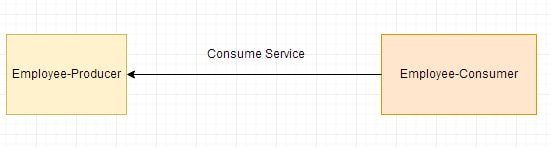
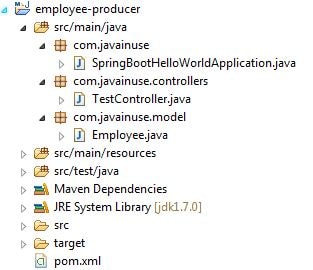
Overview of Netflix components-

Spring Cloud Netflix provides Netflix OSS integrations for Spring Boot apps through autoconfiguration and binding to the Spring Environment and other Spring programming model idioms. With a few simple annotations you can quickly enable and configure the common patterns inside your application and build large distributed systems with battle-tested Netflix components. The patterns provided include Service Discovery (Eureka), Circuit Breaker (Hystrix), Intelligent Routing (Zuul) and Client Side Load Balancing (Ribbon).

|  |  |
| --- | --- |
| **Netflix Component Name** | **Functionality** |
| Eureka | Service Registration and Discovery |
| Ribbon | Dynamic Routing and Load Balancer |
| Hystrix | Circuit Breaker |
| Zuul | Edge Server |

In this post we develop 2 spring boot microservices named as employee-producer and employee-consumer.  
As the name suggests employee-producer will be exposing REST APIs which will be consumed by the employee-consumer.  
  


# Step1

* Develop employee-producer as follows- The maven project we will be creating is as follows-  
    
    
  The pom.xml with spring batch dependencies is 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.javainuse</groupId>
* <artifactId>employee-producer</artifactId>
* <version>0.0.1-SNAPSHOT</version>
* <packaging>jar</packaging>
* <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.boot</groupId>
* <artifactId>spring-boot-starter-web</artifactId>
* </dependency>
* <dependency>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-starter-test</artifactId>
* <scope>test</scope>
* </dependency>
* </dependencies>
* <build>
* <plugins>
* <plugin>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-maven-plugin</artifactId>
* </plugin>
* </plugins>
* </build>
* </project>

Define the domain class Employee

package com.javainuse.model;

public class Employee {

private String empId;

private String name;

private String designation;

private double salary;

public Employee() {

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDesignation() {

return designation;

}

public void setDesignation(String designation) {

this.designation = designation;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

public String getEmpId() {

return empId;

}

public void setEmpId(String empId) {

this.empId = empId;

}

}

Expose the service using the Controller as-

package com.javainuse.controllers;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RestController;

import com.javainuse.model.Employee;

@RestController

public class TestController {

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

public Employee firstPage() {

Employee emp = new Employee();

emp.setName("emp1");

emp.setDesignation("manager");

emp.setEmpId("1");

emp.setSalary(3000);

return emp;

}

}

Finally define the Spring Boot Main class

package com.javainuse;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringBootHelloWorldApplication {

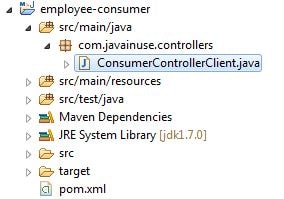
public static void main(String[] args) {

SpringApplication.run(SpringBootHelloWorldApplication.class, args);

}

}

Compile and the run the SpringBootHelloWorldApplication.java as a Java application.  
Go to **localhost:8080/employee**   
 

 Develop employee-consumer as follows- The maven project we will be creating is as follows-  
  


The pom.xml with spring batch dependencies is 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.javainuse</groupId>

<artifactId>employee-consumer</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<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.boot</groupId>

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

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

Define the controller to consume the service exposed by employee-producer above using the RESTTemplate as follows-

package com.javainuse.controllers;

import java.io.IOException;

import org.springframework.http.HttpEntity;

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpMethod;

import org.springframework.http.MediaType;

import org.springframework.http.ResponseEntity;

import org.springframework.web.client.RestClientException;

import org.springframework.web.client.RestTemplate;

public class ConsumerControllerClient {

public void getEmployee() throws RestClientException, IOException {

String baseUrl = "http://localhost:8080/employee";

RestTemplate restTemplate = new RestTemplate();

ResponseEntity<String> response=null;

try{

response=restTemplate.exchange(baseUrl,

HttpMethod.GET, getHeaders(),String.class);

}catch (Exception ex)

{

System.out.println(ex);

}

System.out.println(response.getBody());

}

private static HttpEntity<?> getHeaders() throws IOException {

HttpHeaders headers = new HttpHeaders();

headers.set("Accept", MediaType.APPLICATION\_JSON\_VALUE);

return new HttpEntity<>(headers);

}

}

Finally create the Bean for the above controller, load it and call the getEmployee() Method.

package com.javainuse;

import java.io.IOException;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.Bean;

import org.springframework.web.client.RestClientException;

import com.javainuse.controllers.ConsumerControllerClient;

@SpringBootApplication

public class SpringBootHelloWorldApplication {

public static void main(String[] args) throws RestClientException, IOException {

ApplicationContext ctx = SpringApplication.run(

SpringBootHelloWorldApplication.class, args);

ConsumerControllerClient consumerControllerClient=ctx.getBean(ConsumerControllerClient.class);

System.out.println(consumerControllerClient);

consumerControllerClient.getEmployee();

}

@Bean

public ConsumerControllerClient consumerControllerClient()

{

return new ConsumerControllerClient();

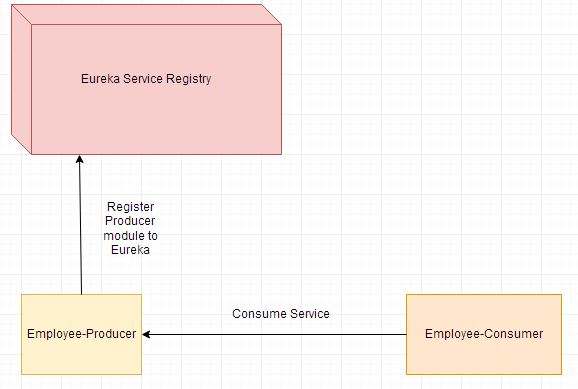
}

}

Define the application.properties as

server.port=8091

**Step2**

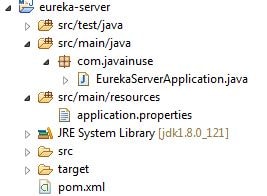
This post we do service discovery using the employee-consumer module.  
  


Video

This tutorial is explained in the below Youtube Video.

Lets Begin-

* Develop the Eureka server Service as follows-

The maven project we will be creating is as follows-  
  
  
The pom.xml with spring cloud dependencies is 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.javainuse</groupId>

<artifactId>eureka-server</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>eureka-server</name>

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

<parent>

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

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

<version>1.5.2.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-eureka-server</artifactId>**

**</dependency>**

<dependency>

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

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

<scope>test</scope>

</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>

</build>

</project>

Define the Spring Boot class with annotations @SpringBootApplication and @EnableEurekaServer.

package com.javainuse;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication

**@EnableEurekaServer**

public class EurekaServerApplication {

public static void main(String[] args) {

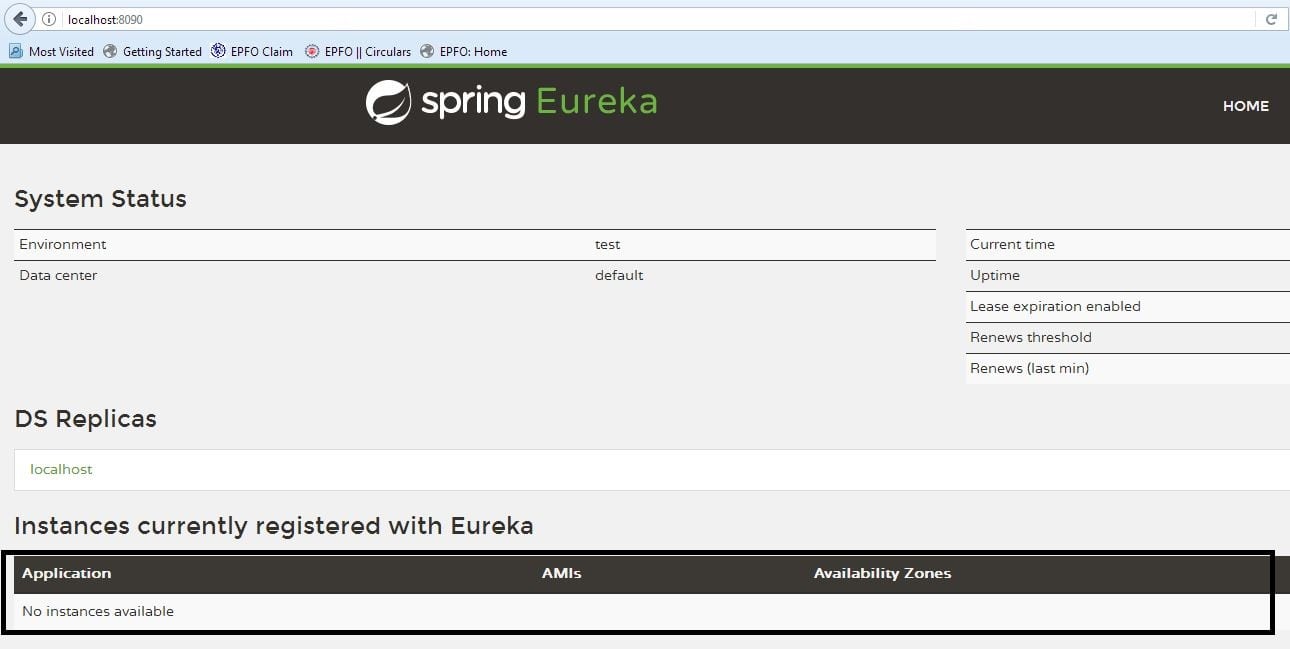
SpringApplication.run(EurekaServerApplication.class, args);

}

}

Finally specify the port on which the eureka application will get started -  
Next define the batch configuration for creating the tasklet as follows-

server.port=8090

Run this as java application. Go to URL-  
**http://localhost:8090/**  
We can see the Eureka Server page as follows-  
  


 Next we modify the [employee-producer module](https://www.javainuse.com/spring/spring_eurekaregister2) we defined previously to register to the Eureka server.

Add the Spring cloud dependencies to the pom.xml 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.javainuse</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.boot</groupId>

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

</dependency>

**<dependency>**

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

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

**</dependency>**

<dependency>

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

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

<scope>test</scope>

</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>

</build>

</project>

Next define the Spring Boot class with annotations @SpringBootApplication and @EnableDiscoveryClient.

package com.javainuse;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication

**@EnableDiscoveryClient**

public class SpringBootHelloWorldApplication {

public static void main(String[] args) {

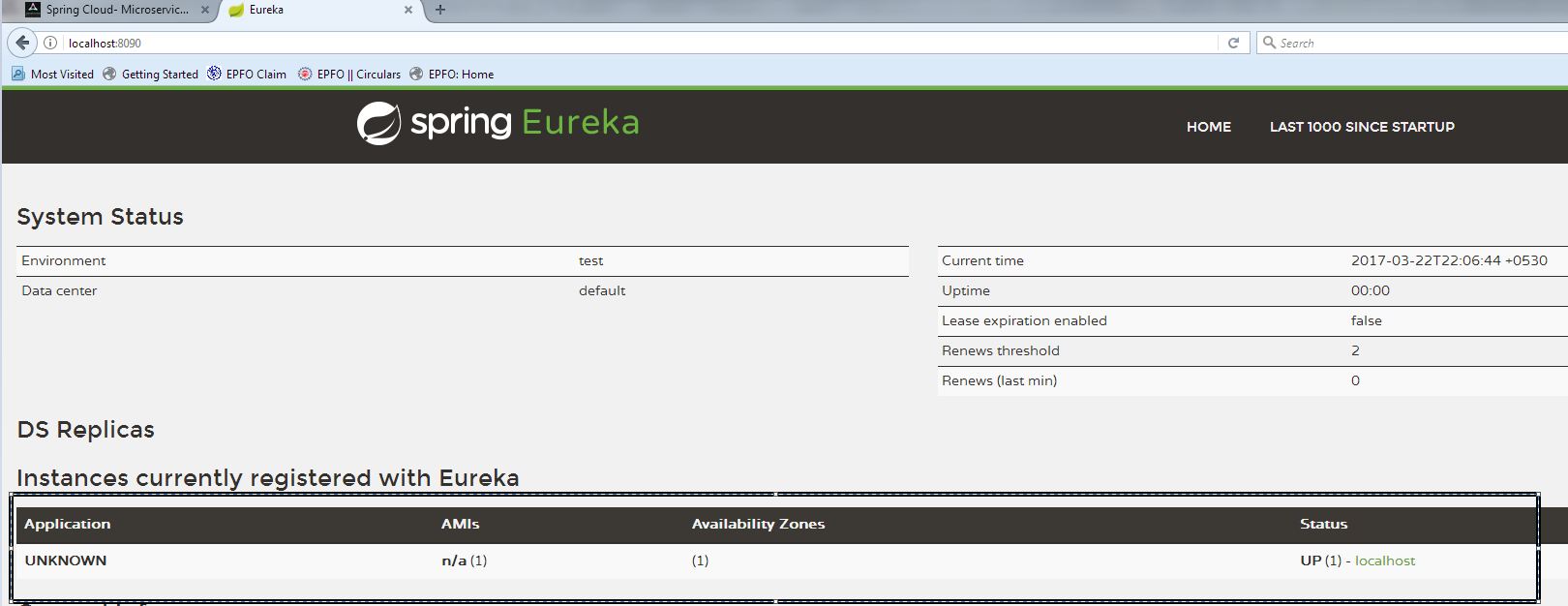
SpringApplication.run(SpringBootHelloWorldApplication.class, args);

}

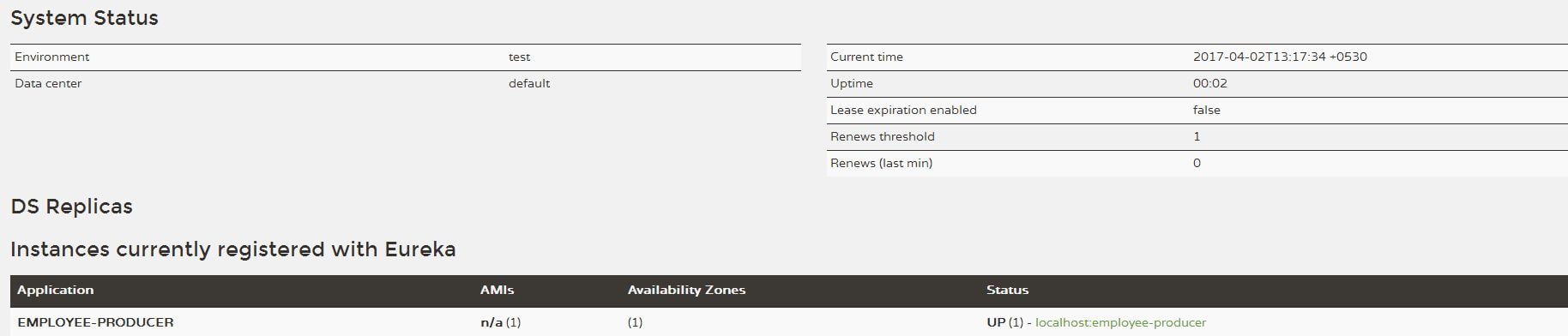
}

In the application.properties specify the url on which the Eureka server is up and running.

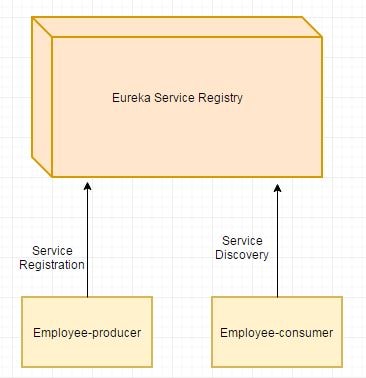
eureka.client.serviceUrl.defaultZone=http://localhost:8090/eureka

Now start this employee-producer by running this as a Java Application and go to url.  
**http://localhost:8090/**  
We can see the Eureka Server page with the employee-producer module registered as follows-  
  
  
We can see here that the registered application name is coming as UNKNOWN. To change this create in resources a file named bootstrap.properties

spring.application.name=employee-producer

We can now see the sevice registered as-  
  


This post we consume this service by discovering the employee-producer service from the Eureka server.



Lets Begin-

* Modify the employee-consumer service as follows-

Modify the pom.xml with spring cloud dependencies 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.javainuse</groupId>

<artifactId>employee-consumer</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<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.boot</groupId>

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

</dependency>

**<dependency>**

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

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

**</dependency>**

<dependency>

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

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

<scope>test</scope>

</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>

</project>

Modfy the ConsumerControllerClient class to autowire the **DiscoveryClient** dependency.

package com.javainuse.controllers;

import java.io.IOException;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.cloud.client.ServiceInstance;

import org.springframework.cloud.client.discovery.DiscoveryClient;

import org.springframework.http.HttpEntity;

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpMethod;

import org.springframework.http.MediaType;

import org.springframework.http.ResponseEntity;

import org.springframework.stereotype.Controller;

import org.springframework.web.client.RestClientException;

import org.springframework.web.client.RestTemplate;

@Controller

public class ConsumerControllerClient {

**@Autowired**

**private DiscoveryClient discoveryClient;**

public void getEmployee() throws RestClientException, IOException {

**List<ServiceInstance> instances=discoveryClient.getInstances("employee-producer");**

**ServiceInstance serviceInstance=instances.get(0);**

**String baseUrl=serviceInstance.getUri().toString();**

baseUrl=baseUrl+"/employee";

RestTemplate restTemplate = new RestTemplate();

ResponseEntity<String> response=null;

try{

response=restTemplate.exchange(baseUrl,

HttpMethod.GET, getHeaders(),String.class);

}catch (Exception ex)

{

System.out.println(ex);

}

System.out.println(response.getBody());

}

private static HttpEntity<?> getHeaders() throws IOException {

HttpHeaders headers = new HttpHeaders();

headers.set("Accept", MediaType.APPLICATION\_JSON\_VALUE);

return new HttpEntity<>(headers);

}

}

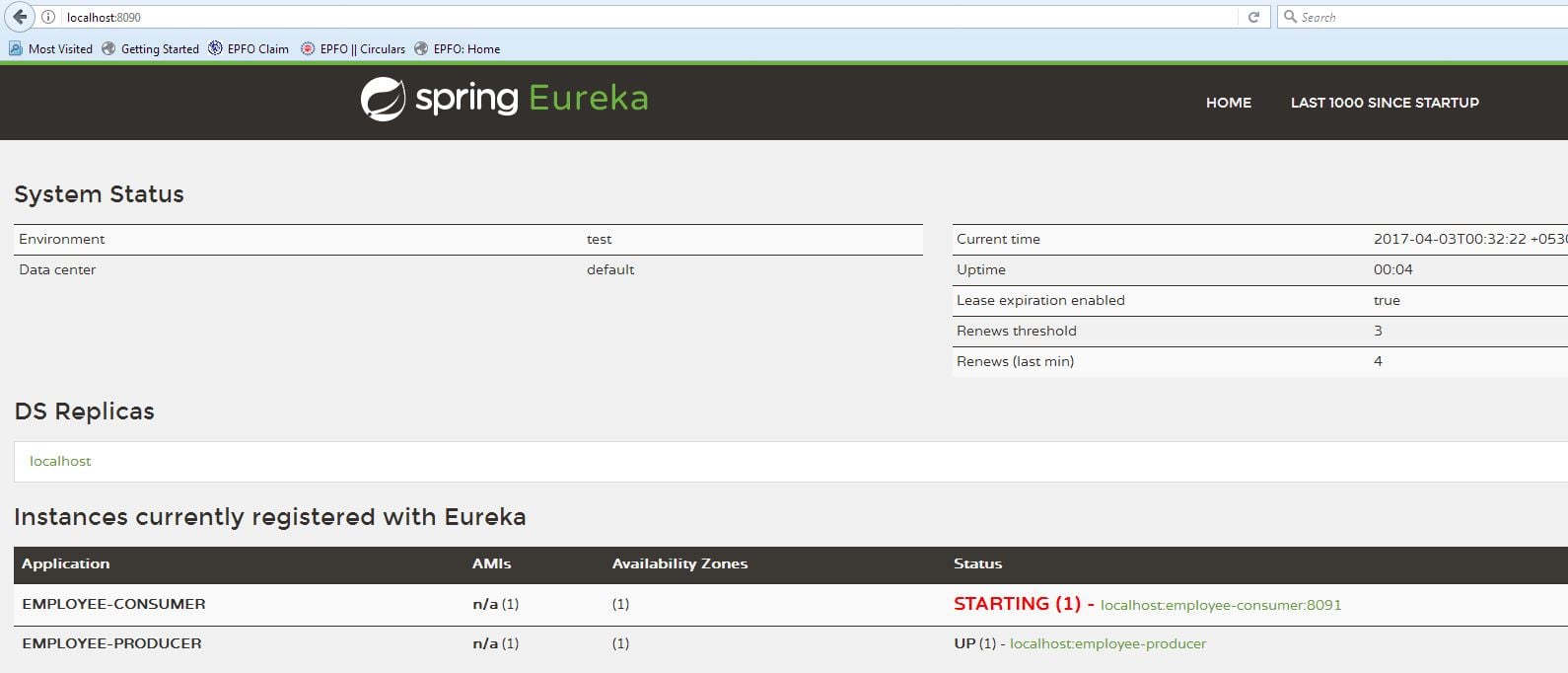
Modify the application.properties to include the eureka server url-

server.port=8091

eureka.client.serviceUrl.defaultZone=http://localhost:8090/eureka

Add the bootstrap.properties as follows-

spring.application.name=employee-consumer

Run this as java application. We can see the the employee-producer is successfully consumed-  
  
  
Go to URL-  
**http://localhost:8090/**  
We can see the Eureka Server page as follows-  
  


 So the producer and client are successfully registered with Eureka Server.