

# Services Discovery

In this tutorial we will:

1. Create a Eureka Registration Server.
2. Create a REST Producer – will register to the server in multiple instances.
3. Create a REST Consumer - will discover the available services.
4. Test it

## 1. Create a Eureka Registration Server

Create a new spring boot project using gradle.

Add the follow to the build.gradle file:

```
dependencies {
    implementation 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-server'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}

dependencyManagement {
    imports {
        mavenBom "org.springframework.cloud:spring-cloud-dependencies:${springCloudVersion}"
    }
}
```

Create SpringBootApplication. Add the annotation **@EnableEurekaServer** to allow Configuration Server:

```
@EnableEurekaServer
@SpringBootApplication
public class ServerRegistrationDemo {

    public static void main(String[] args) {
        SpringApplication.run(ServerRegistrationDemo.class, args);
    }

}
```

The content of the files **application.properties**:

**application.properties**

```
server.port=9090
```

Create and define configurations in the bootstrap.properties:

/src/main/resources/ bootstrap.properties

```
spring.application.name=demo-registration-server
```

Run the `ServerRegistrationDemo`

GoTo: <http://localhost:9090/> to enter the Eureka service server dashboard.

## 2. Create a REST Producer

Create a new spring boot project using gradle.

Add dependency to the build.gradle file:

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.boot:spring-boot-starter-actuator'
    implementation 'org.springframework.cloud:spring-cloud-starter-netflix-
eureka-client'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}

dependencyManagement {
    imports {
        mavenBom "org.springframework.cloud:spring-cloud-
dependencies:${springCloudVersion}"
    }
}
```

Create `SpringBootApplication`. Add the annotation `@EnableDiscoveryClient` to allow Registration to the Eureka Server:

```
@EnableDiscoveryClient
@SpringBootApplication
public class HelloProducerApplication{

    public static void main(String[] args) {
        SpringApplication.run(HelloProducerApplication.class, args);
    }

}
```

Create a RestController:

```
@RestController
public class HelloController {

    @RequestMapping(path="/hello",
                    method=RequestMethod.GET)
    public String sayHallo() {
        return "demo.registration.hello.producer says Hello!";
    }
}
```

Define configurations in the application.properties:

**/src/main/resources/application.properties**

```
eureka.client.serviceUrl.defaultZone=http://localhost:9090/eureka

#for random port
server.port=0

#for registration identification
eureka.instance.instance-
id=${spring.application.name}:${spring.application.instance_id:${random.val
ue}}

#for indicating the app is up
eureka.client.healthcheck.enabled=true
```

Create and define configurations in the bootstrap.properties:

**/src/main/resources/ bootstrap.properties**

```
spring.application.name=demo-registration-hello-producer
```

**\*\* INFO:** We define these configurations in the bootstrap.properties since we need them before the Application instanced.

Run the **HelloProducerApplication** in several instances.

GoTo: <http://localhost:9090/> to see the up instances

The screenshot shows the Spring Eureka dashboard. At the top, there's a header with the Spring Eureka logo and navigation links for HOME and LAST 1000 SINCE STARTUP. Below the header, the 'System Status' section displays two tables. The left table shows Environment (test) and Data center (default). The right table shows Current time (2019-05-29T20:29:23 +0300), Uptime (00:04), Lease expiration enabled (true), Renewal threshold (1), and Renewal (last min) (5). The 'DS Replicas' section shows a single replica at localhost. The 'Instances currently registered with Eureka' section displays a table with columns for Application, AMIs, Availability Zones, and Status. It shows one instance of DEMO-REGISTRATION-HELLO-PRODUCER with 2 AMIs and 2 Availability Zones, in an UP state. The 'General Info' section at the bottom has a table with columns for Name and Value.

Application	AMIs	Availability Zones	Status
DEMO-REGISTRATION-HELLO-PRODUCER	n/a (2)	(2)	UP (2) - demo-registration-hello-producer:cdca01e8facd2886d513901b249d70ca , demo-registration-hello-producer:197a63217989114bb61e718c51b0eaa3

### 3. Create a REST Consumer

Create a new spring boot project using gradle.

Add dependency to the build.gradle file:

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.boot:spring-boot-starter-actuator'
    implementation 'org.springframework.cloud:spring-cloud-starter-netflix-
eureka-client'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}

dependencyManagement {
    imports {
        mavenBom "org.springframework.cloud:spring-cloud-
dependencies:${springCloudVersion}"
    }
}
```

Create SpringBootApplication. Add the annotation **@EnableDiscoveryClient** to allow Registration to the Eureka Server:

```
@EnableDiscoveryClient
@SpringBootApplication
public class HelloConsumerApplication {

    public static void main(String[] args) {
        SpringApplication.run(HelloConsumerApplication.class, args);
    }

}
```

Create a RestController:

```
@RestController
public class ConsumerController {

    @Autowired
    DiscoveryClient discoveryClient;

    @Autowired
    EurekaClient discoveryClientEureka;

    @RequestMapping(path="/consume",
        method=RequestMethod.GET)
    public String consume() {
        InstanceInfo serviceInstance
            = discoveryClientEureka
                .getNextServerFromEureka("demo-registration-hello-
                    producer", false);
        String baseUrl = serviceInstance.getHomePageUrl();
        RestTemplate restTemplate = new RestTemplate();

        String responseString = "";

        try{
            ResponseEntity<String> response =
                restTemplate.getForEntity(baseUrl + "/hello",
                    String.class);
            if (response.getStatusCode().is2xxSuccessful())
                responseString = response.getBody();
            else
                throw new HttpClientErrorException
                    (HttpStatus.INTERNAL_SERVER_ERROR);
        }catch (Exception ex)
        {
            System.out.println(ex);
        }

        return "The producer on address: " + baseUrl + " produced the
            message: " + responseString;
    }

    @RequestMapping(path="/services",
        method=RequestMethod.GET)
    public String[] getServices() {
        return discoveryClient.getServices().toArray(new String[0]);
    }
}
```

Define configurations in the application.properties:

#### **/src/main/resources/application.properties**

```
eureka.client.serviceUrl.defaultZone=http://localhost:9090/eureka

#We set hardcoded port for easy testing...
server.port=9095

#for registration identification
eureka.instance.instance-
id=${spring.application.name}:${spring.application.instance_id:${random.val
ue}}

#for indicating the app is up
eureka.client.healthcheck.enabled=true
```

Create and define configurations in the bootstrap.properties:

#### **/src/main/resources/ bootstrap.properties**

```
spring.application.name=demo-registration-hello-consumer
```

**\*\* INFO:** We define these configurations in the bootstrap.properties since we need them before the Application instanced.

Run the **HelloConsumerApplication**.

## **4. Test it**

Make multiple GET requests to: <http://localhost:9095/consume>

Then we get the response, each time from different instance:

---

The producer on address: <http://Naor-PC.mynet:62828/> produced the message: demo.registration.hello.producer says Hello!

Make a GET request: <http://localhost:9091/services>

Then we get the response with the registered applications:

---

```
["demo-registration-hello-producer","demo-registration-hello-consumer"]
```

**\*\* Credit:** [https://www.javainuse.com/spring/spring\\_eurekaregister](https://www.javainuse.com/spring/spring_eurekaregister)