프레임워크와 라이브러리의 차이

- 프레임워크 : 프로그램작성시 전체적인 뼈대(구조)를 제공. (필수코드, 알고리즘, 보안, DB연동 기능 등)
- 라이브러리 : 특정한 기능에 대한 API(도구/함수)를 모아놓은 친구.

좋은 프레임워크란

- 재사용성이 좋은 것
- 확장성이 좋은것
- 사용하기 쉬운 것
- 플랫폼으로 부터의 독립성
- 지속적인 지원

Spring Framework

- JAVA 플랫폼을 위한 오픈소스 어플리케이션 프레임워크로 동적인 웹사이트를 개발하기 위한 여러 서비스를 제공
- Spring Framework 프로젝트 종류
    - MVC : 웹 어플리케이션 개발을 위한 프레임워크
    - Security : 어플의 인증 / 권한인가 같은 보안 기능을 쉽게 구현 가능
    - Data : NoSQL, Key-Value 등의 데이터 저장소에 쉽게 접근 가능
    - Batch : 스프링 배치 어플을 개발하기 위한 경량 프레임워크
    - Intergration : 시스템 연계 어플을 개발하기 위한 프레임워크 (AMQP, File, FTP, REST, JDBC 등)
    - Cloud : 클라우드 환경에 최적화된 어플을 개발히기 위한 프레임워크 (Cloud config, cloud netflix 등)
    - STS : 스프링앱 개발을 위한 이클립스 기반 통합 개발환경
    - IO Platform : 라이브러리(스프링 및 서드파티)들의 버전을 결정하고 의존성을 해결
    - Boot : 최소한의 설정으로 스프링 어플리케이션을 만들 수 있게 해줌
- 전자정부 표준 프리엠워크
    - 국내 공공사업에 적용되는 개발 프레임워크
    - 응용 SW 표준화, 품질 및 재사용성 향상을 목표
    - 2009년 1.0, 2021년 기준 3.10.0 (Spring Framework 4.3.25 기반)

Spring Boot

- 단독으로 실행 가능한 스프링 어플리케이션을 생성
- 스프링을 쉽게 사용할 수 있도록 필요한 설정 대부분을 미리 세팅해 놓음
- 통계, 상태체크, 외부설정 등을 제공
- 설정을 위해 XML코드를 요구하지 않음
- Tomcat, jetty, undertow 웹서버 내장
- Jar, War 파일로 배포 가능

MVC 모델

사용자 인터페이스와 비즈니스 로직을 분리하여 개발하는 방식으로, 서로 영향을 최소화하여 개발 및 변경이 쉬운 어플리케이션을 만들 수 있음.

- Model : 어플리케이션의 정보(데이터) Database
- View : 사용자에게 보여질 화면 (Html)
- Controller : 모델과 뷰의 중계역할 (html과 데이터의 결합)
    - 사용자가 웹브라우저를 통해 어떠한 요청을 하면 그 요청을 처리할 컨트롤러를 호출하게 됨.
    - 컨트롤러는 사용자 요청을 처리하기 위한 비즈니스 로직을 호출하고 그 결과를 사용자에게 전달함.
- Service : 사용자의 요청 처리를 위한 비즈니스 로직이 수행됨.
- DAO : (DataAcessObject) 데이터베이스에 접속해 비즈니스 로직 실행에 필요한 쿼리를 호출함.
- MVC 모델 호출 순서
    - Controller -> View(html) -> Controller -> Service -> Mapper(QUERY)

Spring Boot 배치

- 백엔드 단의 배치 처리 프로그램으로 대용량 데이터 처리에 최적화되어 고성능 / 자동화
- 효과적인 로깅, 통계처리, 트랜잭션 관리 / 예외사항과 비정상 동작에 대한 방어기능

Annotation

- @를 이용한 주석으로, 자바코드에 주석을 달아 특별한 의미를 부여한 것
    - 메타데이터라고도 불리며 JDK5부터 사용
    - 컴파일러가 특정 오류를 억제하도록 지시하는 것과 같이 프로그램 코드의 일부가 아닌 프로그램에 관한 데이터를 제공, 코드에 정보를 추가하는 정형화된 방법.
- 등장배경
    - 프로그램의 규모가 방대해지면서 XML이 가지는 설정정보의 양이 많아짐.
    - Annotation을 이용하면 직관적으로 메타데이터 설정이 가능함.
        - Model 클래스에 직접명시함으로써 해당 데이터들에 유효성을 쉽게 파악할 수 있으며, 코드양이 줄어듬.

Thymeleaf

- HTML, XML, JS, CSS, TEXT 를 처리할 수 있는 Java 템플릿 엔진
- html파일을 가져와 파싱/분석하여 정해진 위치에 데이터를 치환해 웹페이지를 생성함.
- JSP(Java Server Pages)파일에 비즈니스 로직을 넣으면 디버깅 및 유지보수가 힘들어져 Java코드를 잘 넣지 않음.
- Layout 기능 제공

Loombok

- VO(Value Object) : 사용되는 값이 객체로 표현됨
- DTO(Data Transfer Object) : 데이터 전송을 위한 객체로, 비즈니스 로직까지 담아 사용
- Annotation 설정으로 Setter/Getter, toString, equals, hashcode 함수 자동생성

MyBatis

- 복잡한 쿼리나 다이나믹 쿼리의 적용
- 프로그램 코드와 SQL 쿼리를 분리하여 코드의 간결성 및 유지보수성을 향상시킴
- resultType, resultClass 등 VO를 사용하지 않고, 조회결과를 사용자 정의 DTO, MAP 등으로 맵핑
- 빠른 개발이 가능하여 생산성 향상

Logback

- log4j보다 약 10배 정도 빠르게 수행됨 (메모리 효율성 향상)
- 설정파일을 변경하였을 때, 서버 재가동 없이 변경내용 적용
- RollingFileAppender를 사용하면 자동으로 오래된 로그를 삭제하고 Rolling 백업 처리
- 로그메세지를 레벨별로 나누어 제어 가능
    1. Error : 일반 에러 (상용서비스 중일때)
    2. Warn : 에러는 아니지만 주의할 필요가 있을 때
    3. Info : 일반 정보를 나타냄
    4. Debug : 일반 정보를 상세히 나타냄 (개발 중일때)
    5. Trace : 경로 추적을 위해 사용

인터셉터 사용

- 필터는 디스패처 서블릿 앞 단에서 동작
- 인터셉터는 디스패처 서블릿에서 핸들러 컨트롤러로 가기 전에 동작
- 인터셉터에서는 스프링 빈을 사용할 수 있음
- 인터셉터 메서드
    - preHandle : 컨트롤러 실행전에 수행
    - postHandle : 컨트롤러 수행 후 결과를 뷰로 보내기 전 에 수행
    - afterCompletion : 뷰의 작업까지 완료 후 수행

AOP(관점지향 프로그래밍)

실행 시점을 지정할 수 있는 애노테이션

- @Before (이전) : 어드바이스 타겟 메소드가 호출되기 전에 어드바이스 기능을 수행
- @After (이후) : 타겟 메소드의 결과에 관계없이 타겟 메소드가 완료 되면 어드바이스 기능 수행
- @AfterReturning (정상적 반환)타겟 메소드가 성공적으로 결과반환 후, 어드바이스 기능 수행
- @AfterThrowing (예외 발생 이후) : 타겟 메소드가 수행 중 예외를 던지면 어드바이스 기능 수행
- @Around (메소드 실행 전후) : 어드바이스가 타겟 메소드를 감싸서 타겟 메소드 호출전과 후에 어드바이스 기능을 수행