

# Load Balancer 적용



사용자 서비스에서 알림 서비스를 호출 할 수 있도록 만들어 보자.

# 프로젝트 생성 – user project

<https://start.spring.io/> 접속



## Project

☒ Gradle - Groovy ☐ Gradle - Kotlin ☐ Maven

## Language

☒ Java ☐ Kotlin ☐ Groovy

## Spring Boot

☐ 4.0.0 (SNAPSHOT) ☐ 3.5.1 (SNAPSHOT) ☒ 3.5.0 ☐ 3.4.7 (SNAPSHOT)  
☐ 3.4.6 ☐ 3.3.13 (SNAPSHOT) ☐ 3.3.12

## Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☐ 24 ☐ 21 ☒ 17

## Dependencies

ADD DEPENDENCIES... CTRL + B

### Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

### Cloud LoadBalancer SPRING CLOUD ROUTING


Client-side load-balancing with Spring Cloud LoadBalancer.

### Lombok DEVELOPER TOOLS

Java annotation library which helps to reduce boilerplate code.

# 프로젝트 생성 – alim project

<https://start.spring.io/> 접속



**Project**  
☒ Gradle - Groovy ☐ Gradle - Kotlin ☐ Maven

**Language**  
☒ Java ☐ Kotlin ☐ Groovy

**Spring Boot**  
☐ 4.0.0 (SNAPSHOT) ☐ 3.5.1 (SNAPSHOT) ☒ 3.5.0 ☐ 3.4.7 (SNAPSHOT)  
☐ 3.4.6 ☐ 3.3.13 (SNAPSHOT) ☐ 3.3.12

**Project Metadata**  
Group   
Artifact   
Name   
Description   
Package name   
Packaging ☒ Jar ☐ War  
Java ☐ 24 ☐ 21 ☒ 17

**Dependencies** ADD DEPENDENCIES... CTRL + B

**Spring Web** WEB  
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

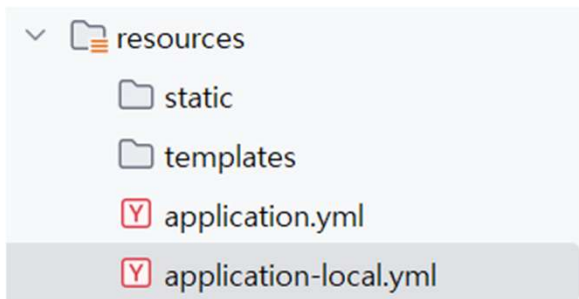
**Cloud LoadBalancer** SPRING CLOUD ROUTING  
Client-side load-balancing with Spring Cloud LoadBalancer.

**Lombok** DEVELOPER TOOLS  
Java annotation library which helps to reduce boilerplate code.

## 설정 분리 – 모든 프로젝트 대상

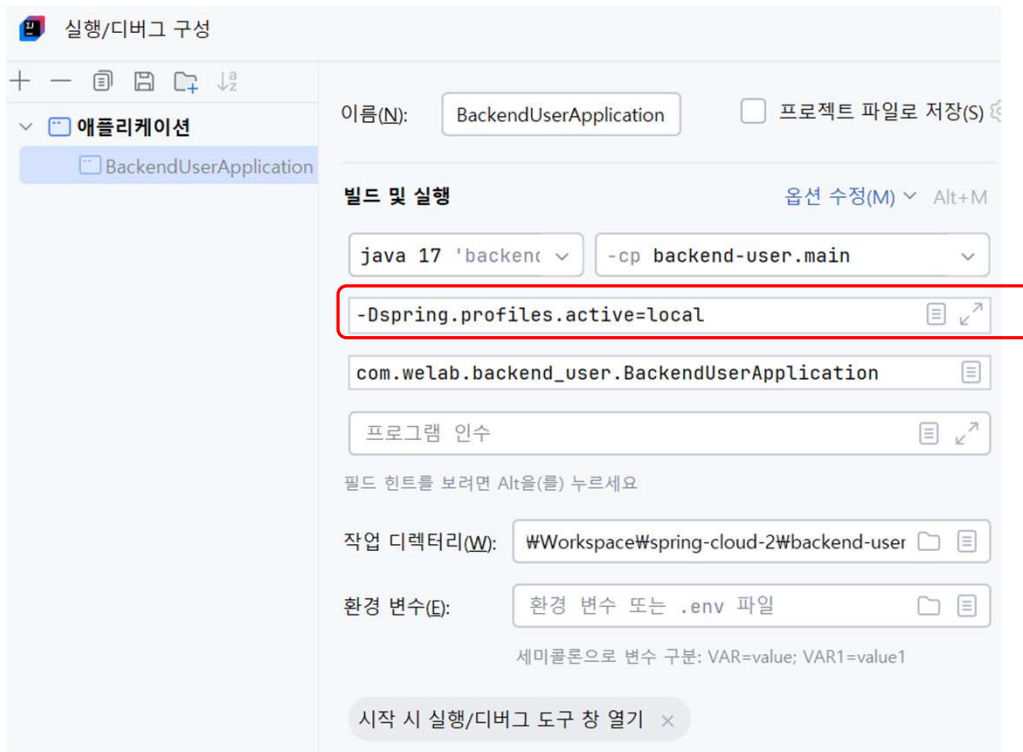
---

설정을 독립적으로 관리해야 하기 때문에 application.yml을 분리하자.



# 설정 분리 – 모든 프로젝트 대상

PC에서는 active profile을 local로 지정하자.



# Alim 프로젝트 설정

---

```
spring:  
  application:  
    name: backend-alim
```

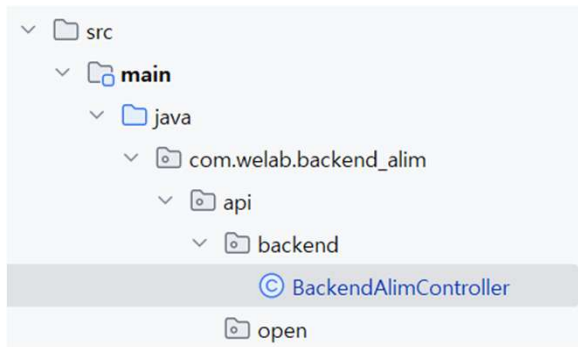
application.yml

```
server:  
  port: 8082
```

application-local.yml

# Alim 프로젝트 코드 작성

User 서비스에서 Alim 서비스를 호출할 것이기 때문에 Alim 서비스에서는 Rest API를 제공하자.



```
@Slf4j
@RestController
@RequestMapping(value = "/backend/alim/v1", produces = MediaType.APPLICATION_JSON_VALUE)
public class BackendAlimController {
    @GetMapping(value = "/hello")
    public String hello() {
        return "알림 백엔드 서비스가 호출되었습니다.";
    }
}
```

## User 프로젝트 설정

```
spring:
  application:
    name: backend-user
```

application.yml

```
server:
  port: 8081
```

```
spring:
  cloud:
    discovery:
      client:
        simple:
          instances:
```

# 'my-backend-service'에 대한 인스턴스 정의

alim-service:

- service-id: alim-service  
uri: http://localhost:8082
- service-id: alim-service  
uri: http://localhost:8082

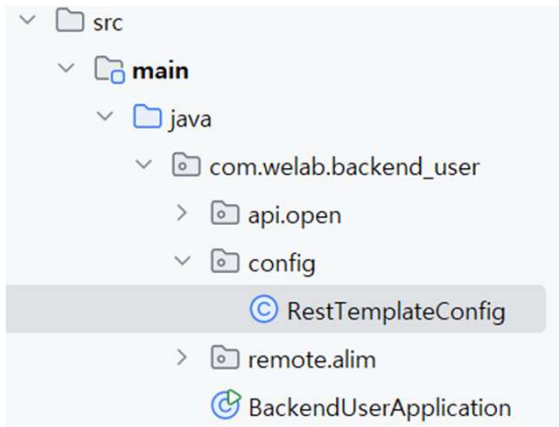
알림 서비스 IP 2개에 대해 로드밸런싱  
(자동이 아닌 수동 명시)

eureka를 통해  
서비스 이름을 통해  
uri 자동 매핑

application-local.yml



# User 프로젝트 코드 작성



@Configuration

```
public class RestTemplateConfig {
```

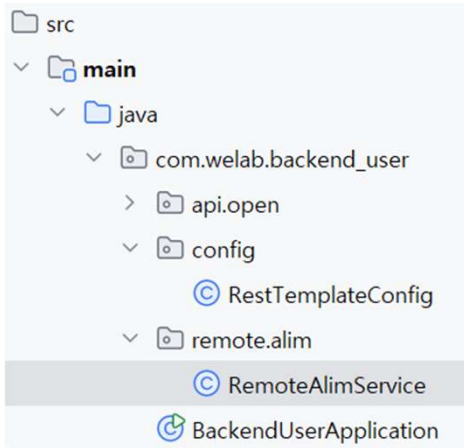
@Bean

@LoadBalanced

API

```
public RestTemplate restTemplate() {  
    return new RestTemplate();  
}  
}
```

# User 프로젝트 코드 작성



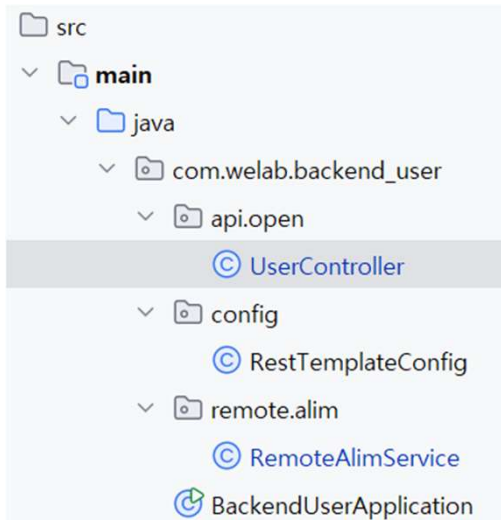
```
@Service
@RequiredArgsConstructor
public class RemoteAlimService {
    private final RestTemplate restTemplate;

    public String callAlimHello() {
        return restTemplate.getForObject(
            "http://alim-service/backend/alim/v1/hello",
            String.class
        );
    }
}
```

URI  
String

API

# User 프로젝트 코드 작성



```
@Slf4j
@RestController
@RequestMapping(
    value = "/api/user/v1",
    produces = MediaType.APPLICATION_JSON_VALUE)
@RequiredArgsConstructor
public class UserController {
    private final RemoteAlimService remoteAlimService;

    @GetMapping(value = "/test")
    public String test() {
        return remoteAlimService.callAlimHello();
    }
}
```