

전자정부 표준프레임워크 개발환경 실습교재



1. 실습 순서



1. LAB 1-1 개발환경 설치
2. LAB 1-2 프로젝트 생성
3. LAB 1-3 API 호출 및 JUnit 테스트

LAB 1-1 개발환경 설치(1/2)

Step 1-1-00. 표준프레임워크 포털에서 운영체제에 맞는 개발자용 개발환경을 다운로드 받고 설치방법을 참고하여 설치한다.

<https://www.egovframe.go.kr/home/sub.do?menuNo=94>

- 개발자용 개발환경 64bit(Implementation Tool) Version 4.1.0
- 개발자용 개발환경 for Mac (x86_64/AArch64)(Implementation Tool) Version 4.1.0
- 개발자용 개발환경 for Linux 64bit(Implementation Tool) Version 4.1.0

(개발환경 4.1.0 에는 lombok이 이미 설치되어 있어 Step 1-1-07 까지는 참고만 한다.)

Step 1-1-01. Eclipse IDE 에 lombok 을 설치하기 위해

<https://projectlombok.org/download>

주소로 접속해서 jar 파일을 다운로드한다.

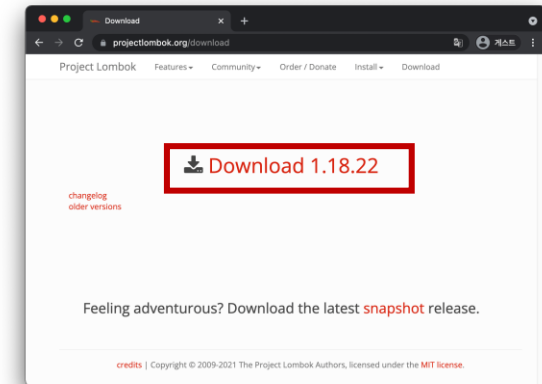
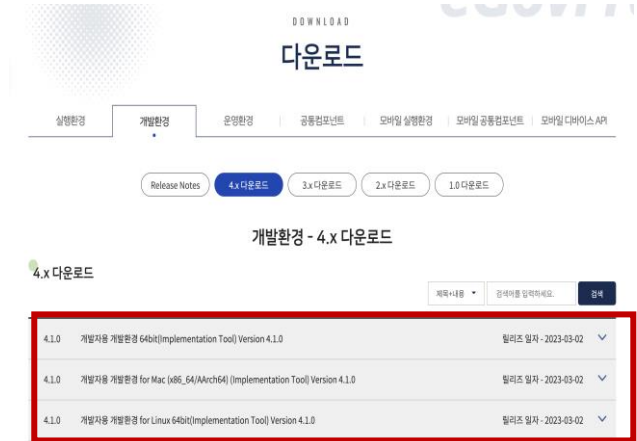
Step 1-1-02. 터미널을 열고 다운로드한 lombok.jar 파일이 있는 디렉토리로 이동해서 `java -jar lombok.jar` 을 실행한다.

Step 1-1-03. Project Lombok Installer 창이 열리면 Specify location... 버튼을 클릭한다.

Step 1-1-04. 파일 탐색기가 열리면 개발자용 개발환경(Eclipse IDE)의 실행파일을 선택한다.

Step 1-1-05. 선택한 Eclipse IDE의 경로를 확인하고 Install / Update 버튼을 클릭한다.

Step 1-1-06. Install successful을 확인하고 Quit Installer 버튼을 클릭한다.



LAB 1-1 개발환경 설치(2/2)

Step 1-1-07. 위 방법으로 Lombok 이 정상 작동하지 않는 경우 eclipse.ini 파일에 마지막에 `-javaagent:${user.home}/Download/lombok.jar` 을 추가한다. (Eclipse IDE 실행파일 경로에 jar 파일을 복사한 경우 - `javaagent:lombok.jar` 를 추가하면 된다.)

* eclipse.ini 파일 경로

Windows -

`%HOMEDRIVE% %HOMEPATH% \Downloads \eGovFrameDev-3.10.0-64bit \eclipse \eclipse.exe`

Mac - `/Applications/eGovFrameDev-3.10.0-Mac-`

`64bit.app/Contents/Eclipse/eclipse.ini`

```
eclipse.ini X
Applications > eGovFrameDev-3.10.0-Mac-64bit.app > Contents > Eclipse > eclipse.ini
1  -startup
2  ..\Eclipse\plugins\org.eclipse.equinox.launcher_1.5.700.v20200207-2156.jar
3  --launcher.library
4  ..\Eclipse\plugins\org.eclipse.equinox.launcher.cocoa.macosx.x86_64_1.1.1200.v20200508-1552
5  -product
6  org.eclipse.epp.package.jee.product
7  --showplash
8  org.eclipse.epp.package.common
9  --launcher.defaultAction
10 openFile
11 --launcher.defaultAction
12 openFile
13 --launcher.appendVmargs
14 -vm
15 /Library/Java/JavaVirtualMachines/jdk1.8.0_301.jdk/Contents/Home/bin/java
16 -vmargs
17 -Dfile.encoding=UTF-8
18 -Dosgi.requiredJavaVersion=1.8
19 -Dosgi.instance.area.default=@user.home/eclipse-workspace
20 --XX:+UseG1GC
21 --XX:+UseStringDuplication
22 --add-modules=ALL-SYSTEM
23 --XstartOnFirstThread
24 -Dorg.eclipse.swt.internal.carbon.smallFonts
25 -Dosgi.requiredJavaVersion=1.8
26 -Dosgi.dataAreaRequiresExplicitInit=true
27 -Xms256m
28 -Xmx2048m
29 --add-modules=ALL-SYSTEM
30 -Xdock:icon=../Resources/Eclipse.icns
31 --XstartOnFirstThread
32 -Dorg.eclipse.swt.internal.carbon.smallFonts
33 -javaagent:lombok.jar
```

LAB 1-2 프로젝트 생성(1/6)

Step 1-2-00. 개발환경 Eclipse IDE 를 실행한다.

Step 1-2-01. Eclipse IDE Launcher 에서 Workspace 를 github 에서 내려 받은 `/${home}/workspace.edu/egovframe-msa-edu/backend` 로 선택하고 Launch 버튼을 클릭한다.

Step 1-2-02. Eclipse IDE 메뉴에서 File>Import... 를 클릭한다.

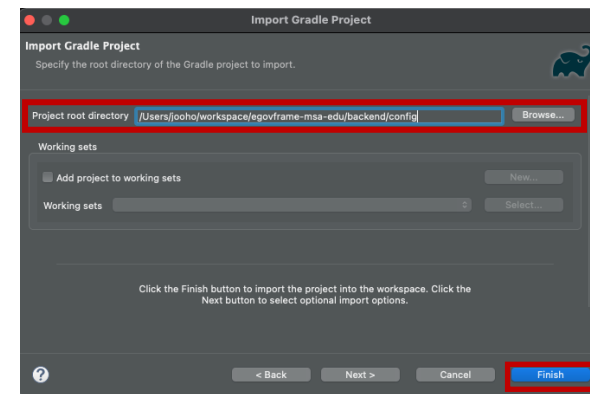
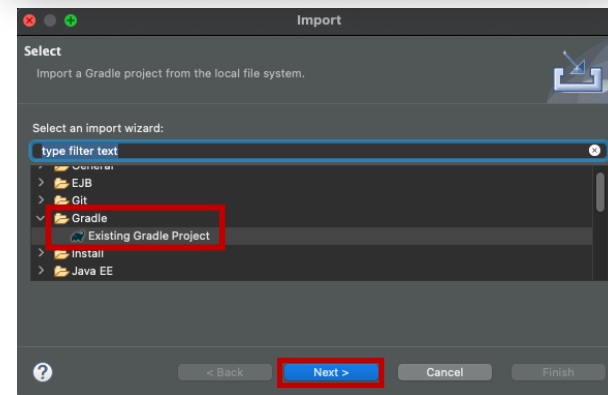
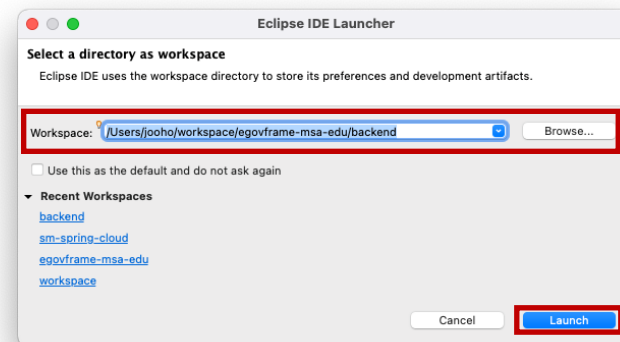
Step 1-2-03. Import 창이 열리면 Gradle>Existing Gradle Project 를 선택하고 Next 버튼을 클릭한다.

Step 1-2-04. Import Gradle Project 창이 열리면 Next 버튼을 클릭한다.

Step 1-2-05. Project root directory 에서 `/${home}/workspace.edu/egovframe-msa-edu/backend/config`를 선택하고 Finish 버튼을 클릭한다.

Step 1-2-06. 02 ~ 05 의 과정을 반복해서 아래의 프로젝트(소규모는 1~6, 대규모는 1~9)를 import 한다.

- | | |
|-------------------|----------------------------|
| 1. config | 6. board-service |
| 2. discovery | 7. reserve-check-service |
| 3. apigateway | 8. reserve-item-service |
| 4. user-service | 9. reserve-request-service |
| 5. portal-service | |



개발환경 4.x 버전은 위의 9개 서비스에 대해 소스 자동생성 기능을 제공한다. ([첨부] 개발환경에서 프로젝트 생성 참조)

LAB 1-2 프로젝트 생성(2/6)

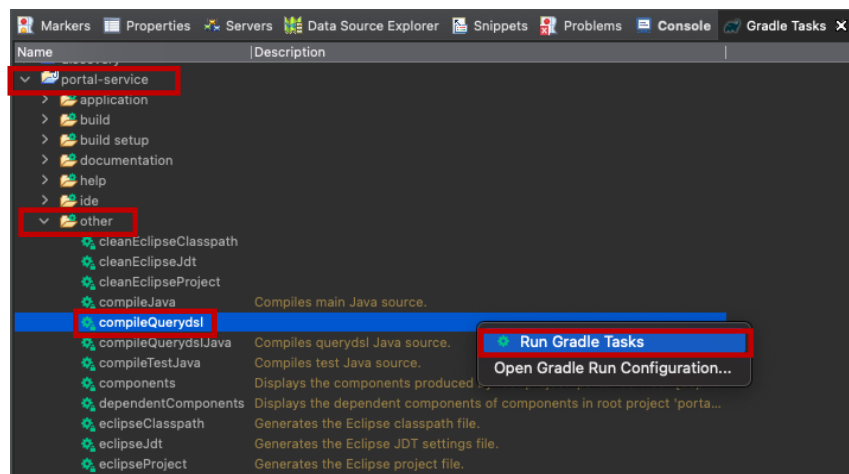
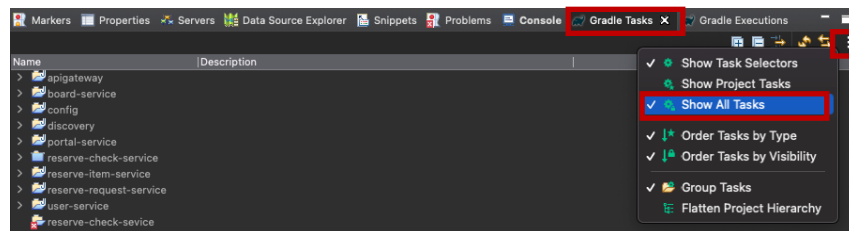
Step 1-2-07. 모든 프로젝트를 import 하고 Project Explorer 를 확인하면 board-service, portal-service, user-service 프로젝트에 오류 표시가 출력된다. querydsl 로 generate 되는 클래스들을 build path 에 추가 해야 한다.

Step 1-2-08. 상단 메뉴에서 Window>Show View>Other 을 클릭해서 열린 창에서 Gradle>Gradle Tasks 를 선택하고 Open 버튼을 클릭하면 Gradle Tasks 탭이 열린다.

Step 1-2-09. Gradle Tasks 오른쪽 윗부분의 View Menu 버튼을 클릭해서 Show All Tasks 를 체크한다.

Step 1-2-10. Gradle Tasks 에서 portal-service>other>compileQuerydsl 을 더블클릭 또는 우클릭 후 Run Gradle Tasks 를 클릭하면 build 가 시작된다.

Step 1-2-11. Project Explorer 에서 board-service, portal-service, user-service 를 선택하고 F5 또는 우클릭 후 Refresh 를 클릭해서 프로젝트를 새로고침한다.



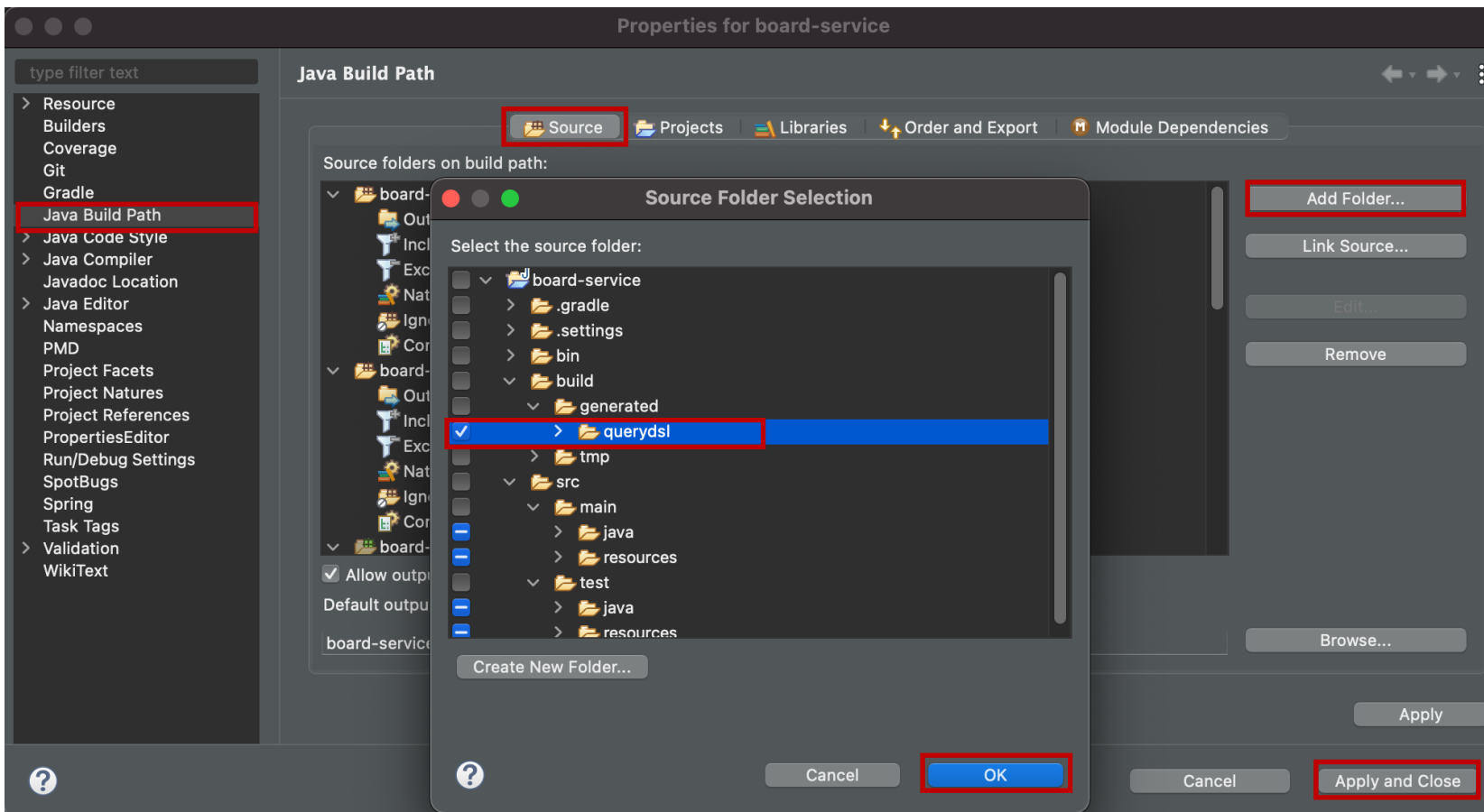
LAB 1-2 프로젝트 생성(3/6)

Step 1-2-12. Project Explorer 에서 board-service, portal-service, user-service 를 우클릭하고 Properties 를 클릭한다.

Step 1-2-13. Properties 창이 열리면 왼쪽 메뉴에서 Java Build Path를 선택하고 오른쪽 Source 탭에서 Add Folder... 버튼을 클릭한다.

Step 1-2-14. Source Folder Selection 창이 열리면 build>generated>querydsl 을 체크하고 OK 버튼을 클릭한다.

Step 1-2-15. Properties 창에서 Apply and Close 버튼을 클릭하면 창이 닫히면서 프로젝트를 다시 빌드하고 오류 표시가 없어진다.



LAB 1-2 프로젝트 생성(4/6)

Step 1-2-16. ELK 사용 시 프로파일 설정(optional)

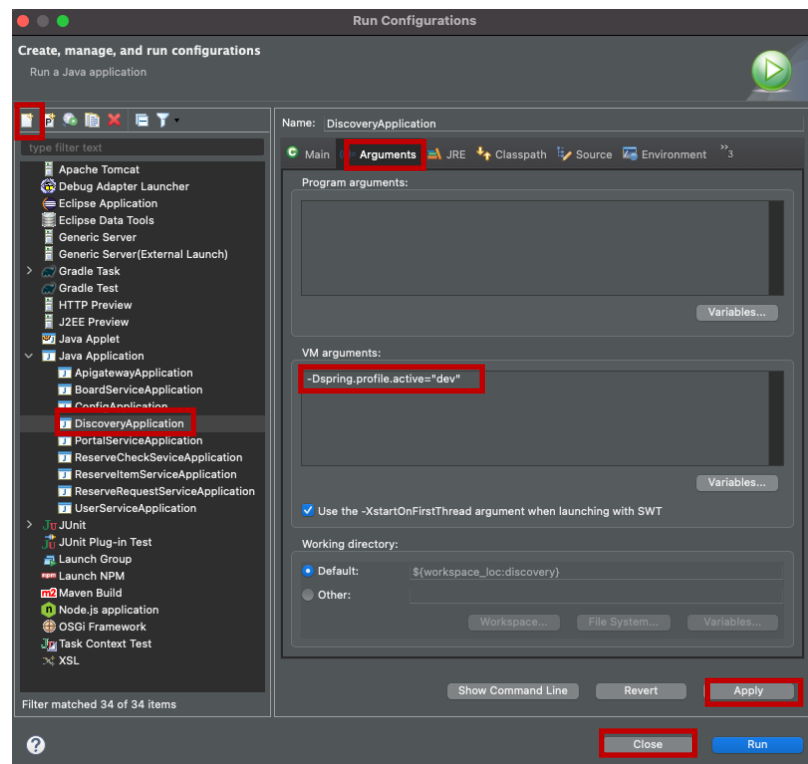
- 프로파일은 기본적으로 "default" 가 적용된다.
- "test" 로 프로파일을 변경할 경우 각 어플리케이션 실행 시 config 에서 *-test.yml 파일을 불러와서 설정이 적용되고 mysql 이 아닌 h2 in-memory 데이터베이스와 연결된다. h2 in-memory 데이터베이스는 휘발성이고 초기 데이터가 없으므로 JUnit Test 를 실행할 경우에서만 적용되어야 한다.
- ELK 를 사용하기 위해서는 "default" 가 아닌 "local", "dev", "prod" 등 다른 프로파일을 적용하거나, 모든 프로젝트의 /src/main/resources/logback-spring.xml 파일 내용을 수정해야 한다. 한가지 방법만 적용하면 된다.

1. 프로파일 변경

- 1) /src/main/resources/application.yml 파일에서 spring.profile.active 를 변경한다.

```
spring:  
  profile:  
    active: dev  
  ...
```

- 2) 각 프로젝트의 *Application.java 파일(DiscoveryApplication 의 경우 /src/main/java/org/egovframe/cloud/discovery/DiscoveryApplication.java)을 우클릭하고 Run AS>Run Configurations... 를 클릭한다. 창이 열리면 Arguments 탭으로 이동해 VM arguments 항목에 -Dspring.profiles.active="dev" 값을 입력하고 Apply 버튼을 클릭한다.
- 3) ConfigApplication의 경우 설정 파일을 로컬 경로에서 읽어올 수 있도록 -Dspring.profiles.active="native,dev" 과 같이 "native"도 함께 추가해야 한다. Run Configurations 창에서 Java Application 에 보이지 않을 경우 화면 상단의 추가 버튼을 클릭하면 Application 이 추가된다.



LAB 1-2 프로젝트 생성(5/6)

Step 1-2-16. ELK 사용 시 프로파일 설정(optional) 계속

- 아래와 같이 profile이 적용되지 않도록 logback-spring.xml 파일의 내용을 수정한다.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
    <encoder>
      <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} %thread %-5level %logger - %m%n</pattern>
    </encoder>
  </appender>

  <property name="destination" value="${logstash_hostname:-localhost:5001}" />
  <property name="app_name" value="${app_name:-config-server}" />

  <!-- ELK - Logstash 로 로그를 전송하기 위한 appender -->
  <appender name="LOGSTASH" class="net.logstash.logback.appender.LogstashTcpSocketAppender">
    <destination>${destination}</destination><!-- native profile => localhost:8088 -->
    <encoder class="net.logstash.logback.encoder.LogstashEncoder">
      <customFields>{"app.name": "${app_name}"</customFields>
    </encoder>
  </appender>
  <root level="WARN">
    <appender-ref ref="LOGSTASH" />
    <appender-ref ref="STDOUT" />
  </root>
</configuration>
```

- ※ application 실행 시 **java.lang.IllegalStateException: Logback configuration error detected** 예외가 발생하는 경우 logback-spring.xml 파일 하단의 typeAliases 를 삭제한다. Eclipse IDE 에서 xml 형식의 파일을 Mapper로 인식해서 파일 저장 시 자동으로 입력되는 문제가 있다.

```
...
<typeAliases></typeAliases>
</configuration>
```

LAB 1-2 프로젝트 생성(6/6)

Step 1-2-17. 설정 파일 경로(optional)

설정 파일은 Windows 기준으로 경로가 설정되어 있다. MacOS 인 경우 config 프로젝트에서 /src/main/resources/application.yml 파일을 수정한다.

```
spring:
  application:
    name: config-service
  profiles:
    active: native,default # native file repository
  cloud:
    config:
      server:
        native:
          # search-locations: file:///${user.home}/workspace.edu/egovframe-msa-edu/config # Windows
          search-locations: file:///${user.home}/workspace.edu/egovframe-msa-edu/config # MacOS
```

Step 1-2-18. OAuth 2.0(optional)

OAuth 2.0 을 사용하기 위해서는 google, naver, kakao 에서 제공하는 가이드 문서를 참고하여 애플리케이션을 등록하고 client id, client secret 을 user-service/src/main/resources/application.yml 파일에 설정해야 한다.

- google : <https://console.cloud.google.com/apis/credentials>
- naver : <https://developers.naver.com/apps/#/register?api=nvlogin>
- kakao : <https://developers.kakao.com/product/kakaoLogin>

```
security:
  oauth2:
    client:
      registration:
        # /oauth2/authorization/google
      google:
        client-id: google_client_id # TODO https://console.cloud.google.com
        client-secret: google_client_secret # TODO
  ...
```

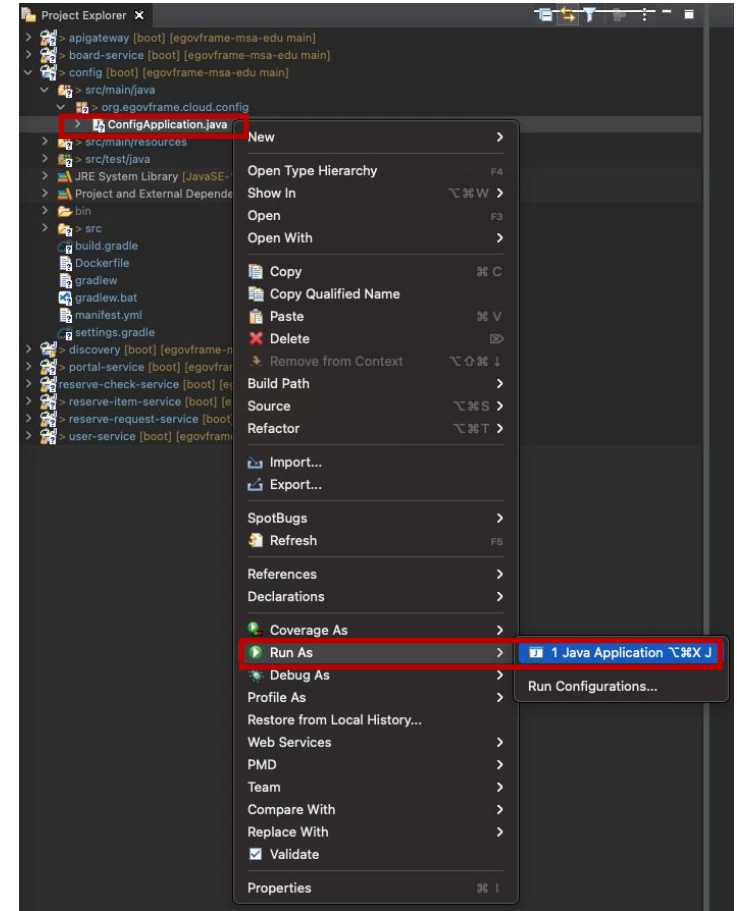
LAB 1-3 API 호출 및 JUnit 테스트(1/8)

Step 1-3-00. Project Explorer 에서 dicovery 프로젝트의 src/main/java/org/egovframe/cloud/discovery/DiscoveryApplication.java 파일을 우클릭하고 Run As>Java Application을 클릭한다.

Step 1-3-01. *-service 는 config 에서 설정 정보를 가져오고 discovery 의 client 로 등록되어야 하고 apigateway 를 통해서 외부와 통신 한다. 따라서 discovery, config, apigateway, *-service 순으로 application 을 구동한다.

Step 1-3-02. Console 창에서 모든 application 이 정상 시작 된 것을 확인 한다. 오류가 발생할 경우 config 파일 경로, RabbitMQ 연결 정보 등을 확인한다.

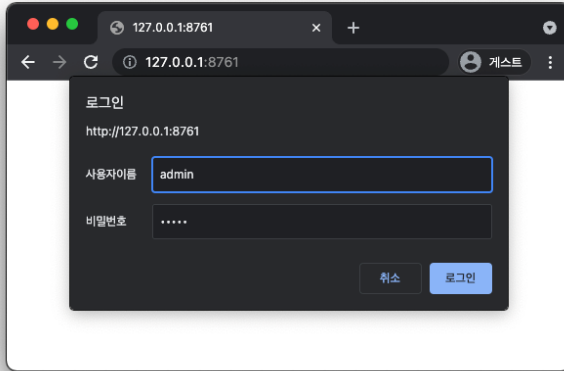
config/src/main/resources/application.yml 파일에서 spring.cloud.config.server.native.search-locations 가 Windows 로 설정되어 있으므로 다른 운영체제를 사용하는 경우 변경한다.



LAB 1-3 API 호출 및 JUnit 테스트(2/8)

Step 1-3-03. Eureka – discovery 확인

- 브라우저 > <http://localhost:8761> 접속 > admin/admin 입력 > 로그인



Application	AMIs	Availability Zones	Status
APIGATEWAY	n/a (1)	(1)	UP (1) - apigateway:cf98e6dbb52ca44f13dda02c028244e8
BOARD-SERVICE	n/a (1)	(1)	UP (1) - board-service:adadc9b6320f773a5a49085ac201c894
PORTAL-SERVICE	n/a (1)	(1)	UP (1) - portal-service:a4a776d48b916c0aa3e3db8f69f3663
RESERVE-CHECK-SERVICE	n/a (1)	(1)	UP (1) - reserve-check-service:4311a1bc1b919f2ec0e15b4e5d82fd39
RESERVE-ITEM-SERVICE	n/a (1)	(1)	UP (1) - reserve-item-service:a0d8f7efef51d0c16a17f79e87df5867
RESERVE-REQUEST-SERVICE	n/a (1)	(1)	UP (1) - reserve-request-service:b8dc5e0d09f65d3e6c7c75fe71cc09b4
USER-SERVICE	n/a (1)	(1)	UP (1) - user-service:8d3bc595e1a61c0255e85ddfd1f8f6e0

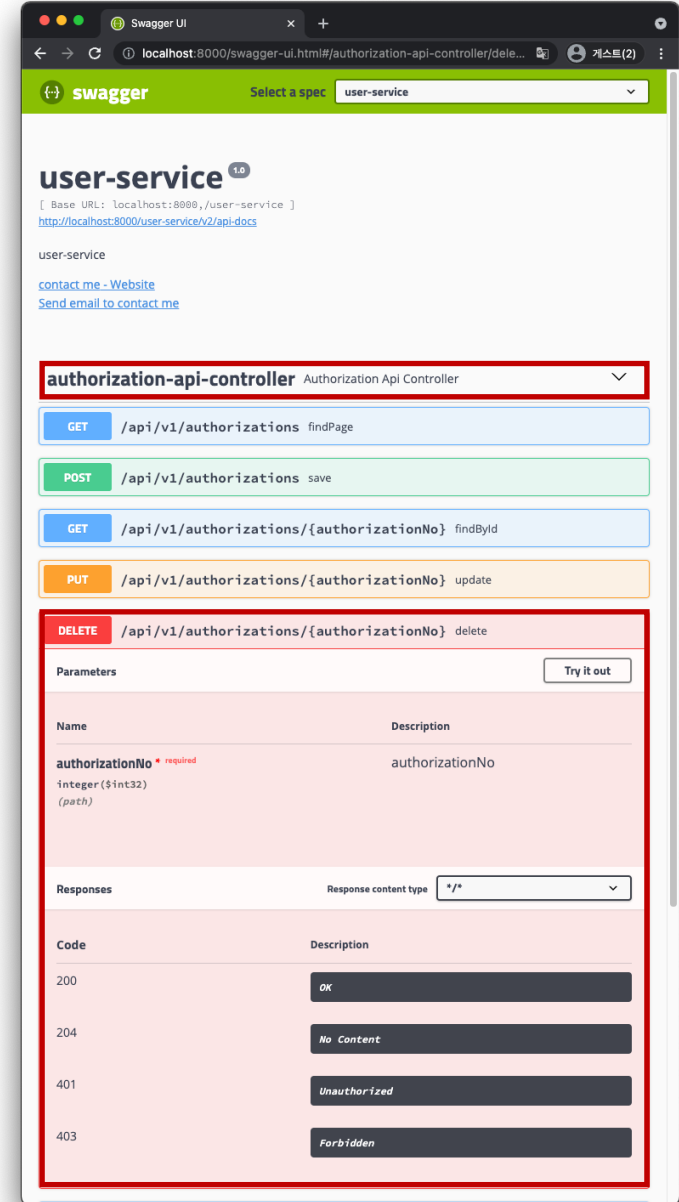
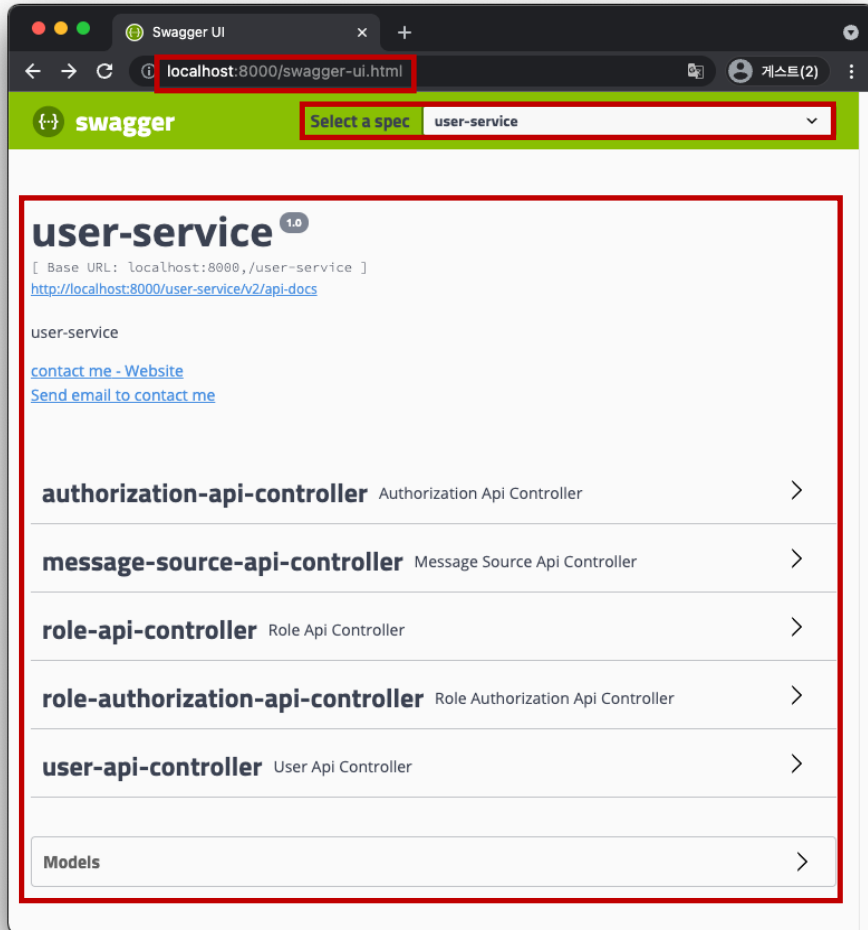
Name	Value
total-avail-memory	828mb
num-of-cpus	8
current-memory-usage	354mb (42%)
server-upptime	03:20
registered-replicas	http://localhost:8761/eureka/
unavailable-replicas	http://localhost:8761/eureka/
available-replicas	

Name	Value
IpAddr	192.168.50.145
status	UP

LAB 1-3 API 호출 및 JUnit 테스트(3/8)

Step 1-3-04. Swagger - API 확인

- 브라우저 > <http://localhost:8000/swagger-ui.html> 접속
- Select a spec 에서 확인하고자 하는 서비스 선택
- 조회된 api controller, Model 을 클릭해서 API의 상세 정보 확인



LAB 1-3 API 호출 및 JUnit 테스트(4/8)

Step 1-3-05. 설정 정보 확인

- Postman 실행 > GET <http://localhost:8888/user-service/dev> 입력 > Send > 사용자 서비스의 개발 설정 정보 확인
- Postman 실행 > GET <http://localhost:8888/user-service/test> 입력 > Send > 사용자 서비스의 테스트 설정 정보 확인

```
1 {
2   "name": "user-service",
3   "profiles": [
4     "dev"
5   ],
6   "label": null,
7   "version": null,
8   "state": null,
9   "propertySources": [
10    {
11      "name": "file:/Users/jooho/workspace/egovframe-msa-edu/config/user-service.yml",
12      "source": {
13        "database.url": "jdbc:mysql://localhost:3306/msaportal",
14        "spring.datasource.url": "${database.url}?serverTimezone=Asia/Seoul",
15        "spring.datasource.username": "msaportal",
16        "spring.datasource.password": "msaportal",
17        "spring.datasource.driver-class-name": "com.mysql.cj.jdbc.Driver",
18        "spring.mail.host": "smtp.gmail.com",
19        "spring.mail.port": 587,
20        "spring.mail.username": "email_username",
21        "spring.mail.password": "email_password",
22        "spring.mail.properties.mail.smtp.auth": true,
23        "spring.mail.properties.mail.smtp.starttls.enable": true,
24        "spring.mail.properties.mail.smtp.starttls.required": true
25      }
26    },
27    {
28      "name": "file:/Users/jooho/workspace/egovframe-msa-edu/config/application.yml",
29      "source": {
30        "token.expiration_time": 7200000,
31        "token.refresh_time": 86400000,
32        "token.secret": "token_secret",
33        "eureka.instance.instance-id": "${spring.application.name}:${spring.application.instance_id:${random.value}}",
34        "eureka.client.register-with-eureka": true,
35        "eureka.client.fetch-registry": true,
36        "eureka.client.service-url.defaultZone": "http://admin:admin@localhost:8761/eureka",
37        "file.directory": "${user.home}/msa-attach-volume",
38        "file.url": "http://localhost:8080",
39        "messages.directory": "${file.directory}/messages",
40        "logstash.url": "localhost:8086",
41        "apigateway.host": "http://localhost:${server.port}",
42        "spring.rabbitmq.host": "localhost",
43        "spring.rabbitmq.port": 5672,
44        "spring.rabbitmq.username": "guest",
45        "spring.rabbitmq.password": "guest",
46        "spring.zipkin.base-url": "http://localhost:8085"
47      }
48    }
49  ]
50 }
```

```
1 {
2   "name": "user-service",
3   "profiles": [
4     "test"
5   ],
6   "label": null,
7   "version": null,
8   "state": null,
9   "propertySources": [
10    {
11      "name": "file:/Users/jooho/workspace/egovframe-msa-edu/config/user-service-test.yml",
12      "source": {
13        "spring.datasource.url": "jdbc:h2:tcp://localhost/~querydsl",
14        "spring.datasource.username": "sa",
15        "spring.datasource.password": "",
16        "spring.datasource.driver-class-name": "org.h2.Driver"
17      }
18    },
19    {
20      "name": "file:/Users/jooho/workspace/egovframe-msa-edu/config/user-service.yml",
21      "source": {
22        "database.url": "jdbc:mysql://localhost:3306/msaportal",
23        "spring.datasource.url": "${database.url}?serverTimezone=Asia/Seoul",
24        "spring.datasource.username": "msaportal",
25        "spring.datasource.password": "msaportal",
26        "spring.datasource.driver-class-name": "com.mysql.cj.jdbc.Driver",
27        "spring.mail.host": "smtp.gmail.com",
28        "spring.mail.port": 587,
29        "spring.mail.username": "email_username",
30        "spring.mail.password": "email_password",
31        "spring.mail.properties.mail.smtp.auth": true,
32        "spring.mail.properties.mail.smtp.starttls.enable": true,
33        "spring.mail.properties.mail.smtp.starttls.required": true
34      }
35    },
36    {
37      "name": "file:/Users/jooho/workspace/egovframe-msa-edu/config/application.yml",
38      "source": {
39        "token.expiration_time": 7200000,
40        "token.refresh_time": 86400000,
41        "token.secret": "token_secret",
42        "eureka.instance.instance-id": "${spring.application.name}:${spring.application.instance_id:${random.value}}",
43        "eureka.client.register-with-eureka": true,
44        "eureka.client.fetch-registry": true,
45        "eureka.client.service-url.defaultZone": "http://admin:admin@localhost:8761/eureka",
46        "file.directory": "${user.home}/msa-attach-volume",
47        "file.url": "http://localhost:8080",
48        "messages.directory": "${file.directory}/messages",
49        "logstash.url": "localhost:8086",
50        "apigateway.host": "http://localhost:${server.port}",
51        "spring.rabbitmq.host": "localhost",
52        "spring.rabbitmq.port": 5672,

```

LAB 1-3 API 호출 및 JUnit 테스트(5/8)

Step 1-3-06. API 직접 호출(portal-service 의 콘텐츠 상세 조회 API 호출)

- Eureka 에서 Instances currently registered with Eureka 의 Status 중 portal-service:xxx 우클릭 후 링크 주소 복사로 포트 확인
- Postman 실행 > GET <http://localhost:포트주소/api/v1/contents/1> 입력 > Send

The screenshot shows a Postman interface with a GET request to `http://localhost:51051/api/v1/contents/1`. The response is in JSON format, containing details for content ID 1, including its name, remark, and a detailed value field with HTML formatting.

```
GET http://localhost:51051/api/v1/contents/1
```

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body: Cookies Headers (10) Test Results Status: 200 OK Time: 154 ms Size: 4.69 KB Save Response

```
1 {
2   "contentNo": 1,
3   "contentName": "소개",
4   "contentRemark": "소개 설명",
5   "contentValue": "<h3><strong>등장배경 및 목적</strong></h3><p>개발프레임워크는 정보시스템 개발을 위해 필요한 기능 및 아키텍처를 미리 만들어 제공함으로써 효율적인 어플리케이션 구축을 지원합니다. "전자정부 표준프레임워크"는 공공사업에 적용되는 개발프레임워크의 표준 정립으로 응용 SW 표준화, 품질 및 재 사용성 향상을 목표로 합니다. 이를 통해 "전자정부 서비스의 품질향상" 및 "정보화 투자 효율성 향상"을 달성하고, 대·중소기업이 동일한 개발기반 위에서 경쟁이 가능하게 됩니다. <br><br>※ 표준프레임워크는 기존 다양한 플랫폼(.NET, php 등) 환경을 대체하기 위한 표준은 아니며, java 기반의 정보시스템 구축에 활용하실 수 있는 개발·운영 표준 환경을 제공하기 위한 것입니다.</p><h3><strong>특징</strong></h3><p></p><h4>eGovFrame</h4><p><p>상용 솔루션 연계</p><p>민관학계로 구성 된 자문협의회를 통해 국가적 차원의 표준화 수행</p><p><p>국가적 표준화 지향</p><p>민·관·학계로 구성 된 자문협의회를 통해 국가적 차원의 표준화 수행</p><p><p>개발형 표준 준수</p><p>오픈소스 기반의 범용화되고 공개된 기술의 활용으로 특정 사업자에 대한 종속성 배제</p><p>변화 유연성</p><p>각 서비스의 모듈화로 교체가 용이하며 인터페이스 기반 연동으로 모듈간 변경영향 최소화</p><p>모바일 환경 지원</p><p>모바일 환경을 위한 모바일 웹(UX/UI) 및 하이브리드 앱 지원</p><p>편리하고 다양한 환경 제공</p><p>Eclipse 기반의 모델링(UML, ERD), 에디팅, 컴파일링, 디버깅 환경 제공</p><h3><strong>적용 가능 시스템 조건</strong></h3><p>아래 세가지 조건을 모두 만족하는 경우 표준프레임워크 적용 가능</p><ul><li>1. 자바 기반의 웹 응용 시스템(WAS가 존재하는 경우)</li><li>2. (2.7 기준) JavaEE(J2EE) <strong>JDK1.5 ~ 1.8</strong>의 환경 (단, 개발환경 2.7 이상에서는 JDK 1.6 필요)<br>(3.0 이상) JavaEE(J2EE) <strong>JDK1.6 ~ 1.8</strong>의 환경<br>(3.5 이상) JavaEE(J2EE) <strong>JDK1.7 ~ 1.8</strong>의 환경 (단, 개발환경 3.5.1 부터 JDK 1.8 적용 가능)<br>(3.6 이상) JavaEE(J2EE) <strong>JDK1.7 ~ 1.8</strong>의 환경<br>(3.7 이상) JavaEE(J2EE) <strong>JDK1.7 ~ 1.8</strong>의 환경 (단, 개발환경 3.7 이상에서는 JDK 1.8 필요)<br>(3.8 이상) JavaEE(J2EE) <strong>JDK1.7 ~ 1.8</strong>의 환경 (단, 개발환경 3.8 이상에서는 JDK 1.8 필요)<br>(3.9 이상) JavaEE(J2EE) <strong>JDK1.7 ~ 1.8</strong>의 환경 (단, 개발환경 3.9 이상에서는 JDK 1.8 필요)<br>(3.10 이상) JavaEE(J2EE) <strong>JDK1.7 ~ 1.8</strong>의 환경 (단, 개발환경 3.10 이상에서는 JDK 1.8 필요)</li><li>3. 신규 개발시스템으로써, 기존 시스템과 물리적 혹은 논리적으로 구분되는 경우</li></ul><p>실행환경 내 모바일 표준프레임워크의 사용자 경험(UX) 지원 기능은 프레임워크와 개발 언어 종류에 상관없이 활용가능 (javascript 기반)</p><h3><strong>적용 효과</strong></h3><p>정보시스템을 개발하거나 운영할 때 필요한 기본 기능을 미리 구현한 것으로 이를 기반으로 추가 기능을 개발하여 조립함으로써 전체 정보시스템을 완성할 수 있습니다.</p><p> 표준프레임워크 적용 전 : 1. 정보화사업별 동일한 기능들의 중복 개발, 2. 기술 종속으로 인해 선행사업자 의존도 높음, 3. 프레임워크 미 보유업체는 경쟁 불리, 4. 정보시스템간 상호 연계 시 많은 기간과 인력이 소요, 5. 개발표준 미흡으로 유지보수가 어려움, 표준프레임워크 적용 후 : 1. 공통컴포넌트 재사용으로 중복 예산 절감, 2. 표준화된 개발기반으로 사업자 종속성 해소, 3. 프레임워크 무상제공으로 중소기업 경쟁력 향상, 4. 표준화된 연계모듈 활용으로 상호운용성 향상, 5. 개발표준에 의한 모듈화로 유지보수가 용이">
```

LAB 1-3 API 호출 및 JUnit 테스트(6/8)

Step 1-3-07. Gateway 로 API 호출(portal-service 의 콘텐츠 상세 조회 API 호출)

- Postman 실행 > GET <http://localhost:8000/portal-service/api/v1/contents/1> 입력 > Send
- 콘텐츠 상세 조회 API는 모든 권한에 인가가 있어서 200 OK 상태를 응답하고 데이터가 정상적으로 조회된다.

The screenshot shows a Postman interface with a GET request to `http://localhost:8000/portal-service/api/v1/contents/1` that has been sent successfully. The response status is 200 OK, with a time of 490 ms and a size of 4.67 KB. The response body is displayed in JSON format:

```
1 {
2   "contentNo": 1,
3   "contentName": "소개",
4   "contentRemark": "소개 설명",
5   "contentValue": "<h3><strong>등장배경 및 목적</strong></h3><p>개발프레임워크는 정보시스템 개발을 위해 필요한 기능 및 아키텍처를 미리 만들어 제공함으로써 효율적인 어플리케이션 구축을 지원합니다. "전자정부 표준프레임워크"는 공공사업에 적용되는 개발프레임워크의 표준 정립으로 응용 SW 표준화, 품질 및 재 사용성 향상을 목표로 합니다. 이를 통해"전자정부 서비스의 품질향상" 및 "정보화 투자 효율성 향상"을 달성하고, 대·중소기업이 동일한 개발기반 위에서 공정 경쟁이 가능하게 됩니다. <br><br>※ 표준프레임워크는 기존 다양한 플랫폼(.NET, php 등) 환경을 대체하기 위한 표준은 아니며, java 기반의 정보시스템 구축에 활용하실 수 있는 개발·운영 표준 환경을 제공하기 위한 것입니다.</p><h3><strong>특징</strong></h3><p><h4>egovFrame</h4><p>상용 솔루션 연계</p><p>민관학계로 구성 된 전문협의회를 통해 국가적 차원의 표준화 수행</p><p>국가적 표준화 지향</p><p>민·관·학계로 구성 된 전문협의회를 통해 국가적 차원의 표준화 수행</p><p>개방형 표준 준수</p><p>오픈소스 기반의 범용화되고 공개된 기술의 활용으로 특정 사업자에 대한 종속성 배제</p><p>변화 유연성</p><p>각 서비스의 모듈화로 교체가 용이하며 인터페이스 기반 연동으로 모듈간 변경영향 최소화</p><p>모바일 환경 지원</p><p>모바일 환경을 위한 모바일 웹(UX/UI) 및 하이브리드 앱 지원</p><p>편리하고 다양한 환경 제공</p><p>Eclipse 기반의 모델링(UML, ERD), 에디팅, 컴파일링, 디버깅 환경 제공</p><h3><strong>적용 가능 시스템 조건</strong></h3><p>아래 세가지 조건을 모두 만족하는 경우 표준프레임워크 적용 가능</p><ul><li><i>1</i></li></ul>자바 기반의 웹 응용 시스템(WAS가 존재하는 경우)</li><li><i>2</i></li></ul><li><i>3</i></li></ul><li><i>4</i></li></ul><li><i>5</i></li></ul><p><strong>JDK1.5 ~ 1.8</strong>의 환경 (단, 개발환경 2.7 이상에서는 JDK 1.6 필요)<br><strong>(3.0 이상) JavaEE(J2EE) <strong>JDK1.6 ~ 1.8</strong>의 환경<br><strong>(3.5 이상) JavaEE(J2EE) <strong>JDK1.7 ~ 1.8</strong>의 환경 (단, 개발환경 3.5.1 부터 JDK 1.8 적용 가능)<br><strong>(3.6 이상) JavaEE(J2EE) <strong>JDK1.7 ~ 1.8</strong>의 환경<br><strong>(3.7 이상) JavaEE(J2EE) <strong>JDK1.7 ~ 1.8</strong>의 환경 (단, 개발환경 3.7 이상에서는 JDK 1.8 필요)<br><strong>(3.8 이상) JavaEE(J2EE) <strong>JDK1.7 ~ 1.8</strong>의 환경 (단, 개발환경 3.8 이상에서는 JDK 1.8 필요)<br><strong>(3.9 이상) JavaEE(J2EE) <strong>JDK1.7 ~ 1.8</strong>의 환경 (단, 개발환경 3.9 이상에서는 JDK 1.8 필요)<br><strong>(3.10 이상) JavaEE(J2EE) <strong>JDK1.7 ~ 1.8</strong>의 환경 (단, 개발환경 3.10 이상에서는 JDK 1.8 필요)</li></ul><p>신규 개발시스템으로써, 기존 시스템과 물리적 혹은 논리적으로 구분되는 경우</p><p>실환경 내 모바일 표준프레임워크의 사용자 경험(UX) 지원 기능은 프레임워크와 개발 언어 종류에 상관없이 활용가능 (javascript 기반)</p><h3><strong>적용 효과</strong></h3><p>정보시스템을 개발하거나 운영할 때 필요한 기본 기능을 미리 구현한 것으로 이를 기반으로 추가 기능을 개발하여 조립함으로써 전체 정보시스템을 완성할 수 있습니다.</p><p>표준프레임워크 적용 전 : 1.정보화사업별 동일한 기능들의 중복 개발, 2.기술 종속으로 인해 선행사업자 의존도 높음, 3.프레임워크 미 보유업체는 경쟁 불리, 4.정보시스템간 상호 연계 시 많은 시간과 인력인 소요, 5.개발표준 미흡으로 유지보수가 어려움, 표준프레임워크 적용 후 : 1. 공통컴포넌트 재사용으로 중복 예산 절감, 2.표준화된 개발기반으로 사업자 종속성 해소, 3.프레임워크 무상제공으로 중소기업 경쟁력 향상, 4.표준화된 연계모듈 활용으로 상호운용성 향상, 5.개발표준에 의한 모듈화로 유지보수가 용이"
```


LAB 1-3 API 호출 및 JUnit 테스트(7/8)

Step 1-3-08. Gateway 로 API 호출(portal-service 의 콘텐츠 목록 조회 API 호출)

- Postman 실행 > GET <http://localhost:8000/portal-service/api/v1/contents> 입력 > Send

- 콘텐츠 목록 조회 API는 ROLE_ANONYMOUS 권한에 인가가 없어서 401 Unauthorized 상태를 응답하고 데이터가 조회되지 않는다.

The screenshot shows the Postman interface for a GET request to `http://localhost:8000/portal-service/api/v1/contents`. The request is highlighted with a red box. Below the request bar, the 'Query Params' section is visible, showing a table with columns 'KEY', 'VALUE', and 'DESCRIPTION'. The table contains one row with 'Key' and 'Value'.

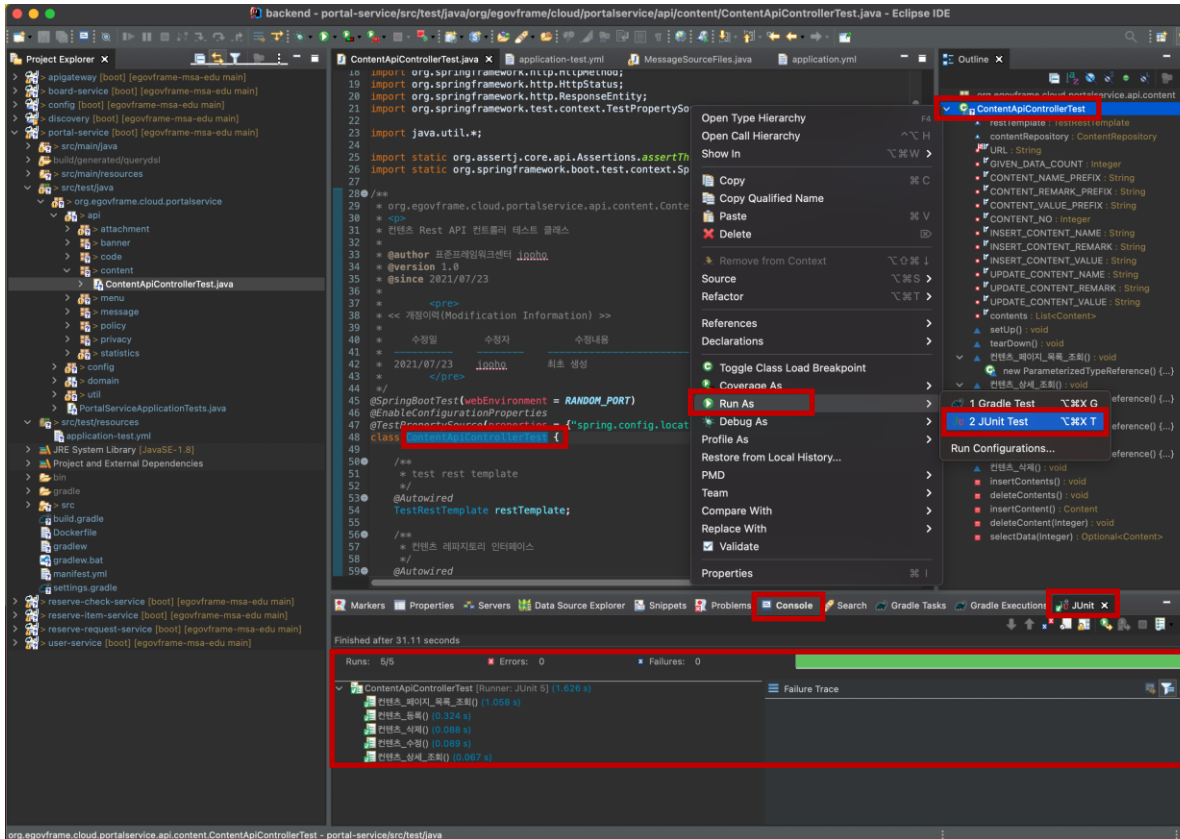
KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

At the bottom of the interface, the response status is shown as 'Status: 401 Unauthorized', which is also highlighted with a red box. Other response details include 'Time: 468 ms', 'Size: 238 B', and a 'Save Response' button. The response body is currently empty, and the 'Body' tab is selected.

LAB 1-3 API 호출 및 JUnit 테스트(8/8)

Step 1-3-09. JUnit 테스트

- Eclipse IDE > portal-service > /src/test/java/org.egovframe.cloud.portalservice.api.content.ContentApiControllerTest.java > 에디터 영역 또는 Outline의 클래스/메서드 우클릭 > Run As > JUnit Test
- Console 탭에서 로그, JUnit 탭에서 테스트 결과를 확인할 수 있다.
- 테스트는 H2 In-Memory Database 에서 수행되기 때문에 localhost:3306/msaportal 에 영향을 끼치지 않는다.



[첨부] 개발환경에서 프로젝트 생성

Step 1-2-00. 개발환경 Eclipse IDE 를 실행한다.

Step 1-2-01. Eclipse IDE > eGovFrame menu > Start > New MSA Boot Template Project 를 선택한다.

Step 1-2-02. 다이얼로그 창에서 Spring Cloud Config를 선택 후 Next 버튼을 클릭한다.

Step 1-2-03. Project name 에 config 로 입력 후 Finish 버튼을 클릭한다.

Step 1-2-04. 01 ~ 03 의 과정을 반복해서 아래의 프로젝트(소규모는 1~6, 대규모는 1~9)를 생성한다.

- | | |
|-------------------|----------------------------|
| 1. config | 6. board-service |
| 2. discovery | 7. reserve-check-service |
| 3. apigateway | 8. reserve-item-service |
| 4. user-service | 9. reserve-request-service |
| 5. portal-service | |

