**Secret**

보안포털 개발표준가이드(온라인서비스)

Ver. 1.1

|  |  |
| --- | --- |
|  | |
|  |  |
|  |  |
| Copyright ⓒ LG CNS  LG CNS의 사전 승인 없이 본 내용의 전부 또는 일부에 대한 복사, 배포, 사용을 금합니다. | |

개 정 이 력

| 버전 | 작성일 | 변경내용[[1]](#footnote-1) | 작성자 | 승인자 |
| --- | --- | --- | --- | --- |
| V1.0 |  | 최초작성 |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

목 차

[1. 개요 1](#_Toc176520776)

[2. 개발표준 1](#_Toc176520777)

[2.1 디렉토리 구조 1](#_Toc176520778)

[2.2 패키지 구조 1](#_Toc176520779)

[2.3 명명 규칙 2](#_Toc176520780)

[3. 개발가이드 2](#_Toc176520781)

[3.1 요소 별 개발가이드 2](#_Toc176520782)

[3.1.1 Controller 2](#_Toc176520783)

[3.1.2 Service 3](#_Toc176520784)

[3.1.3 ServiceImple 3](#_Toc176520785)

[3.1.4 SQL 4](#_Toc176520786)

[3.2 컴포넌트 별 개발가이드 6](#_Toc176520787)

[3.2.1 비동기 메시지 6](#_Toc176520788)

[3.2.2 Data Cache 14](#_Toc176520789)

[3.2.3 Transaction 처리 17](#_Toc176520790)

[4. 별첨 19](#_Toc176520791)

[4.1 Swagger 19](#_Toc176520792)

[4.1.1 swagger ui주소 19](#_Toc176520793)

[4.1.2 어노테이션 작성 20](#_Toc176520794)

# 개요

## 목적

본 문서는 보안 관련 포털(privacy center) 구축 프로젝트에 투입되는 개발자들에게 필요한 개발표준을 제공하기 위한 문서이다. 각 단계는 절차 중심으로 작성되어 개발자들이 쉽게 이해할 수 있도록 하였다.

## 적용범위

OpenJDK 17 + SpringBoot 3.2.2 + gradle + PostgreSQL 개발 환경에 대한 가이드이다.

# 로컬 개발 환경 세팅

## openJDK 17 설치

다음 경로에서 OpenJDK 17 버전을 다운받아 C드라이브에 압축 해제한다.

<https://download.java.net/java/GA/jdk17.0.2/dfd4a8d0985749f896bed50d7138ee7f/8/GPL/openjdk-17.0.2_windows-x64_bin.zip>

텍스트, 스크린샷, 번호, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

시스템 변수 생성

- 변수 이름: JAVA\_HOME

- 값: C:\jdk-17.0.2

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명

Path에 %JAVA\_HOME%\bin 추가

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명

## IntelliJ Community Edition 설치

자바 및 자바스크립트를 포함한 다양한 프로그래밍 언어를 위한 통합 개발 환경(IDE)이다. 공식 사이트에서 **Community Edition**을 다운받아 설치하도록 한다.

(다운로드 링크: <https://www.jetbrains.com/ko-kr/idea/download/?section=windows>)

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

## Git 설치

Git은 분산 버전 관리 시스템으로, 소스코드의 변경 이력을 관리하고 협업을 용이하게 한다.

공식 사이트에서 설치 프로그램을 다운받아 설치하도록 한다.

(다운로드 링크: <https://git-scm.com/downloads>)

## 프로젝트 구동

### 소스코드 다운로드

Repository 주소: https://github.com/LGU-CTO/nppm-svc.git

CMD 또는 git bash에서 다음 명령어 실행

- 명령어 실행: git clone https://github.com/LGU-CTO/nppm-svc.git

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

### IntelliJ 환경 세팅

IntelliJ 실행 후 다운로드 받은 프로젝트 폴더 오픈

ctrl + alt + shift + s 키를 눌러 Project Structure 화면을 활성화

Platform Settings > SDKs에서 JDK home path 경로 변경

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Project Settings > Project에서 SDK경로 변경, Language level ‘SDK default’ 선택

스크린샷, 텍스트, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Project Settings > Modules에서 Language level ‘Project default’ 선택

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

ctrl + alt + s 버튼을 눌러 Settings 창 활성화

Build, Execution, Deployment > Compiler > Java Compiler에서 Prjoect bytecode version ‘17’ 선택

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Build Tools > Gradle > Gradle JVM에서 JDK 17 선택

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

### Lombok 설치

ctrl + alt + s > plugin > Lombok 설치

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Build, Execution, Deployment > Annotation Processors > Enable annotation processing 체크

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

### 프로젝트 실행

상단 Current File 클릭 > Edit Configurations

텍스트, 소프트웨어, 스크린샷, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

좌측 상단 + 버튼을 눌러 Application 선택 후 Modify options > Add VM options 선택

소프트웨어, 텍스트, 스크린샷, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

아래와 같이 입력(-Dspring.profiles.active=local은 프로필 설정 부분) 후 Apply > OK

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

이후 시작 버튼 클릭으로 실행할 수 있다.

폰트, 스크린샷, 텍스트, 디자인이(가) 표시된 사진

자동 생성된 설명

# 개발표준

## 디렉토리 구조

Spring Boot 사용 시 기본적으로 구성되는 모듈 구조이다.

[Spring Boot]

| **폴더 구조** | | | **용도** |
| --- | --- | --- | --- |
| {프로젝트 모듈 경로} | /src/main/java | | java 소스 |
| /src/main/resources | application.yml | 어플리케이션 설정 파일(공통) |
| application-(배포환경).yml | 어플리케이션 설정 파일(배포 환경별) |
| log4j2-(배포환경).xml | 로그 설정 파일 |
| build.gradle | | gradle 파일 |

## 패키지 구조

패키지는 /src/main/java 하위에 생성하며, 영문 소문자를 사용하여 아래의 규칙에 따라 정의한다.

- level0: 보안관련포털 프로젝트를 지칭하는 com.lguplus.(업무코드-미정)를 공통으로 사용한다.

|  |
| --- |
| 예시)  privacy-center  ├─ src/main/java/com/lguplus/nppm \* default 패키지 (Level 0)  │ └─ online \* 온라인 개발 대상 (Level 2)  │ └─ config \* Config 클래스 패키지  │ ├─ controller \* Controller 클래스 패키지  │ ├─ service \* Service 구현 클래스 패키지  │ ├─ entity \* Entity 클래스 패키지  │ ├─ repository \* Repository 클래스 패키지  │ └─ dto \* DTO 클래스 패키지 |

## 명명 규칙

모든 Class와 Interface명은 표준 단어를 조합하여 사용하며, Pascal 표기법을 따른다.

| **구분** | **컴포넌트 구분** | **클래스명** | **컴포넌트 Postfix** |
| --- | --- | --- | --- |
| java | Config | 컴포넌트명 + Config | XxxConfig |
| Controller | 컴포넌트명 + Controller | XxxController |
| Service | 컴포넌트명 + Service | XxxService |
| ServiceImpl | 컴포넌트명 + Service + Impl | XxxServiceImpl |
| Repository | 컴포넌트명 + Repository | XxxRepository |
| Entity | 컴포넌트 내 의미있는 명 + Entity | XxxEntity |
| DTO | 데이터객체 명사조합 + DTO | XxxDTO |

# 개발가이드

## 요소 별 개발가이드

### Controller

클라이언트Request 요청을 받아들이고 응답을 반환한다.

Controller 클래스는 Spring에서 Bean으로 관리되는 대상으로 Singleton으로 동작되기 때문에 멀티스레드 구조 상 Method 밖에 멤버변수는 반드시 상태값을 갖고 있지 않는 무상태(Stateless)로 개발되어야 한다. (Primitive type, DTO등 사용 금지)

|  |
| --- |
| @RestController  @CrossOrigin(origins = "\*", allowedHeaders = "\*")  @RequestMapping(value = "/example")  @RequiredArgsConstructor  @Tag(name = "ExampleController", description = "Example Controller")  public class ExampleController {  private final ExampleService exampleService;    @GetMapping("/member")  @Operation(summary = "멤버 조회", description = "멤버를 조회한다")  public ResponseEntity<List<ExampleDTO>> getAllEmployees() {  List<ExampleDTO> employees = exampleService.getAllEmployees();  log.info("get!");  return ResponseEntity.ok(employees);  }  } |

- 클래스 상단에 @RestController를 선언한다.

- 비즈니스 로직 처리를 위한 Service Bean객체를 생성자 방식으로 주입받는다. (@Autowired방식 지양)

- 각 메소드에 상단에 CRUD 동작에 따라 @GetMapping, @PostMapping, @PutMapping, @DeleteMapping 어노테이션을 선언한다.

### Service

Interface로 비즈니스 로직을 선언하며, ServiceImple에서 이를 상속받아 비즈니스 로직을 구현한다.

|  |
| --- |
| public interface ExampleService {  List<ExampleDTO> getAllEmployees();  } |

- 인터페이스 내부에 추상 method를 선언한다.

### ServiceImple

Service에서 추상 method를 상속받아 비즈니스 로직을 구현하며 결과를 Controller에 반환한다. 또한, 다른 서비스 API를 호출하거나 이벤트 기반의 메시지 발생 및 메시지 수신에 대한 비즈니스 로직 처리도 Service에서 구현한다.

Service 클래스는 Spring에서 Bean으로 관리되는 대상으로 Singleton으로 동작되기 때문에 멀티스레드 구조 상 Method 밖에 멤버변수는 반드시 상태값을 갖고 있지 않는 무상태(Stateless)로 개발되어야 한다. (Primitive type, DTO등 사용 금지)

|  |
| --- |
| @Service  @Transactional  @RequiredArgsConstructor  public class ExampleServiceImpl implements ExampleService {  private final ExampleRepository exampleRepository;  @Override  public List<ExampleDTO> getAllEmployees() {  return exampleRepository.findAll().stream().map(this::convertToDTO).collect(Collectors.toList());  }  @Override  public ExampleDTO convertToDTO(ExampleEntity employee) {  ExampleDTO employeeDTO = ExampleDTO.builder()  .id(employee.getId())  .name(employee.getName())  .position(employee.getPosition())  .build();  return employeeDTO;  }  } |

- 클래스 상단에 @Service를 선언한다.

- 클래스 전체에 메소드 별 트랜잭션 처리를 위해 @Transactional를 선언한다.

- Data Access를 위해 Repository Bean 객체를 생성자 방식으로 주입받는다. (@Autowired방식 지양)

### DTO

DB에서 데이터를 얻어 Service나 Controller에 데이터를 전송하는데 사용되는 전송용 객체이다.

순수 데이터 전달만을 위한 객체이기 때문에 getter와 setter method 만을 가지며, 로직을 지니지 않는다(별도의 method 작성 지양).

|  |
| --- |
| @Getter  @Setter  @Builder  @NoArgsConstructor  @AllArgsConstructor  public class ExampleDTO {  long id;  String name;  String position;  } |

- 클래스 상단에 @Getter, @Setter를 선언한다.

- Builder Pattern 사용 시 @Builder, @NoArgsConstructor, @AllArgsConstructor를 선언한다(선택사항).

### Entity

실제 DB테이블과 mapping되는 객체로, 테이블이 가지지 않는 컬럼을 필드로 가져서는 안 된다.

객체의 불변성을 보장해야 하기 때문에 setter method를 지양한다(생성자 또는 Builder 사용). DTO와 달리 비즈니스 로직을 포함할 수 있다.

|  |
| --- |
| @Entity  @Table(name = "employees")  @Getter  @Builder  @NoArgsConstructor  @AllArgsConstructor  public class ExampleEntity {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private Long id;  private String name;  private String position;  private Double salary;  private String hireDate;  } |

- 클래스 상단에 @Entity를 선언한다.

- @Table을 선언하고, name 속성에 mapping할 테이블명을 지정한다. uniqueConstraintse등의 옵션을 추가하여 유니크 제약조건을 생성할 수 있다.

- @Getter를 선언한다.

- Builder Pattern 사용 시 @Builder, @NoArgsConstructor, @AllArgsConstructor를 선언한다(선택사항).

- DB 테이블과 mapping되는 컬럼을 필드로 선언한다(자세한 내용은 하단 JPA 항목 참조).

### Repository

DB Layer 접근자로, query method를 정의한다. JPA Repository를 상속받으며, JPA Repository는 Spring Data JPA에서 제공하는 인터페이스로, 데이터베이스와 상호작용하는데 필요한 CRUD 메소드를 정의하고 있다.

|  |
| --- |
| @Repository  public interface ExampleRepository extends JpaRepository<ExampleEntity, Long>{  List<ExampleEntity> findByPosition(String position);  } |

- 클래스 상단에 @Repository를 선언한다.

- JpaRepsitory <Entity 클래스, PK 타입>을 상속받는다.

- query method를 정의한다(자세한 내용은 하단 JPA 항목 참조).

### JPA

JPA(Java Persistacne API)는 관계형 데이터베이스를 관리하고 조작하기 위한 표준 API이다. JPA는 객체-관계 mapping(ORM)을 통해 자바 객체와 데이터베이스 테이블 간의 mapping을 정의하고 관리한다. 개발자는 SQL 쿼리를 직접 작성하지 않으며, 쿼리 method를 정의하거나, JPA Repository에서 제공하는 CRUD method를 사용하여 DB에 접근할 수 있다. Entity 클래스(DB 테이블에 mapping)와 Repository(DB에 접근하여 CRUD 수행) 클래스를 생성한다. JPA 기본 사용법에 대해서는 아래 사이트를 참조한다(가이드 추가 예정).

- JPA 기본 사용법: <https://m.blog.naver.com/hj_kim97/222780110215>

- Entity 클래스는 클래스 상단에 @Entity와 @Talbe 어노테이션을 선언하고, 필드에는 @ID 또는 @Column등을 선언하여 PK와 컬럼을 mapping한다. 추가 옵션과 생략 가능한 옵션 등에 대해서는 아래 사이트를 참조한다(가이드 추가 예정).

- 어노테이션 정보1: <https://www.digitalocean.com/community/tutorials/jpa-hibernate-annotations>

- 어노테이션 정보2: <https://seungh1024.tistory.com/78>

- Repository 클래스는 클래스 상단에 @Repository를 선언하고, JpaRepository를 상속받는다. JpaRepository를 상속 받을때에는 <Entity 클래스, PK 타입>을 같이 명시해 주어야 한다. 만약 Entity의 PK가 두 개 이상일 경우, 이를 복합키(Composite Key)라고 하며 @IdClass 또는 @EmbeddedId를 사용하여 복합키를 구현한 후, 해당 키 클래스를 사용하여 JpaRepository 인터페이스를 확장한다. 복합키 구현은 아래 사이트를 참조한다(가이드 추가 예정).

- 복합키 사용: <https://syk531.tistory.com/94>

- 또한 Repository 클래스 내부에는 Query method를 선언하여 사용하거나 JpaRepository에서 제공하는 CRUD 메소드를 사용할 수 있다. Jpa에서 제공하는 기본 메소드와 Query method 작성 방법에 대해서는 아래 사이트를 참조한다(가이드 추가 예정).

- 기본 메소드: <https://docs.spring.io/spring-data/data-jpa/docs/current/api/org/springframework/data/jpa/repository/support/SimpleJpaRepository.html>

- Query method 작성: <https://docs.spring.io/spring-data/jpa/reference/jpa/query-methods.html>

# Amazon Q developer

Amazon Q developer는 생성형 인공지능(AI) 기반 대화형 어시스턴트로, AWS 아키텍처, AWS 리소스, 모범 사례, 문서, 지원 등에 대해 질문할 수 있다.

특히 IntelliJ와 같은 통합 개발 환경(IDE)에서 사용될 때, Amazon Q는 코드에 대한 채팅, 인라인 코드 완성, 새로운 코드 생성, 코드 업그레이드 및 개선을 수행할 수 있다.

## IntelliJ 용 Amazon Q 플러그인 설치

공식 사이트에서 플러그인 최신 버전을 다운받는다.

다운로드 링크: <https://plugins.jetbrains.com/plugin/24267-amazon-q/versions#tabs>

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

Intellij Settings > Plugins > 톱니바퀴 모양 클릭 > Install Plugin from Disk 선택 후 다운받은 파일 선택하여 설치

스크린샷, 멀티미디어 소프트웨어, 소프트웨어, 그래픽 소프트웨어이(가) 표시된 사진

자동 생성된 설명

설치 후 IDE 재시작

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

사용자 인증 후 사용

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

# 별첨

## Swagger

### swagger ui주소: 도메인명/swagger-ui/index.html (ex. http://localhost:8080/swagger-ui/index.html)

### API 그룹 추가

SwaggerConfig.java

|  |
| --- |
| @Configuration  @OpenAPIDefinition(  info = @Info(title = "LG Uplus Privacy Center",  description = "LG Uplus용 보안 포털용 API 명세",  version = "v1")  )  public class SwaggerConfig {  @Bean  GroupedOpenApi getExampleApi() {  String[] paths = {"/example/\*\*"};    return GroupedOpenApi  .builder()  .group("Examples")  .pathsToMatch(paths)  .build();  }  } |

GroupedOpenApi get + 서비스명 + Api()를 정의하여 API 그룹을 묶을 수 있다.

- paths: 그룹을 묶을 path를 지정한다.

- group(그룹명): 그룹 명을 지정한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Controller

|  |
| --- |
| @GetMapping("/member")  @Operation(summary = "멤버 조회", description = "멤버를 조회한다")  public ResponseEntity<List<ExampleDTO>> getAllEmployees() {    List<ExampleDTO> employees = exampleService.getAllEmployees();    return ResponseEntity.ok(employees);  } |

@Operation : API 상세 정보를 기술한다.

summary: API에 대한 간략 설명

description: API에 대한 상세 설명

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

## Actuator

Spring Boot Actuator를 적용하여 Actuator에서 제공하는 endpoint를 통해 애플리케이션의 상태를 확인하고 다양한 정보를 확인할 수 있다.

다양한 endpoint가 있지만, 기본값으로 health endpoint만 HTTP와 JMX를 통해 노출되도록 설정되어 있다. 이를 통해 별도의 health check API를 개발하지 않아도 애플리케이션의 health check를 수행할 수 있다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

참고 사이트: <https://docs.spring.io/spring-boot/docs/3.2.2/reference/html/actuator.html#actuator.endpoints>

1. 변경 내용: 변경이 발생되는 위치와 변경 내용을 자세히 기록(장/절과 변경 내용을 기술한다.) [↑](#footnote-ref-1)