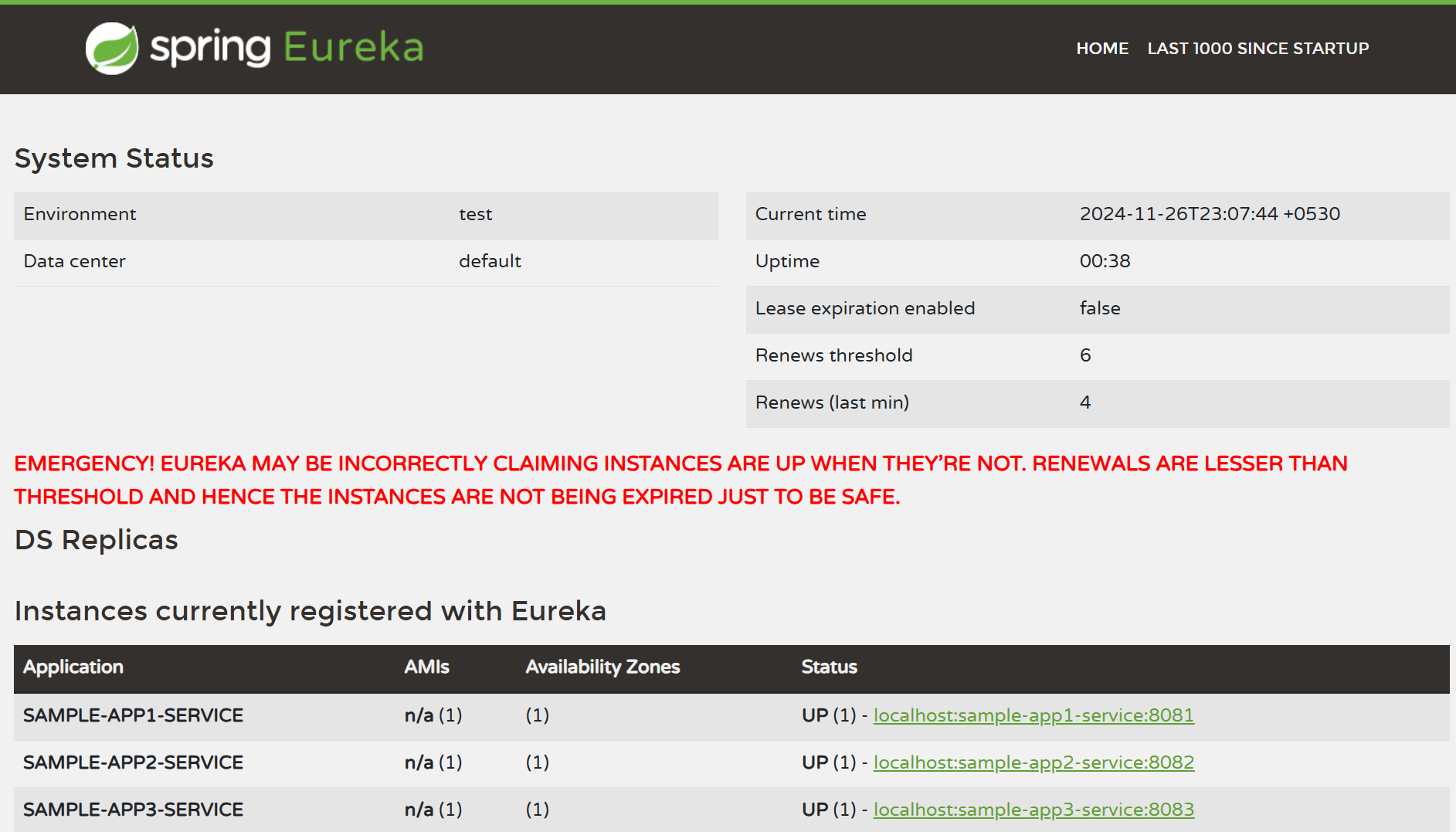
**application-dev.properties**

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);  
 }  
}

**Enter the url localhost:8761 in the browser, you will see the below image.**

****

**Create a small microservice with Feign Client to call the end point/s from sample-app3-service**

**Microservice Name: feignclient-with-eureka-discovery**

**Pom.xml**

**You have use the following dependency along with Spring Cloud BOM.**

<dependency>

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

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

</dependency>

**Main Class**

@SpringBootApplication

**@EnableFeignClients 🡸 Use it for Feign Client**

**public** **class** FeignClientForEurekaDiscoveryApp {

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

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

}

}

**application.properties**

**server.port=8085**

**Define a Sample Feign Client**

@FeignClient(name = "SomeFeignClientName", url = "http://localhost:8080") 🡸 Dummy URL

**public** **interface** SampleFeignClient {

@GetMapping("/hello/{id}")

String getHello(URI baseUri, @PathVariable("id") String id);

}

**Case-1: If you want to call the above feign client by passing the URI, use the code in AutoRun class**

@Autowired

**private** SampleFeignClient client;

@Autowired

**private** DiscoveryClient discoveryClient;

**private** String discoveryServiceName = "sample-app3-service";

**public** **void** invokeServiceUsingBaseURI() {

List<ServiceInstance> list = discoveryClient.getInstances(discoveryServiceName);

URI baseURI = **null**;

**if**(!list.isEmpty()) {

baseURI = list.get(0).getUri();

}

System.***out***.println("What is base URI : "+baseURI); // localhost:8083

URI mainUri = URI.*create*(baseURI.toString()+"/"+discoveryServiceName);

String response = client.getHello(mainUri,"123");

System.***out***.println("Response: "+response);

}

**Case-2: Invoke/Call Feign Client simply with a name**

**Create another Feign Client like this**

// Below is the name of the application to be invoked by Feign Client

@FeignClient(name="SAMPLE-APP3-SERVICE",path="/sample-app3-service") //path is required in case of Eureka

**public** **interface** AnotherFeignClient {

@GetMapping("/hello/{id}")

String getHelloFromService(@PathVariable("id") String id);

}

**Note: In the above case name should be in upper case and, you have to provide the path or context path**

**You have to use the below code in AutoRun class**

**public** **void** invokeDirectService() {

String response = anotherClient.getHelloFromService("123");

System.***out***.println("Another client Response: "+response);

}

**Complete AutoRun class is given below**

@Component

**public** **class** AutoRun {

@Autowired

**private** SampleFeignClient client;

@Autowired

**private** DiscoveryClient discoveryClient;

**private** String discoveryServiceName = "sample-app3-service";

@Autowired

**private** AnotherFeignClient anotherClient;

**public** **void** showServiceDetails() {

discoveryClient.getServices().forEach(value -> System.***out***.println("Service Name: "+value));

}

**public** **void** invokeServiceUsingBaseURI() {

List<ServiceInstance> list = discoveryClient.getInstances(discoveryServiceName);

URI baseURI = **null**;

**if**(!list.isEmpty()) {

baseURI = list.get(0).getUri();

}

System.***out***.println("What is base URI : "+baseURI);

URI mainUri = URI.*create*(baseURI.toString()+"/"+discoveryServiceName);

String response = client.getHello(mainUri,"123");

System.***out***.println("Response: "+response);

}

**public** **void** invokeDirectService() {

String response = anotherClient.getHelloFromService("123");

System.***out***.println("Another client Response: "+response);

}

@EventListener(ApplicationReadyEvent.**class**)

**public** **void** run() {

System.***out***.println("App started running ...");

// showServiceDetails();

// invokeServiceUsingBaseURI();

invokeDirectService();

}

}

**Notes:**

**To get the list of service, use the below code**

**discoveryClient.getServices()**

**.forEach(value -> System.*out*.println("Service Name: "+value));**

**To know about a particular service, use the below code**

**List<ServiceInstance> list = discoveryClient.getInstances(discoveryServiceName);**