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:

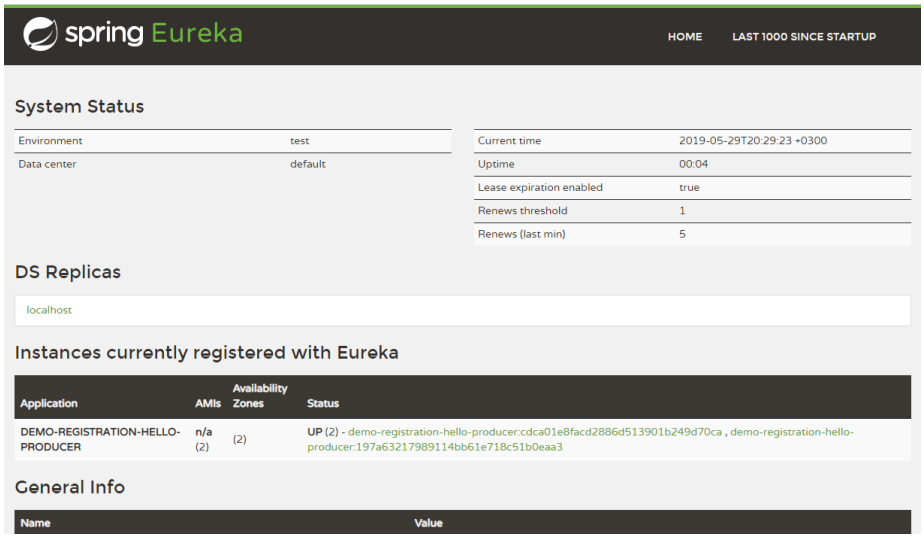
/src/main/resources/ bootstrap.properties

```
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



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

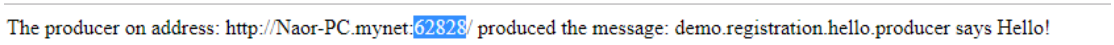
```
spring.application.name=demo-registration-hello-consumer
```

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

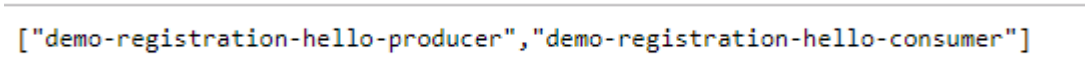
Run the **`HelloConsumerApplication.`**

4. Test it

Make multiple GET requests to: http://localhost:9095/consume
Then we get the response, each time from different instance:



Make a GET request: http://localhost:9091/services
Then we get the response with the registered applications:



** **Credit:** https://www.javainuse.com/spring/spring_eurekaregister