|  |
| --- |
| **Spring Boot 정리** |

**[애너테이션]**

자바의 애너테이션(annotation)이란 자바의 클래스, 메서드, 변수 등에 정보를 부여하여 부가 동작을 할 수 있게 하는 목적으로 사용된다.

**[Controller]**

- 사용자 요청을 받고 알맞은 로직을 수행하고 결과를 반환하는 클래스이다.  
- Django에서는 urls.py 파일과 views.py 파일을 따로 나눴다면  
 Spring에서는 Controller 하나에서 모두 처리함.

**[build.gradle]**

- 프로젝트의 빌드, 생성, 테스트, 의존성 관리, 배포 등을 정의하는 곳.  
- 프로젝트를 어떻게 빌드하고 실행할지, 어떤 라이브러리를 쓸지, 어떤 설정을 적용할지 알려주는 파일.  
- Django에서 requirements.txt + manage.py + settings.py = build.gradle

**[롬복(Lombok)]**

- 롬복(Lombok) 라이브러리는 소스 코드를 작성할 때 자바 클래스에 애너테이션을 사용하여 자주 쓰는 Getter 메서드, Setter 메서드, 생성자 등을 자동으로 만들어주는 도구.

**[Spring Boot 프로젝트 구조 이해하기]**

- src/main/java 디렉터리: 자바 파일을 저장하는 공간  
- com.mysite.sbb 패키지: 스프링 부트의 컨트롤러, 폼과 DTO, 데이터베이스 처리를 위한 엔티티, 서비스 등의 자바 파일이 이 곳에 위치한다.  
- SbbApplication.java 파일: 프로그램 시작을 담당하는 파일.   
 프로젝트명 + Application.java 파일. 스프링 부트 프로젝트를 생성할 때 자동 생성됨.  
 @SpringBootAppliction 애너테이션이 반드시 적용되어 있어야 함.  
 애너테이션을 통해 스프링 부트 애플리케이션을 시작할 수 있음.

- src/main/resources 디렉터리: 자바 파일을 제외한 HTML, CSS, JS, 환경 파일 등을 저장하는 공간.  
- templates 디렉터리: 템플릿 파일을 저장. HTML 형식의 파일  
- static 디렉터리: CSS, JS, Img파일 등을 저장.  
- application.properties 파일: 프로젝트의 환경을 설정하는 파일.  
 프로젝트 환경 변수, 데이터베이스 등의 설정을 이 파일에 저장

- src/test/java 디렉터리: 프로젝트에서 작성한 파일을 테스트하는 코드를 저장하는 공간.  
Junit과 스프링 부트의 테스트 도구를 사용하여 서버를 실행하지 않은 상태에서 src/main/java 디렉터리에 작성한 코드를 테스트할 수 있다.

- build.gradle 파일: Gradle이 사용하는 환경 파일. Gradle은 Groovy를 기반으로 한 빌드 도구로 Ant, Maven과 같은 이전 세대의 단점을 보완하고 장점을 취합하여 만들었다. Build.gradle 파일에는 프로젝트에 필요한 플러그인과 라이브러리를 설치하기 위한   
내용을 작성한다.

**[URL 매핑과 컨트롤러 이해하기]**

- 컨트롤러의 메서드에 @GetMapping 또는 @PostMapping과 같은 애너테이션을 적용하면 해당 URL과 메서드가 실행된다.

- @GetMapping 애너테이션은 요청된 URL과의 매핑을 담당.

- @ResponseBody 애너테이션은 URL 요청에 대한 응답으로 문자열을 리턴하라는 의미로 쓰였다. 만약 @ResponseBody 애너테이션을 생략한다면 스프링 부트는 ‘index’라는 이름의 템플릿을 찾게 된다.

**[ORM과 JPA 이해하기]**

- ORM: SQL을 사용하지 않고 데이터베이스를 관리할 수 있는 도구.  
 ORM은 데이터베이스의 테이블을 자바 클래스로 만들어 관리할 수 있다.  
 ORM의 자바 클래스를 엔티티(entity)라고 한다.   
 엔티티는 데이터베이스의 테이블과 매핑되는 자바 클래스를 말한다.

- JPA: 스프링 부트는 JPA(Java Persistence API)를 사용하여 데이터베이스를 관리함.  
 스프링 부트는 JPA를 ORM 기술의 표준으로 사용함.  
 JPA는 인터페이스 모음으로 이 인터페이스를 구현한 실제 클래스가 필요함.  
 JPA를 구현한 실제 클래스에는 대표적으로 하이버네이트(Hibernate)가 있음.  
 하이버네이트는 JPA의 인터페이스를 구현한 실제 클래스이자  
 자바의 ORM 프레임워크로 스프링 부트에서 데이터베이스를 관리하기 쉽게 도움.  
 인터페이스(interface)란 클래스가 구현해야 하는 메서드 목록을 정의한 틀.

**[엔티티로 테이블 매핑하기]**

- 엔티티를 모델 또는 도메인 모델이라고도 한다.  
테이블과 매핑되는 클래스를 모두 엔티티라고 지칭함. DB 테이블 정의  
예: Question 엔티티, Answer 엔티티 장고에서는 models.py  
Entity = 데이터 구조(스키마) | Repository = 데이터 조작 도구(쿼리 실행)

**[리포지토리(Repository) 생성하기]**

- 엔티티가 데이터베이스 테이블을 생성했다면 리포지토리는 이와 같이 생성된   
데이터베이스 테이블의 데이터들을 저장, 조회, 수정, 삭제 등을 할 수 있도록   
도와주는 인터페이스임. 이때 리포지토리는 테이블에 접근하고 데이터를 관리하는  
메서드를 제공.

**[Junit]**

- 테스트 코드를 작성하고 작성한 테스트 코드를 실행할 때 사용하는   
자바의 테스트 프레임워크. Junit은 리포지토리뿐만 아니라 소프트웨어 개발 시 테스트 작업을 수행할 때 많이 사용.

**[도메인별로 분류]**

- 도메인은 “질문”, “답변”처럼 프로젝트의 주요 기능을 뜻하는 말이고 이러한 주요 기능들을 패키지를 활용하여 자바 파일을 도메인별로 분류한다.  
예: question패키지(게시판의 질문과 관련된 자바 파일 모음),  
 answer패키지(게시판의 답변과 관련된 자바 파일 모음),  
 user패키지(사용자와 관련된 자바 파일 모음)

**[서비스(Service)]**

- 스프링에서 데이터 처리를 위해 작성하는 클래스이다.  
서비스를 이용하면 모듈화가 가능.  
리포지토리 객체는 @ReequiredArgsConstructor에 의해 생성자 방식으로 주입.  
서비스 객체도 @RequiredArgsConstructor에 의해 생성자 방식으로 주입.

**[Spring Boot 웹 동작 아키텍처]**

[클라이언트 브라우저 / 앱]   
▼ (HTTP 요청)

[Controller] ← (DTO/Form으로 데이터 수신)  
▼

[Service] ← (비즈니스 로직 처리, DTO ↔ Entity 변환)  
▼

[Repository] ← (DB CRUD)  
▼

[Database (MySQL, PostgreSQL, etc.)]

**[Controller]** 예: 창구 직원 (신청서 받아서 처리 흐름 연결)

- 역할: 클라이언트(브라우저/앱)로부터 HTTP 요청을 받고 응답을 반환.

- 기능: URL 라우팅(@GetMapping, @PostMapping 등), 요청 데이터 수신(DTO/Form),  
 Service 호출, 처리 결과를 View/JSON 형태로 응답  
 요청 데이터를 Form/DTO로 받음 → Service 호출

**[Servicee]** 예: 회사의 정책/규칙을 집행하는 부서

- 역할: 비즈니스 로직을 담당하는 계층

- 기능: 데이터 검증, 여러 Repository를 조합한 로직 처리, Entity ↔ DTO 변환,  
트랜잭션 관리(@Transaction)  
Form/DTO → Entity 변환, 비즈니스 로직 실행, Repository 호출

**[Repository]** 예: DB 관리 부서

- 역할: DB와 직접 소통하는 데이터 접근 계층.

- 기능: CRUD 수행, JPA/Hibernate를 통해 SQL 실행, Entity를 DB에 저장/조회

**[Entity]** 예: 실제 회사 DB 기록부

- 역할: DB 테이블과 매핑되는 도메인 객체

- 기능: JPA에서 관리하는 영속 객체, @Entity, @Id, @Column 등으로 DB 필드와 매핑

**[DTO(Data Transfer Object)]** 예: 사무원이 상부에 보고하는 보고서

- 역할: 계층 간 데이터 전송용 객체 (특히 Controller ↔ Service ↔ View)

- 기능: Entity를 그대로 노출하지 않고 필요한 데이터만 전달(응답),  
API 응답이나 요청 데이터 구조 정의, Entity 데이터를 가공해서 외부로 전달

**[Form(입력용 객체)]** 예: 고객이 작성하는 신청서

- 역할: 사용자 입력을 받기 위한 객체 (특히 HTML Form 바인딩)

- 기능: 유효성 검증(@NotNull, @Size 등), Controller에서 요청 파라미터 매핑  
외부 입력을 받는 객체(요청)

**[URL 프리픽스]**

- 프리픽스(prefix)란 URL의 접두사 또는 시작 부분을 가리키는 말로 @RequestMappin 애너테이션을 사용하여 URL 매핑은 항상 애너테이션으로 지정한 프리픽스로  
시작하도록 설정.

**[스프링 시큐리티]**

- 스프링 시큐리티를 사용하여 회원가입과 로그인 기능을 만들어서 사용 가능.  
스프링 시큐리티는 스프링 기반 웹 애플리케이션의 인증과 권한을 담당하는   
스프링 하위 프레임워크이다.  
인증은 로그인과 같은 사용자의 신원을 확인하는 프로세스를 관리하는 것.  
권한은 인증된 사용자가 어떤 일을 할 수 있을지(접근 권한) 관리하는 것.

**[빈(bean)]**

- 스프링에 의해 생성 또는 관리되는 객체. @Bean 애너테이션을 통해 자바 코드 내에서 별도로 빈을 정의하고 등록할 수 있음.

**[CSRF]**

- CSRF는 웹 보안 공격 중 하나로 조작된 정보로 웹 사이트사 실행되도록 속이는 공격 기술. 스프링 시큐리티는 이러한 공격을 방지하기 위해 CSRF 토큰을 세션을   
통해 발행하고 웹 페이지에서는 폼 전송 시에 해당 토큰을 함께 전송하여 실제 웹 페이지에서 작성한 데이터가 전달되는지를 검증함.

- 토큰이란 요청을 식별하고 검증하는데 사용하는 특수한 문자열 또는 값을 의미.

- 세션이란 사용자의 상태를 유지하고 관리하는데 사용하는 기능.

**[스프링 부트 개발 플로우]**

1. 데이터베이스 설정 (application.properties 파일에서 DB 설정)

2. Entity 생성 (장고로 비교하면 models.py 정의, DB 테이블 생성)

3. Repository 작성 (JPA/Hibernate를 통해 SQL 실행하여 Entity를 DB에 저장/조회)

4. Service 작성 (데이터/비즈니스 로직 처리, 여러 리포지토리를 결합한 로직 처리, )

5. Controller 작성 (장고로 비교하면 urls.py+views.py, URL 매핑+데이터 수신/응답)

6. 템플릿(Templates)/View 작성 (HTML 템플릿 파일 저장, Thymeleaf, JS 등)

7. 웹 애플리케이션 실행 (mvnspring-boot:run | 브라우저에서 <http://localhost:8080> 접속)