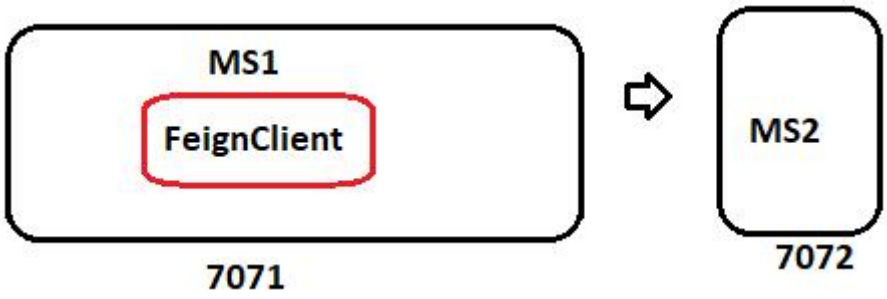
Service communication Using Feign Client

- In earlier examples, we have used *RestTemplate* to call rest endpoint. We have another Rest Client which is Feign Client.
- FeignClient also known as **Spring Cloud OpenFeign**.
- It is Declarative REST Client in Spring Boot Web Application. which means we need to specify the client specification as an Interface and Spring Boot will take care of the implementation for us.
- Writing web services with the help of FeignClient is very easier.
- FeignClient is mostly used to consume REST API endpoints which are exposed by third-party or microservice.



Steps

1. Add below dependency in pom.xml file

```
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-openfeign</artifactId>
<version>4.1.0</version>
</dependency>
```

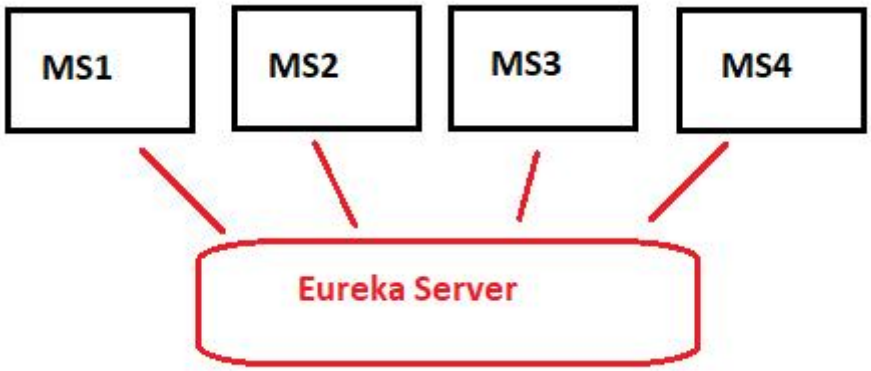
2. Create proxy interface
    a. Create interface
    b. Add **@FeignClient** annotation on top of interface
    c. write a method whose signature and req mapping is same as a controller of calling service
    d. Add **@EnalbeFeignClient** on top of starter class of caller service

Now while calling another service using rest template, use proxy interface. Autowire that interface in services layer and use it wherever is needed.

Note: *We can call those many instances of services which are mentioned in url of @FeignClient. To dynamically call any number of instances/service and scale them scale down is not possible.*

When you have multiple services running together within an application, they need to detect each other to communicate.

In order to make it possible, there should exist one medium where all microservices register themselves. Subsequently, when any service wants to communicate, it can connect to that medium and discovers other service to communicate. This medium is nothing but 'Service Registry & Discovery' which is itself a microservice.



Service registry and Service Discovery using Eureka Server

- Here we will register all microservices to common place so hat they can get each others service Id and port number to communicate.
- If any service wants to call any other microservice, they will get service Id and port number of that and call them.

To implement this, we need to create one application which behaves as eureka server where all services are registered and we have to register rest of the microservices.