



## CSC12108 – ỨNG DỤNG PHÂN TÁN

### HƯỚNG DẪN THỰC HÀNH

#### MICROSERVICE CƠ BẢN

#### I. Thông tin chung

Mã số:	HD01
Thời lượng dự kiến:	3 tiếng
Deadline nộp bài:	-
Hình thức:	-
Hình thức nộp bài:	-
GV phụ trách:	Phạm Minh Tú
Thông tin liên lạc với GV:	pmtu@fit.hcmus.edu.vn

#### II. Chuẩn đầu ra cần đạt

Bài hướng dẫn này nhằm mục tiêu đạt giúp sinh viên được các mục tiêu sau:

- Sử dụng Spring Framework
- Xây dựng **Discovery Service – Eureka Server – Spring Cloud**
- Xây dựng một service cơ bản dùng **Spring Rest**
- Xây dựng một Web Client – **Spring MVC**
- Sử dụng **Spring Boot** để build-triển khai ứng dụng

#### III. Mô tả

##### a. Môi trường phát triển

Trong bài này, các công cụ và môi trường dưới đây sẽ được dùng để hướng dẫn sinh viên thực hành môn học.

- **Eclipse IDE for Enterprise Java Developers.**
  - Version: 2019-03 (4.11.0)
  - Build id: 20190314-1200
- **Java JDK 1.8**

##### b. Cài đặt Plugin cần thiết

Để dùng **Spring** một cách thuận lợi, một plugin cần cài đặt như các bước sau đây:

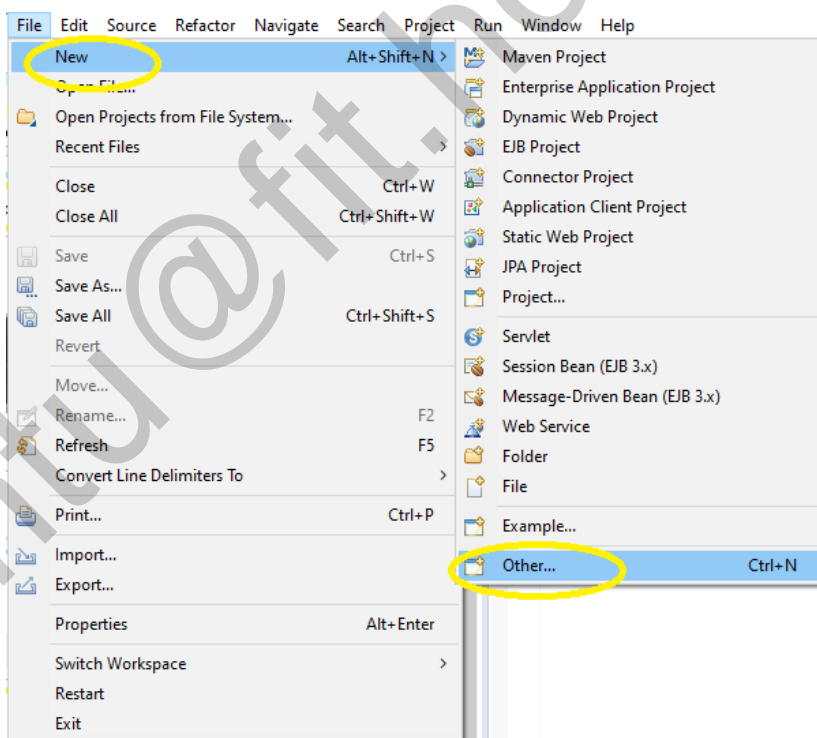
Bước 1: Mở eclipse -> Vào Window -> Chọn Eclipse Marketplace

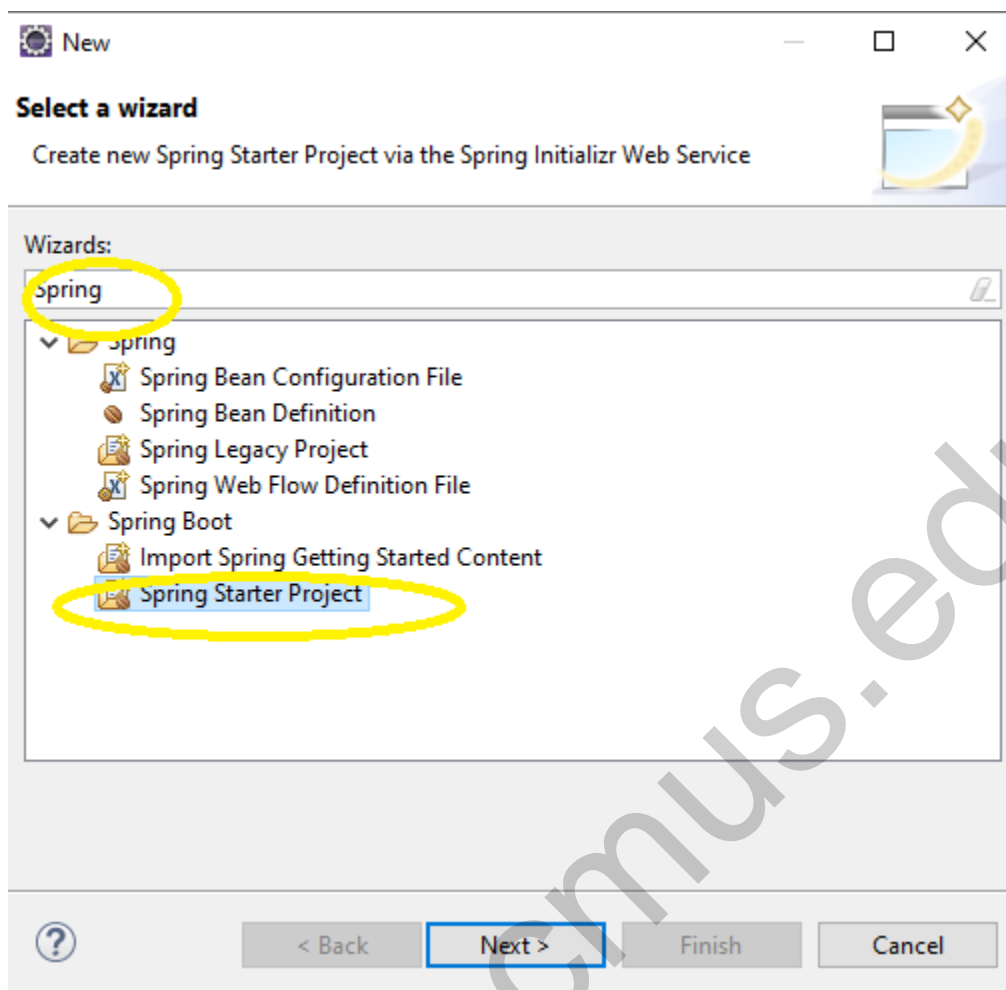
Bước 2: Tìm kiếm “Spring Boot” và tìm kiếm



**Bước 3:** Tiến hành cài đặt theo hướng dẫn eclipse

**Bước 4:** Khởi động lại eclipse, kiểm tra bằng cách tạo Project như sau:





Quá trình cài đặt plugin đã kết thúc thành công.

### c. Giới thiệu Spring Framework

Spring là một Framework phát triển các ứng dụng Java giúp tạo các ứng dụng có hiệu năng cao, dễ kiểm thử, sử dụng lại code...

Spring nhẹ và trong suốt (nhẹ: kích thước nhỏ, version cơ bản chỉ khoảng 2MB; trong suốt: hoạt động một cách trong suốt với lập trình viên)

Spring là một mã nguồn mở, được phát triển, chia sẻ và có cộng đồng người dùng rất lớn.

Spring Framework được xây dựng dựa trên 2 nguyên tắc design chính là: Dependency Injection và Aspect Oriented Programming.

Những tính năng core (cốt lõi) của Spring có thể được sử dụng để phát triển Java Desktop, ứng dụng mobile, Java Web. Mục tiêu chính của Spring là giúp phát triển các ứng dụng J2EE một cách dễ dàng hơn dựa trên mô hình sử dụng POJO (Plain Old Java Object)

### d. Giới thiệu Eureka Server

*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)..*

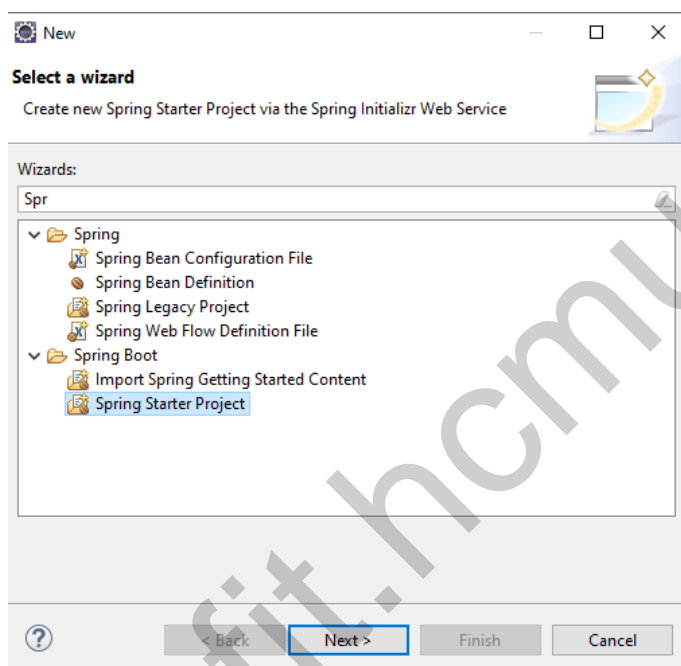
**Eureka Server** là nơi đăng kí của các **service**, các service có thể tìm thông tin của nhau thông qua nơi đăng kí này và giao tiếp.

Đây là mã nguồn mở của **Netflix**, ta có thể dùng Spring Cloud để cài đặt một Eureka Server một cách dễ dàng.

Các bước làm như sau:

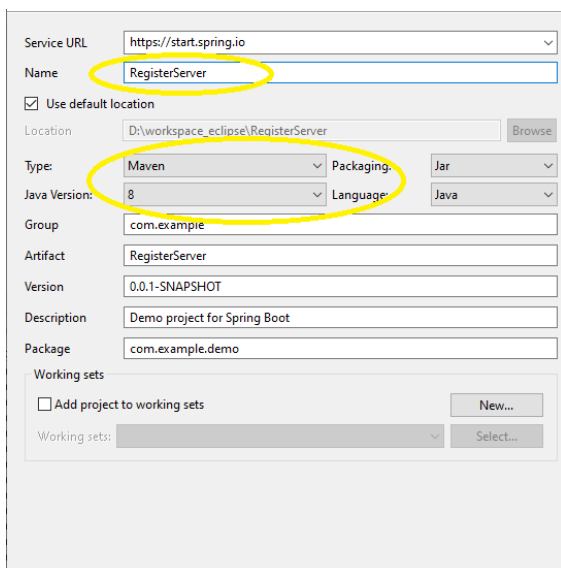
**Bước 1:** Tạo một Project Spring Boot

File -> New -> Other



Nhấn Next

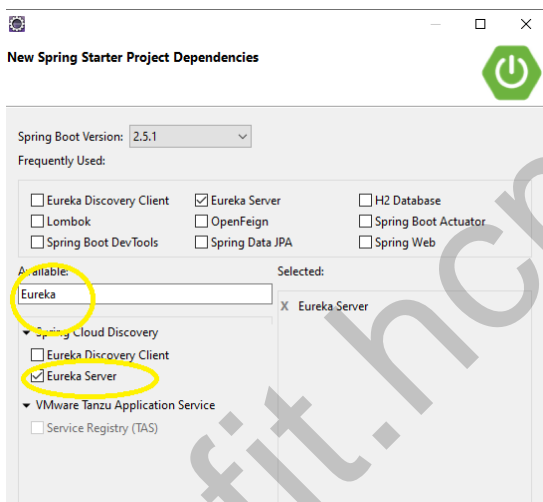
**Bước 2:** Đặt tên Project là RegisterServer



Project dùng maven để build và quản lý các phụ thuộc.

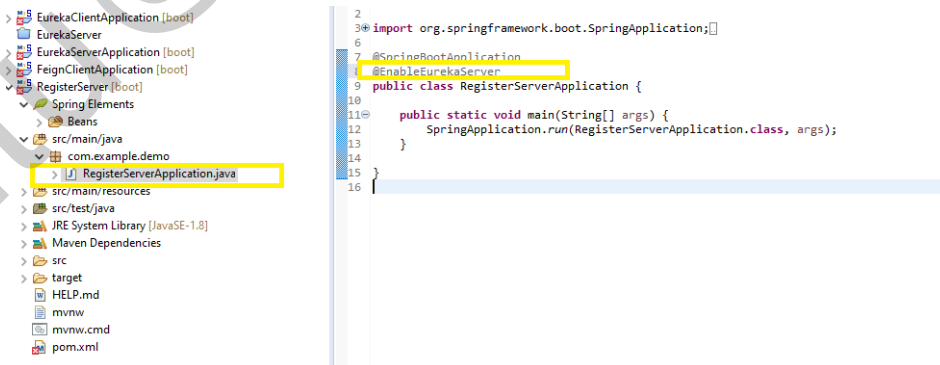
Tìm kiếm Eureka Server và chọn.

Nhấn Next



Nhấn Next và Finish.

**Bước 3:** Dùng Annotation `@EnableEurekaServer` để khai báo ứng dụng trên đóng vai trò Server, là nơi để các service có thể đăng kí và tìm kiếm bởi các service khác.



Để hoàn thành cài đặt server, hãy cấu hình vài thông tin trong file **application.properties**

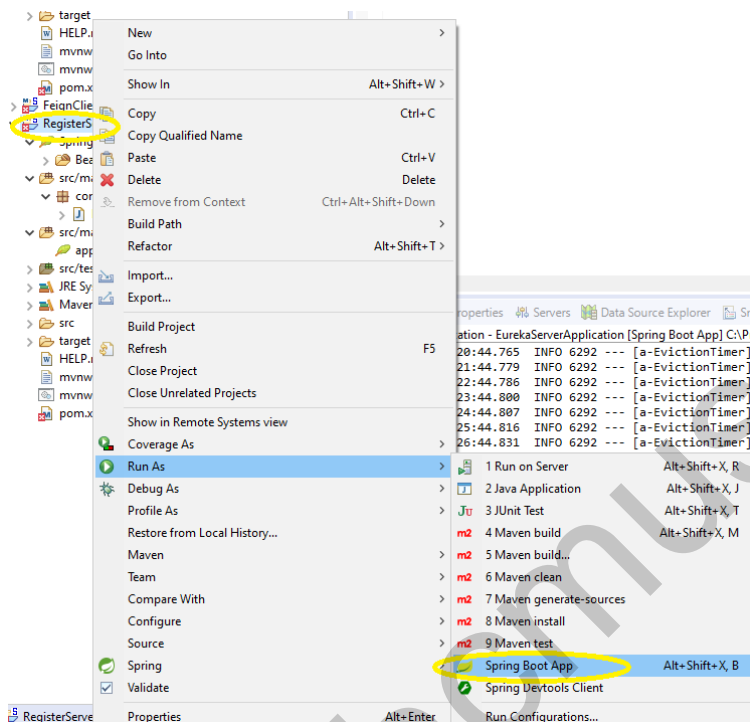
1. `server.port=8761`
2. `eureka.client.register-with-eureka=false`
3. `eureka.client.fetch-registry=false`

Dòng 1: định nghĩa port của server, nơi lắng nghe các yêu cầu đăng kí của các service  
Bởi vì Eureka Server cũng có thể đóng vai trò như một Client. Nên chúng ta cần cấu hình theo nhu cầu như sau:

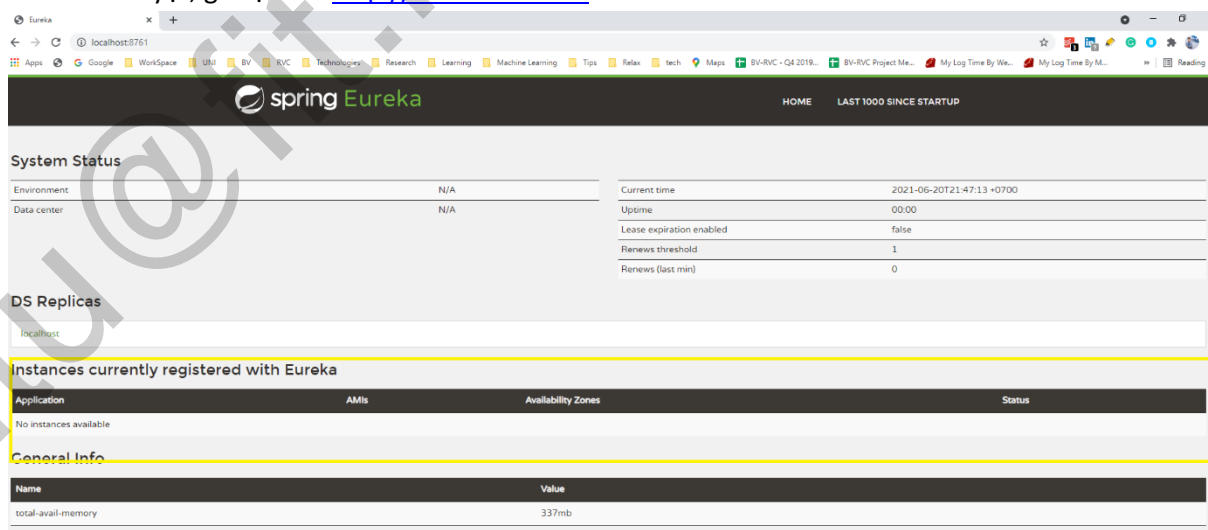
Dòng 2: Đóng vai trò là server, vì vậy không cần đăng kí dịch vụ với chính nó.

Dòng 3: Đóng vai trò là server, không cần tìm kiếm các service khác

**Bước 4:** Build và triển khai Eureka Server dùng Spring Boot



Mở trình duyệt, gõ địa chỉ: <http://localhost:8761>

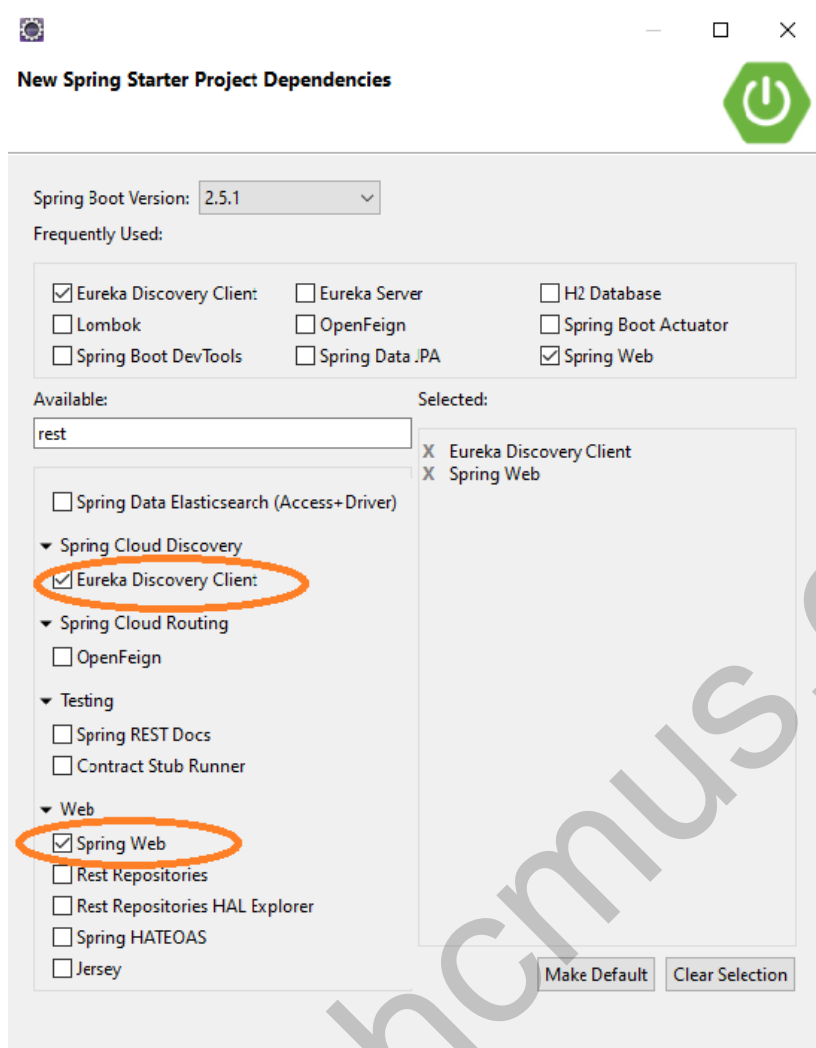


**Kết quả:** Không có bất kì instance nào có service đăng kí. Phần tiếp theo sẽ hướng dẫn tạo các service và đăng kí với Eureka server.

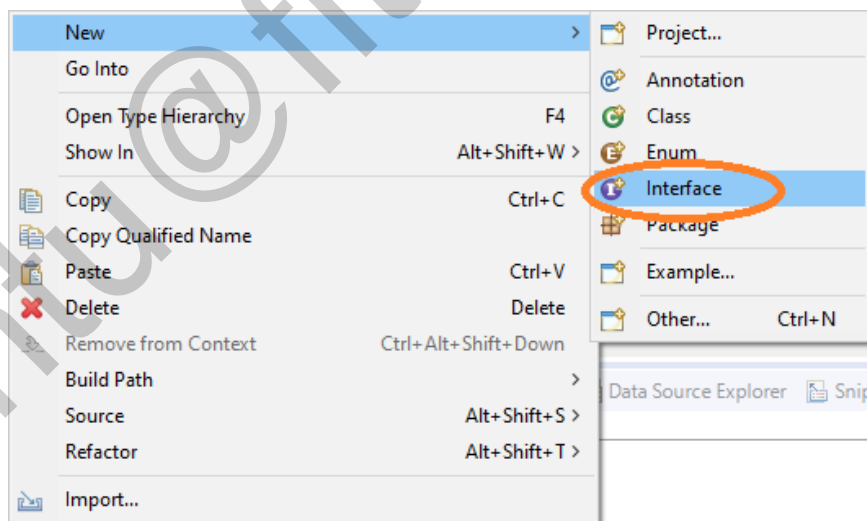
#### e. Tạo service Quản lý nhân sự

Tạo Project tương tự như trên và đặt tên Quản lý nhân sự

Lưu ý: Chọn Eureka Client để đăng kí service đến server và Spring Web



Service có các xử lý nào? Tạo một interface để định nghĩa các hành động sau:





```
package com.example.demo;

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

public interface QuanLyNhanVien {

    @GetMapping(value = "/employee")
    String LayThongTinNhanVien();

    @RequestMapping(value = "/add")
    void ThemNhanVien();

}
```

Tạo một RestController cho phép người dùng truy xuất các thông tin liên quan đến nhân sự của một công ty.

Đây được xem như là một service cung cấp các thông tin liên quan đến nhân sự. Sử dụng RestController của thư viện Spring Web để tạo các service cần thiết.

```
@SpringBootApplication
@RestController
public class QuanLyNhanSuApplication implements QuanLyNhanVien{

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

    @Override
    public String LayThongTinNhanVien() {
        // TODO Auto-generated method stub
        String str = "This is emplolyee information";
        return str;
    }

    @Override
    public void ThemNhanVien() {
        // TODO Open connection
        // Insert database
        // Close connection
        // Done
    }

}
```

Sau khi xây dựng xong service, cần phải đăng kí service đến Server – Eureka Server.

Để thực hiện điều này ta chỉ cần cấu hình thông tin đơn giản như sau:

Vào application.properties và khai báo các thông tin như bên dưới.

1. spring.application.name=spring-cloud-eureka-client
2. server.port=8081



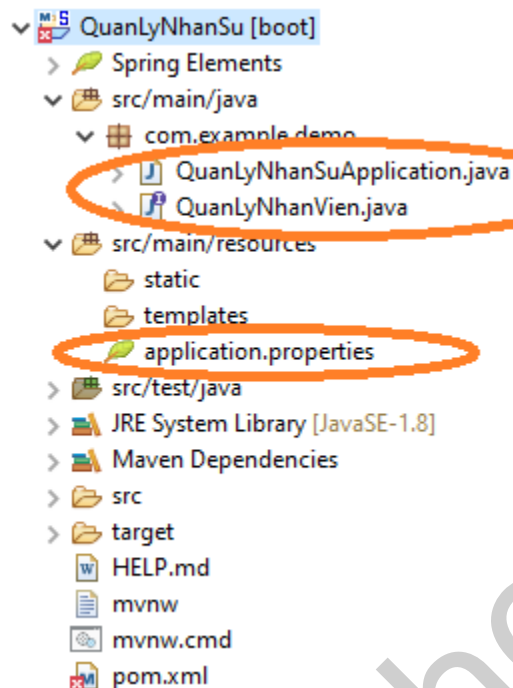
3. `eureka.client.serviceUrl.defaultZone = http://localhost:8761/eureka`

Dòng 1: Tên của service

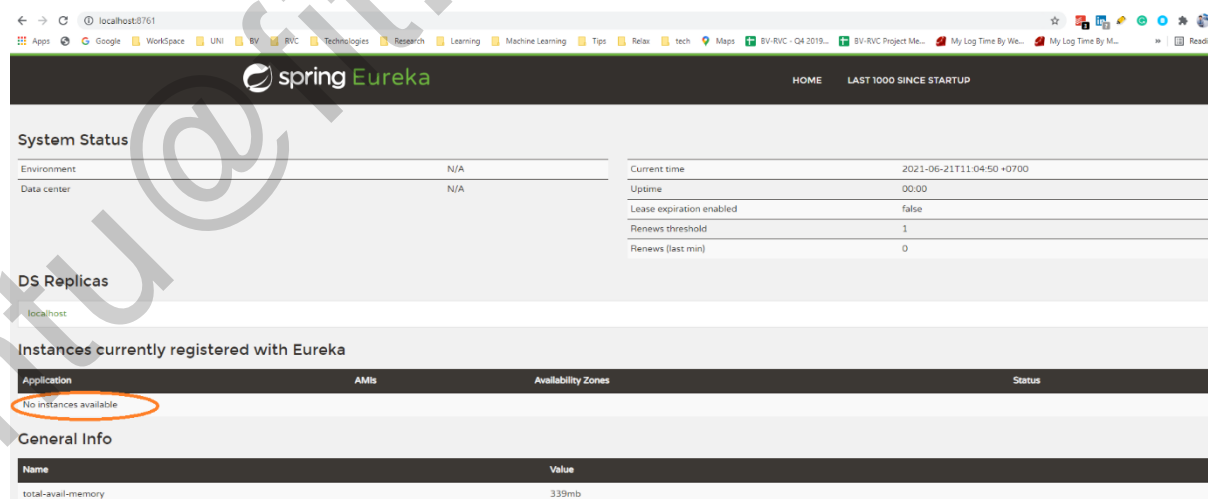
Dòng 2: port của service

Dòng 3: Đăng kí service tại server được xác định bởi địa chỉ như trên.

Tổng quan các files trong Project.

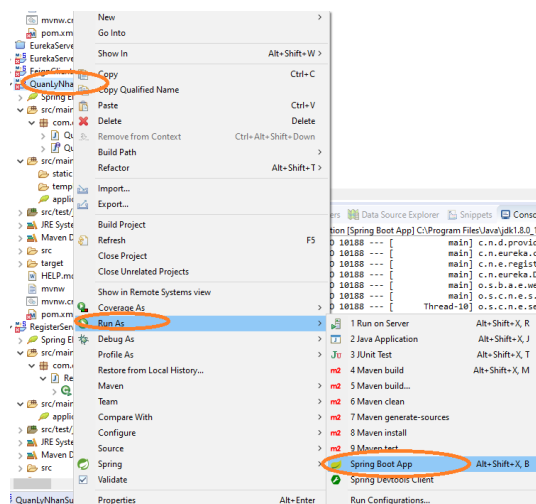


Để kiểm tra hãy triển khai server Eureka Server (RegisterServer) trước và xem kết quả như sau:

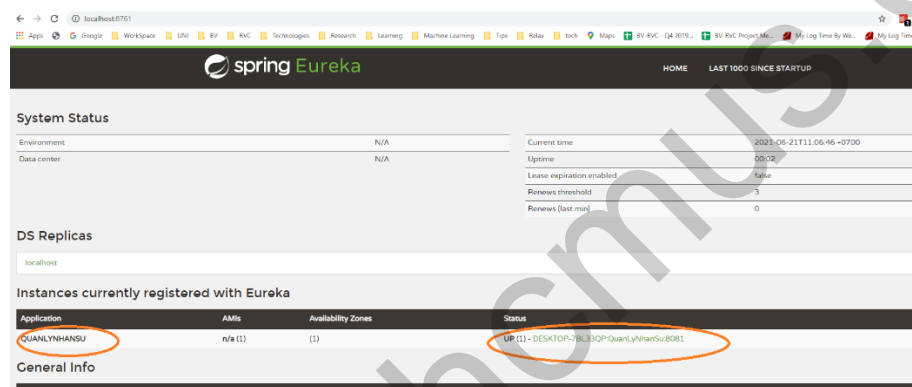


Như thông tin trên, chưa có bất kì service nào được đăng kí.

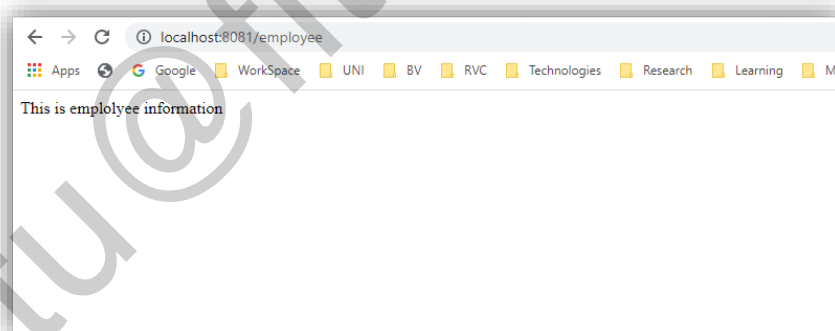
Tiếp theo triển khai service QuanLyNhanSu để tiến hành đăng kí trên ServerRegister



Kiểm tra lại (F5 hoặc refresh) ServerRegister, ta thấy service đã được đăng kí trên Server với thông tin như bên dưới.



Có thể kiểm tra service chạy trên trình duyệt: <http://localhost:8081/employee>



Thông thường, service này có thể được truy xuất bởi một client như Web App, Mobile App,...

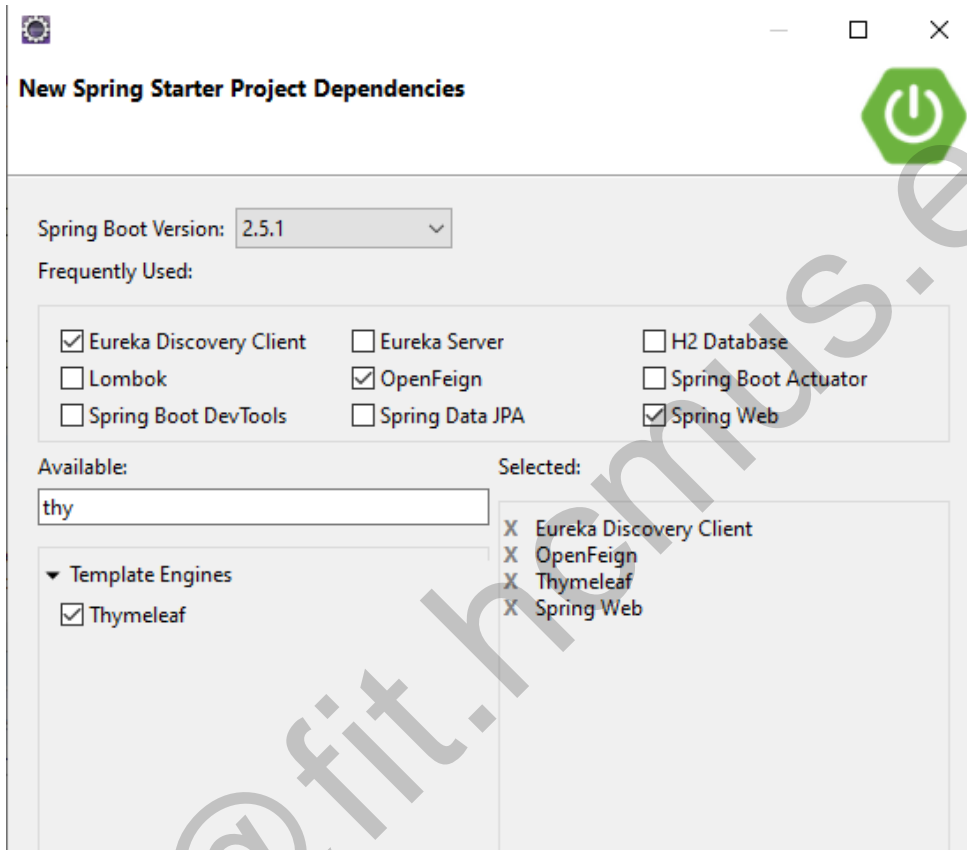
Bằng cách kết nối tới Url cố định như <http://localhost:8081/employee>. Tuy nhiên điều này không hay bởi Url có thể thay đổi theo thời gian để mở rộng hoặc thay đổi môi trường.

Vì vậy một client muốn truy xuất đến dịch vụ này có thể khám phá (discovery) trên Server (Eureka Server – Server Register App) dựa vào tên của service. Chúng ta sẽ thực hành phần này trong phần tiếp theo.

#### f. Tạo ứng dụng client - ứng dụng web

Tạo một Project đặt tên UDPT\_17CLC, chọn các thư viện như sau:

- Eureka Discovery Client
- OpenFeign (Dùng để gọi service một cách dễ dàng, không cần viết code nhiều cho phần gọi services từ client)
- Spring Web
- Thymeleaf (một Java template engine dùng để xử lý và tạo HTML, XML, Javascript, CSS và text – không cần dùng JSP)



Sau khi tạo xong Project, hãy tạo một interface để gọi service QuanLyNhanSu, interface tương tự như Service QuanLyNhanSu

```
package com.example.demo;

import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;

@FeignClient("QuanLyNhanSu")
public interface QuanLyNhanSuClient {

    @GetMapping(value = "/employee")
    String LayThongTinNhanVien();

    @RequestMapping(value = "/add")
    void ThemNhanVien();
}
```

}

@FeignClient("QuanLyNhanSu"): Gọi service QuanLyNhanSu thông qua tên service chứ không phải là một Url cố định.

Bước tiếp theo chúng ta tạo Controller để xử lý các request cho ứng dụng client, ứng dụng dùng mẫu MVC, sử dụng thư viện Spring MVC

```
package com.example.demo;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.openfeign.EnableFeignClients;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

@SpringBootApplication
@EnableFeignClients
@Controller
public class Udpt17ClcApplication{

    public static void main(String[] args) {
        SpringApplication.run(Udpt17ClcApplication.class, args);
    }

    @Autowired
    QuanLyNhanSuClient client;

    @RequestMapping(value = "/employee")
    public ModelAndView LayThongTinNhanVien() {
        String info = client.LayThongTinNhanVien();
        return new ModelAndView("nhanvien.html", "info", info);
    }

}
```

Giải thích:

@EnableFeignClients

Cho phép ứng dụng tìm các interface là @FeignClient

@Controller

Tất cả request gửi đến controller qua url /employee, sẽ được xử lý bởi phương thức

LayThongTinNhanVien() và kết quả trả về là một View ("nhanvien.html") kèm model là giá trị trả về từ service QuanLyNhanSu

Tiếp tục tạo View nhanvien.html như sau:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>Welcome to my client</h1>
<h2 th:text = "${info}" />
</body>
</html>
```

UDPT\_17CLC [boot]

- > Spring Elements
- > src/main/java
  - > com.example.demo
    - > QuanLyNhanSuClient.java
    - > Udp17ClcApplication.java
      - > Udp17ClcApplication
- > src/main/resources
  - > static
  - > templates
    - > nhanvien.html
    - > application.properties
- > src/test/java
- > JRE System Library [JavaSE-1.8]
- > Maven Dependencies
- > src
- > target
- > HELP.md
- > mvnw
- > mvnw.cmd
- > pom.xml

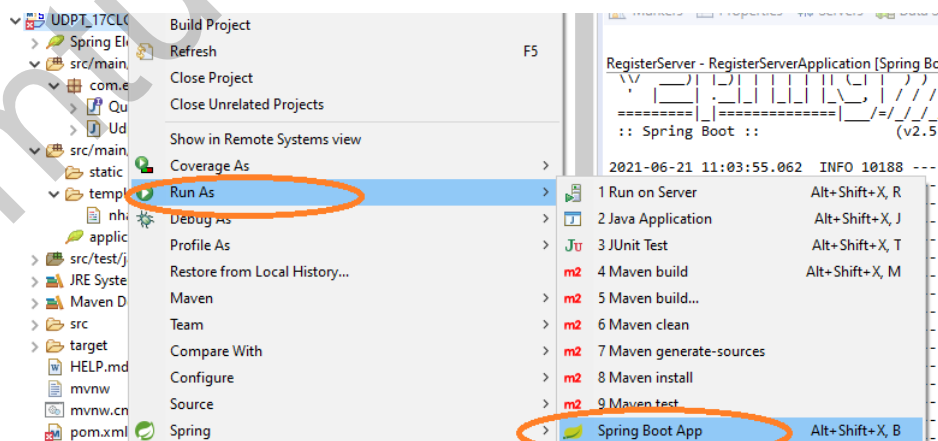
Cuối cùng, cấu hình các thông tin như tên ứng dụng client, đăng kí service trên Server và port trong file application.properties

spring.application.name= Udp17Clc

server.port=8082

eureka.client.serviceUrl.defaultZone = <http://localhost:8761/eureka>

Kiểm tra kết quả bằng cách build và chạy Spring Boot



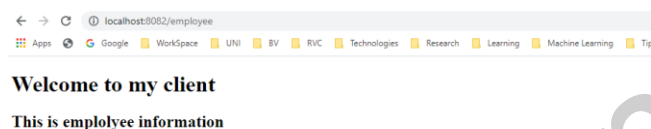
localhost			
Instances currently registered with Eureka			
Application	AMIs	Availability Zones	Status
QUANLYNHANSU	n/a (1)	(1)	UP (1) - DESKTOP-7BL33QP:QuanLyNhanSu:8081
UDPT17CLC	n/a (1)	(1)	UP (1) - DESKTOP-7BL33QP:Udpt17Clc:8082
General Info			
Name	Value		
total-avail-memory	410mb		
num-of-cpus	8		
current-memory-usage	102mb (24%)		
server-uptime	03:25		

Có 2 thể hiện:

(1) là service phục vụ chức năng quản lý nhân sự

(2) là ứng dụng web có sử dụng kết quả service (1)

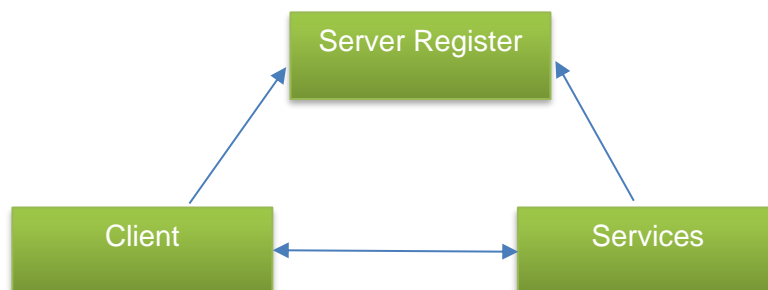
Hãy bắt đầu sử dụng ứng dụng web (2): <http://localhost:8082/employee>



### Tóm tắt:

Chúng ta đã xây dựng một ứng dụng Microservice nhỏ gồm:

1. Discovery Server
2. Service
3. Client





#### IV. Tài liệu tham khảo

[https://cloud.spring.io/spring-cloud-netflix/multi/multi\\_service\\_discovery\\_eureka\\_clients.html](https://cloud.spring.io/spring-cloud-netflix/multi/multi_service_discovery_eureka_clients.html)  
[https://cloud.spring.io/spring-cloud-netflix/multi/multi\\_spring-cloud-eureka-server.html](https://cloud.spring.io/spring-cloud-netflix/multi/multi_spring-cloud-eureka-server.html)  
<https://spring.io/projects/spring-boot>  
<https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html>  
<https://dzone.com/articles/spring-boot-microservices-building-microservices-a>

#### V. Các quy định khác