

230801 Eureka

MicroService

- 어플리케이션을 구성하는 각각의 구성요소 및 서비스의 내용을 분리해서 개발하고 운영하는 방식
- 회사의 여러 서비스들을 작은 단위로 만들어 놓음
 - 오류 났을 때 하나만 고쳐도 되게 작은 단위로 개발하는 것

- RESTful : 각 서비스들은 서로 상태에 대해 REST API로 통신해야 한다.
- Small Well Chosen Deployable Uni : 독립적으로 배포 가능한 형태의 작은 서비스로 이루어짐
- CI/CD : Continuous(지속적인) Integration(통합) / Continuous Delivery(지속적 배포)
- 서로 다른 프로그램 언어와 서로 다른 데이터 베이스를 사용할 수 있다.

▶ 이러한 서비스들은 비즈니스 기능을 중심으로 구축되어야 함

- eureka (netflix) : 디스커버리 서비스
 - 클라우드 환경이 되면서 서비스가 오토스케일링 등에 의해서 동적으로 생성되거나 컨테이너 기반의 배포로 인해서 서비스의 IP가 동적으로 변경

(+) Monolithic

- MicroService가 아닌 방식
- 일체로 되어 있는
- 어플리케이션 개발에 필요한 모든 요소를 하나의 소프트웨어에 포함시켜 개발하는 방법

- [springboot14_discoveryservice]
 - new spring starter project - Eureka Server 추가(패키지명, artifact 이름 맞추기)
 - application.properties 파일 > (오른쪽버튼) Convert properties to YAML > 작성(자동완성 사용)

```
server:
  port: 8761

spring:
  application:
    name: discoveryservice

eureka:
  client:
    register-with-eureka: false #eureka 서버에 등록할지 여부를 설정
    fetch-registry: false      #eureka 서버로부터 정보를 가져올지 여부를 설정
```

- application 파일에 @EnableEurekaServer // 유레카 서버 활성화

- www.naver.com <== IP 변환 : DNS

hosts 파일에서 이런 정보가 있는지 먼저 확인

- C:\Windows\System32\drivers\etc > hosts 메모장에서 열어보기 (관리자권한)
 - localhost 앞에 # 제거

```
# Copyright (c) 1993-2009 Microsoft Corp.
#
```

```
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#       102.54.94.97       rhino.acme.com           # source server
#       38.25.63.10       x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
127.0.0.1       localhost
::1            localhost
# Added by Docker Desktop
#192.168.0.55 host.docker.internal
#192.168.0.55 gateway.docker.internal
# To allow the same kube context to work on the host and the container:
#127.0.0.1 kubernetes.docker.internal
# End of section
```

- [springboot15_discoveryclient]
 - new spring starter project (패키지명, artifact 이름 맞추기)
 - Eureka Server, spring web, devtools, lombok 추가
 - application.properties 파일 > (오른쪽버튼) Convert properties to YAML > 작성(자동완성 사용)

```
server:
  port: 0
  #랜덤 포트를 사용

spring:
  application:
    name: discoveryclient

eureka:
  client:
    register-with-eureka: true #유레카 서버에 자신의 정보를 등록
    fetch-registry: true
    service-url:
      defaultZone: http://localhost:8761/eureka
  instance:
    instance-id: ${spring.application.name}:${spring.application.instance-id:${random.value}}
```

- application 파일에 @EnableEurekaClient // eurekaClient 활성화
- springboot14_discoveryservice 프로젝트 켜고, springboot15_discoveryclient 켜고,
localhost:8761 실행하면 아래와 같이 뜨게 됨

HOME LAST 1000 SINCE STARTUP
Google Translate

System Status

Environment	test	Current time	2023-08-01T09:52:38 +0900
Data center	default	Uptime	00:01
		Lease expiration enabled	false
		Renews threshold	3
		Renews (last min)	2

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
UNKNOWN	n/a (1)	(1)	UP (1) - host.docker.internal

General Info

Name	Value
total-avail-memory	114mb
num-of-cpus	8
current-memory-usage	45mb (39%)
server-uptime	00:01
registered-replicas	http://localhost:8761/eureka/
unavailable-replicas	http://localhost:8761/eureka/
available-replicas	

- 서버 다 끄고, 14 키고 localhost:761에서 확인하고, 15에서 두 번 실행(>세 번)

HOME LAST 1000 SINCE STARTUP

System Status

Environment	test	Current time	2023-08-01T10:47:11 +0900
Data center	default	Uptime	00:01
		Lease expiration enabled	false
		Renews threshold	6
		Renews (last min)	0

DS Replicas

localhost

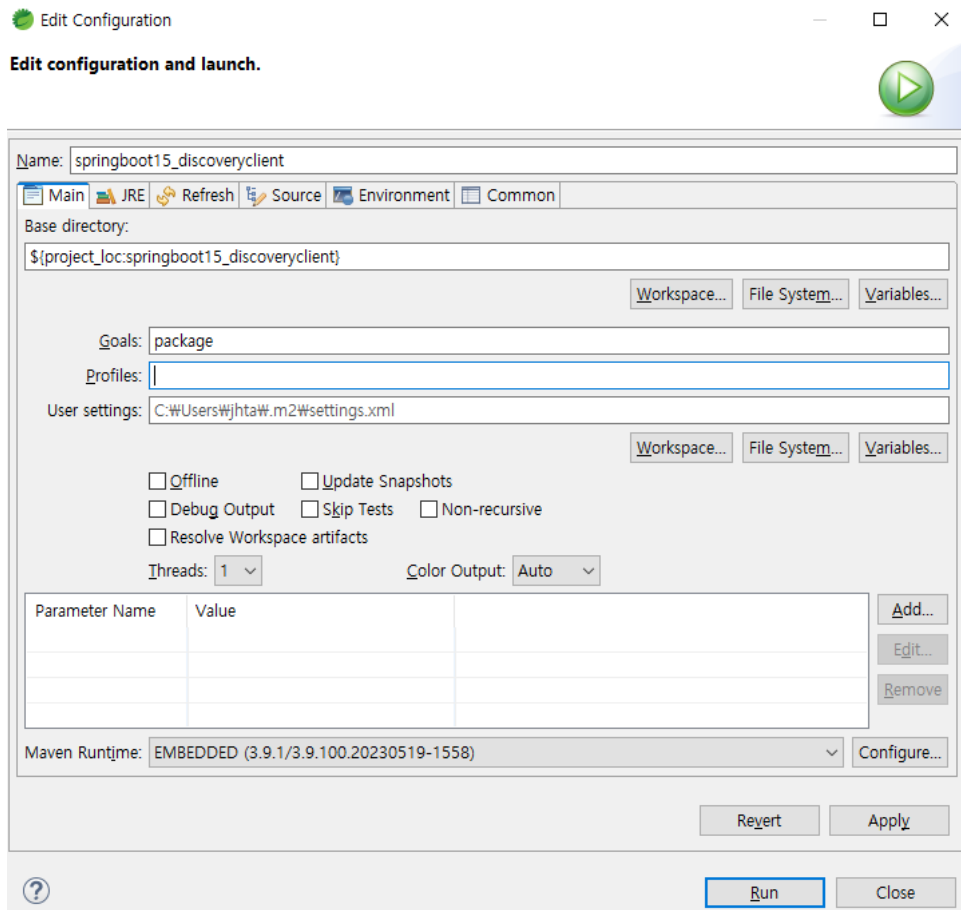
Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
DISCOVERYCLIENT	n/a (3)	(3)	UP (3) - discoveryclient:8d78e0fae80573d83fb0067317c4f4ed , discoveryclient:bdeedd1e44ba0064901b1bbe83c882a , discoveryclient:7da9e890878ed38af1f118c4e383e0f0

General Info

Name	Value
total-avail-memory	114mb
num-of-cpus	8
current-memory-usage	42mb (36%)
server-uptime	00:01
registered-replicas	http://localhost:8761/eureka/
unavailable-replicas	http://localhost:8761/eureka/
available-replicas	

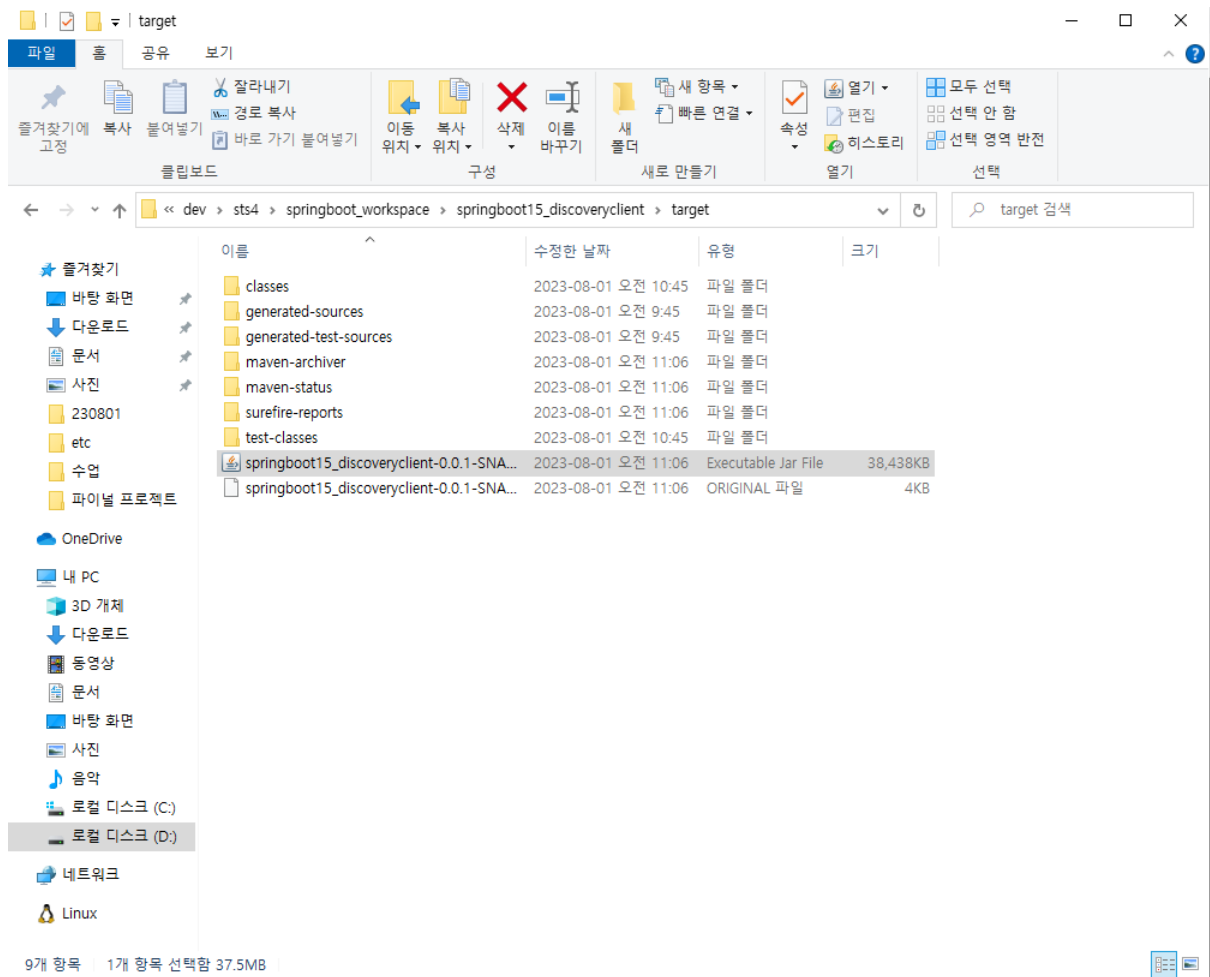
- 추출
 - 지금까지 했던 프로젝트 jar로 저장
 - 15프로젝트에서 오른쪽 버튼 > 5 maven build > run (하고 기다리기..)



- 잘 완료되면 콘솔에 아래와 같이 뜸

```
[INFO] [1m-----[m
[INFO] [1;32mBUILD SUCCESS[m
[INFO] [1m-----[m
[INFO] Total time: 16.663 s
[INFO] Finished at: 2023-08-01T11:06:37+09:00
[INFO] [1m-----[m
```

- 완료되면 아래 폴더에 jar파일로 프로젝트 저장됨(D:\dev\sts4\springboot_workspace\springboot15_discoveryclient\target)
 - —> jar 파일 복사해서 D드라이브에 붙여넣기



- 단독으로 프로그램 실행 > cmd

```

C:\Users\jhta>D:

D:\>java -jar -Dserver.port=9010 D:\springboot15_discoveryclient-0.0.1-SNAPSHOT.jar

  ____ _
 / ___ \| | | |
/ /___ \| |_| |
\___)___|_|_|_|
:: Spring Boot :: (v2.7.14)

2023-08-01 11:20:17.109 INFO 5592 --- [main] d.Springboot15DiscoveryclientApplication : Starting Springboot15Discovery
2023-08-01 11:20:17.111 INFO 5592 --- [main] d.Springboot15DiscoveryclientApplication : No active profile set, falling
2023-08-01 11:20:18.790 INFO 5592 --- [main] o.s.cloud.context.scope.GenericScope : BeanFactory id=e3a6687a-d166-3
2023-08-01 11:20:19.083 INFO 5592 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s)
2023-08-01 11:20:19.091 INFO 5592 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-08-01 11:20:19.092 INFO 5592 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apac
2023-08-01 11:20:26.899 INFO 5592 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded W
2023-08-01 11:20:26.900 INFO 5592 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: ir
2023-08-01 11:20:27.517 INFO 5592 --- [main] DiscoveryClientOptionalArgsConfiguration : Eureka HTTP Client uses RestTe
2023-08-01 11:20:27.661 WARN 5592 --- [main] igation$LoadBalancerCaffeineWarnLogger : Spring Cloud LoadBalancer is c
2023-08-01 11:20:27.686 INFO 5592 --- [main] o.s.c.n.eureka.InstanceInfoFactory : Setting initial instance statu
2023-08-01 11:20:27.801 INFO 5592 --- [main] com.netflix.discovery.DiscoveryClient : Initializing Eureka in region
2023-08-01 11:20:27.813 INFO 5592 --- [main] c.n.d.s.r.aws.ConfigClusterResolver : Resolving eureka endpoints via
2023-08-01 11:20:27.862 INFO 5592 --- [main] com.netflix.discovery.DiscoveryClient : Disable delta property : false
2023-08-01 11:20:27.862 INFO 5592 --- [main] com.netflix.discovery.DiscoveryClient : Single vip registry refresh pr
2023-08-01 11:20:27.863 INFO 5592 --- [main] com.netflix.discovery.DiscoveryClient : Force full registry fetch : fa
2023-08-01 11:20:27.863 INFO 5592 --- [main] com.netflix.discovery.DiscoveryClient : Application is null : false
2023-08-01 11:20:27.865 INFO 5592 --- [main] com.netflix.discovery.DiscoveryClient : Registered Applications size i
2023-08-01 11:20:27.866 INFO 5592 --- [main] com.netflix.discovery.DiscoveryClient : Application version is -1: tru
2023-08-01 11:20:27.867 INFO 5592 --- [main] com.netflix.discovery.DiscoveryClient : Getting all instance registry
2023-08-01 11:20:28.563 INFO 5592 --- [main] com.netflix.discovery.DiscoveryClient : The response status is 200
2023-08-01 11:20:28.566 INFO 5592 --- [main] com.netflix.discovery.DiscoveryClient : Starting heartbeat executor: r

```

```

2023-08-01 11:20:28.571 INFO 5592 --- [          main] c.n.discovery.InstanceInfoReplicator : InstanceInfoReplicator onDemand
2023-08-01 11:20:28.577 INFO 5592 --- [          main] com.netflix.discovery.DiscoveryClient : Discovery Client initialized at
2023-08-01 11:20:28.580 INFO 5592 --- [          main] o.s.c.n.e.s.EurekaServiceRegistry : Registering application DISCOV
2023-08-01 11:20:28.580 INFO 5592 --- [          main] com.netflix.discovery.DiscoveryClient : Saw local status change event
2023-08-01 11:20:28.583 INFO 5592 --- [nfoReplicator-0] com.netflix.discovery.DiscoveryClient : DiscoveryClient_DISCOVERYCLIEN
2023-08-01 11:20:28.621 INFO 5592 --- [          main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 901
2023-08-01 11:20:28.621 INFO 5592 --- [          main] .s.c.n.e.s.EurekaAutoServiceRegistration : Updating port to 9010
2023-08-01 11:20:28.644 INFO 5592 --- [nfoReplicator-0] com.netflix.discovery.DiscoveryClient : DiscoveryClient_DISCOVERYCLIEN
2023-08-01 11:20:28.696 INFO 5592 --- [          main] d.Springboot15DiscoveryclientApplication : Started Springboot15Discoveryc

```

API GATEWAY

- 요청에 대한 진입점
- 마이크로 서비스에서 API gateway : 실제 서비스 요청에 대한 진입점
- netflix Zuul

=====> first-service 8081

/welcome

- client =====> localhost:8000/first-service/welcome



Gateway

localhost:8000/second-service/welcome

=====> second-service:8082

/welcome

- [springboot16_firstservice]
 - spring starter project

 **New Spring Starter Project** 

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:


Description:

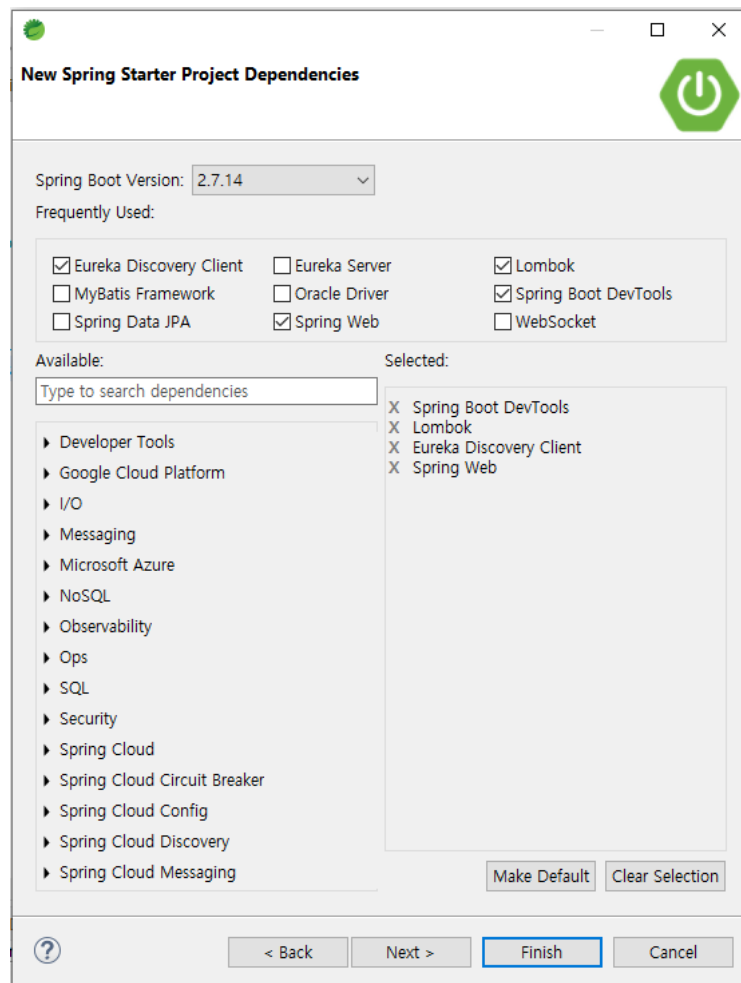
Package:

Working sets

☐ Add project to working sets

Working sets:





- FirstServiceController.java

```
package kr.co.jhta.firstservice.control;

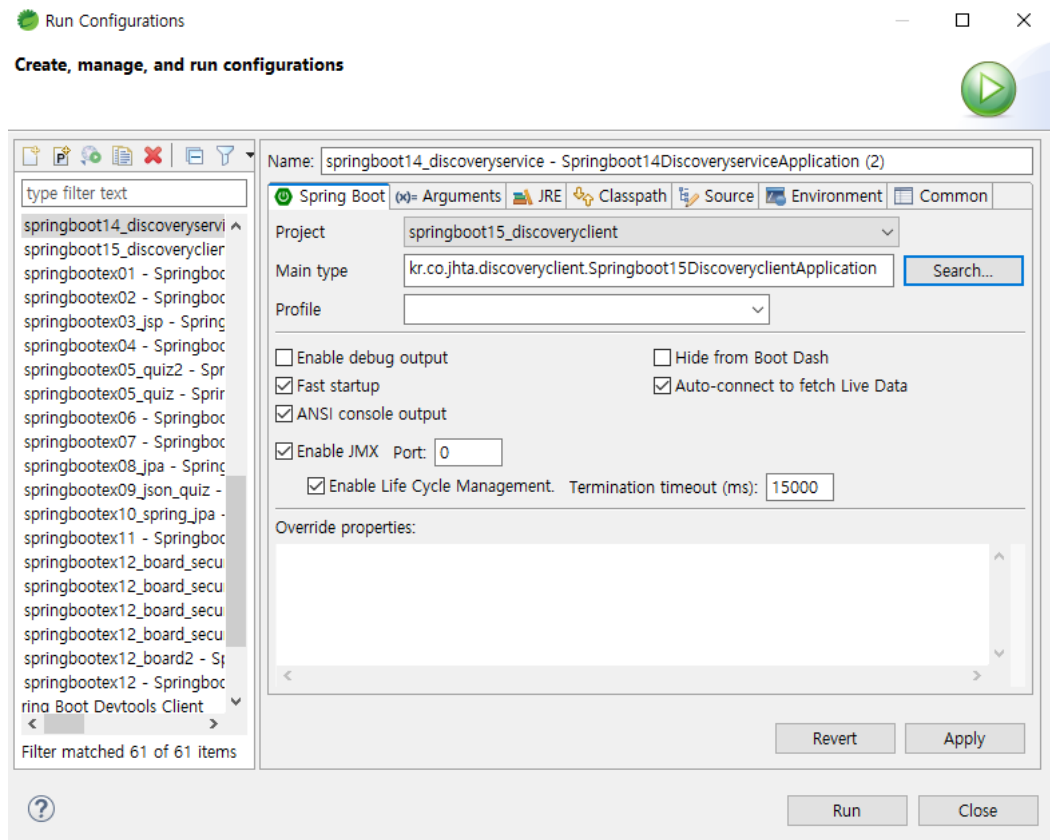
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class FirstServiceController {

    @GetMapping("/welcome")
    public String welcome() {
        return "welcome!!! first Service";
    }


}
```

- build path - configure build path - project facets - java 버전 1.8로




- <http://localhost:8081/welcome>로 실행

- [springboot17_secondservice]



— □ ×



New Spring Starter Project

Service URL

https://start.spring.io

Name

springboot17_secondservice

☒ Use default location

Location

D:\dev\sts4\springboot_workspace\springboot17_secondservice

Browse

Type:

Maven

Packaging:

Jar

Java Version:

8

Language:

Java

Group

kr.co.jhta

Artifact

springboot17_secondservice

Version

0.0.1-SNAPSHOT

Description

Demo project for Spring Boot

Package

kr.co.jhta.secondservice


Working sets

☐ Add project to working sets

New...

Working sets:

Select...

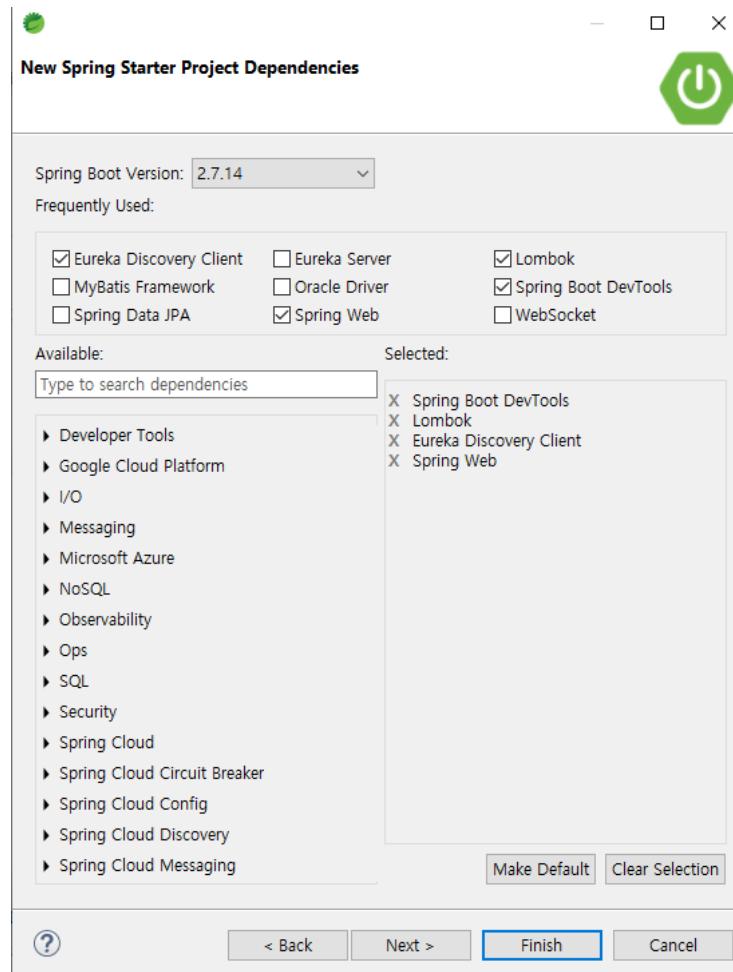


< Back

Next >

Finish

Cancel



- second controller

```
package kr.co.jhta.secondservice.control;

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

@RestController
public class SecondServiceController {

    @GetMapping("/welcome")
    public String welcome() {
        return "welcome!!! Second Service";
    }

}
```

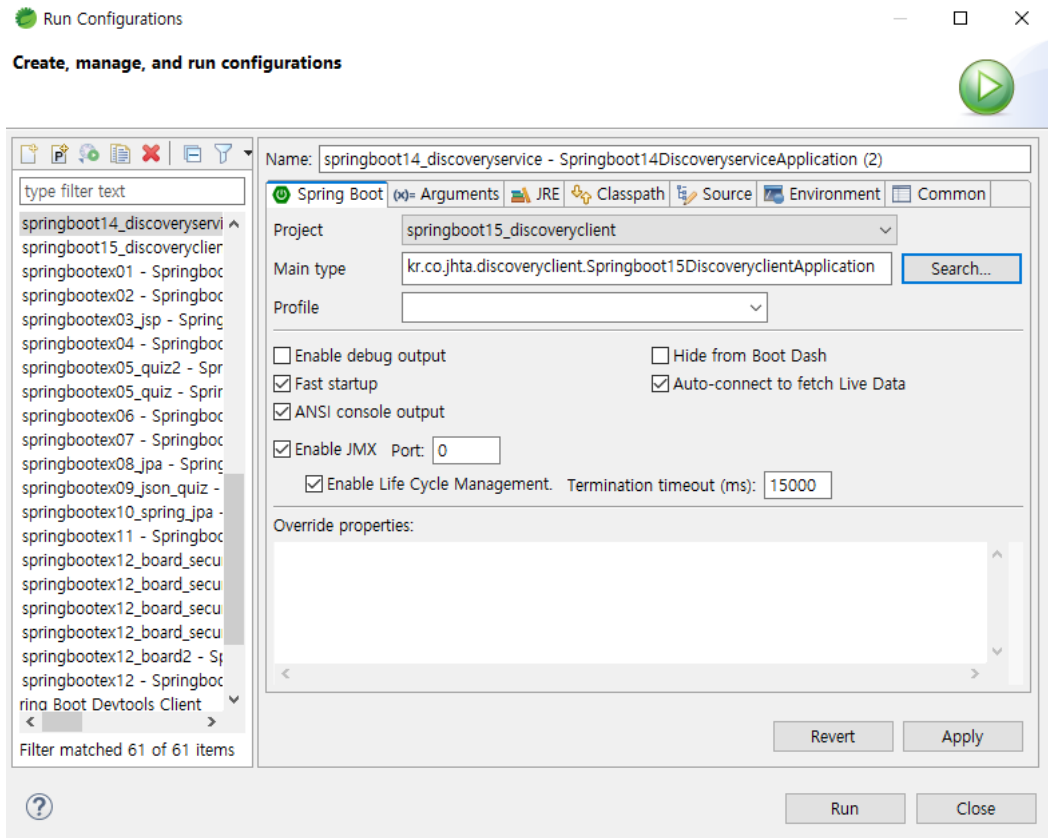
- yaml 파일

```
server:
  port: 8082

spring:
  application:
    name: my-second-service

eureka:
  client:
    register-with-eureka: false
    fetch-registry: false
```

- build path - configure build path - project facets - java 버전 1.8로



- <http://localhost:8082/welcome>로 실행
- [springboot18_zuulservice]
 - pom.xml에 spring-cloud-starter-netflix-zuul 추가

```
<!--https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-netflix-zuul -->
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-zuul</artifactId>
  <version>2.2.10.RELEASE</version>
</dependency>
```

- Springboot18ZuulServiceApplication

```
package kr.co.jhta.zuulservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.zuul.EnableZuulProxy;

@SpringBootApplication
@EnableZuulProxy
public class Springboot18ZuulServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(Springboot18ZuulServiceApplication.class, args);
    }

}
```

- application.configuration 파일을 convert to yaml

```

server:
  port: 8000

spring:
  application:
    name: my-zuul-service

zuul:
  routes:
    first-service:
      path: /first-service/**
      url: http://localhost:8081
    second-service:
      path: /second-service/**
      url: http://localhost:8082

```

- pom.xml에서 버전 수정하기 : zuul의 버전과 spring boot starter의 버전 일치시킴

```

<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.2.10.RELEASE</version>
  <relativePath /> <!-- lookup parent from repository -->
</parent>

```

- 8000 포트번호로 두 개 다 연결 가능

<http://localhost:8000/second-service/welcome>

<http://localhost:8000/first-service/welcome>

요즘에는 netflix zuul 대신 spring cloud로 바뀌고 이름도 다 변경됨

- pom.xml에 롬복 추가

```

<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <version>1.18.28</version>
  <scope>provided</scope>
</dependency>

```

- ZuulLoginFilter

```

package kr.co.jhta.zuulservice.filter;

import javax.servlet.http.HttpServletRequest;

import org.springframework.stereotype.Component;

import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.context.RequestContext;
import com.netflix.zuul.exception.ZuulException;

import lombok.extern.slf4j.Slf4j;

@Slf4j
@Component // 일반적인 빈
public class ZuulLoginFilter extends ZuulFilter {

    @Override
    public boolean shouldFilter() {
        // TODO Auto-generated method stub
        return true; // 필터 사용
    }

    @Override
    public Object run() throws ZuulException {
        // 필터가 동작 된다면 이 부분이 실행됨
        log.info("로그 시작 =====> ");
        // 요청 URI를 로그로 기록
        RequestContext ctx = RequestContext.getCurrentContext();
    }
}

```

```

        HttpServletRequest request = ctx.getRequest();
        String uri = request.getRequestURI();
        log.info("요청 uri : {} ", uri);
        log.info("요청 uri : {} ", RequestContext.getCurrentContext()
                .getRequest()
                .getRequestURI()); // 원래는 위의 코드보다 이렇게 더 많이 씀
        log.info("로그 끝 =====> ");
        return null;
    }

    @Override
    public String filterType() {
        // TODO Auto-generated method stub
        return "pre"; // 진입 쪽 필터
    }

    @Override
    public int filterOrder() {
        // TODO Auto-generated method stub
        return 0;
    }
}

```

◦ 결과

```

[2m2023-08-01 12:49:58.731[0;39m [32m INFO[0;39m [35m3448[0;39m [2m---[0;39m [2m[nio-8000-exec-1][0;39m [36mk.c.j.z.filter.Zuu
[2m2023-08-01 12:49:58.731[0;39m [32m INFO[0;39m [35m3448[0;39m [2m---[0;39m [2m[nio-8000-exec-1][0;39m [36mk.c.j.z.filter.Zuu
[2m2023-08-01 12:49:58.732[0;39m [32m INFO[0;39m [35m3448[0;39m [2m---[0;39m [2m[nio-8000-exec-1][0;39m [36mk.c.j.z.filter.Zuu
[2m2023-08-01 12:49:58.732[0;39m [32m INFO[0;39m [35m3448[0;39m [2m---[0;39m [2m[nio-8000-exec-1][0;39m [36mk.c.j.z.filter.Zuu

```