IntelliJ에서 Spring Boot, Gradle, Mybatis프로젝트 생성 및 설정

-Intellij-

File -> New -> Project

Spring Initializr로 설정 가능

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

자동 생성된 설명

SDK와 JAVA, Spring Boot 버전 주의

Spring Boot 1.3이후로 jsp를 이용해서 view를 만드려면 war로 설정해야함

Spring Boot 2.7.11, SDK 15.0.2, Gradel, Mybatis-Spring-Boot-Starter 2.3.0 사용

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

자동 생성된 설명

Spring Boot 3.0.0이상은 JAVA 17이상부터 지원

Dependency 사전 설정 가능

스크린샷, 소프트웨어, 멀티미디어 소프트웨어, 텍스트이(가) 표시된 사진

자동 생성된 설명

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

자동 생성된 설명

프로젝트 생성 후 **build.gradle** 에서 dependency 확인, 추가

build.gradle 수정 후 우측 상단의 Load Gradle Changes 실행

**Lombok?**

코드 작성 편의를 위한 라이브러리

@Getter, @Setter, @AllArgsConstructor, @NoArgsConstructor, @RequiredArgsConstructor, @Data, @ToString 등 어노테이션들 제공

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

자동 생성된 설명

Build.gradle dependency 설정 후

Resources/**application.properties** 또는 **application.yml**에서 필요한 내용 설정

(Datasource bean생성 위한 DB정보, jsp view 위한 suffix, prefix, mybatis xml파일 경로 설정)

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

자동 생성된 설명

Resources/mybatis 경로에 **MybatisConfig.xml** 생성 및 필요한 내용 작성

\*앞서 application.properties 또는 application.yml에 mybatis: mapper-location을 작성했다면 MybatisConfig.xml에서 mapper경로 설정 X – 중복 에러 뜸

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

자동 생성된 설명

main/java/package경로에 config/**MybatisConfig.java** 작성

Mybatis-Spring Boot에서는 mybatis-spring-boot-autoconfigure에서 자동으로

sqlSessionFactory, sqlSessionTemplate, dataSource가 자동으로 Bean에 등록되서 필요 X

**Spring Bean?**

Spring IoC 컨테이너가 관리하는 자바 객체

Spring 컨테이너에 의해 생성되고 관리되는 자바 객체

@Component 어노테이션 또는 Config에서 @Bean 어노테이션 사용

Bean 객체를 주입받기 위해서 @Autowired 어노테이션 사용

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

자동 생성된 설명

main/java/package경로에 controller/**Controller.java** 작성

요청과 응답을 관리

Client의 요청을 받아 Service의 기능을 호출, 적절한 응답을 VO (DTO) 또는 Map형태로 변환

Controller는 Service를 @Autowired

@Controller, @RequestMapping() Annotation 사용

Class의 @RequestMapping() url과 메소드 @RequestMapping() url을 접속 경로(localhost:8080) 뒤에 붙여서 요청하면 해당 메소드 동작

**@RequestParam**

1개의 http 요청 파라미터를 받는데 사용

필수여부 True여서 쓰이면 반드시 파라미터를 받아야함

필수인 파라미터가 아니라면 required False, defaultValue 로 기본값 지정 가능

**@RequestBody**

Json 형태의 http body를 java 객체로 변환

**@ModelAttribute**

form 형태의 http body와 요청 파라미터를 생성자나 Setter로 바인딩

매핑시키는 파라미터 타입, 객체 타입을 맞춰야함

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

자동 생성된 설명

Main/java/package경로에 service/**Service.java** 작성

DTO를 통해 받은 데이터를 이용해 비즈니스 로직을 처리

Service는 DAO를 @Autowired

@Service annotation 사용

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

자동 생성된 설명

Main/java.package경로에 dao/**Dao.java** 작성

DAO(Repository)는 Data Access Object로 DB의 data에 접근해 Service와 실제 DB를 연결

- DB에서 data를 꺼내오거나 넣는 역할

DB에 접근하기위한 로직과 비즈니스 로직을 분리하기 위해 사용

sql문은 resources/mybatis/mapper/BoardMapper.xml에 작성

@Repository annotation 사용

Bean에 자동등록 되어있는 SqlSessionTemplate를 @Autowired

**SqlSessionTemplate?**

SqlSession을 구현하고 코드에서 SqlSession을 대체

SqlSession이 스프링 트랜잭션에 사용될 수 있도록 보장하고,

필요한 시점에 세션을 닫고, 커밋하거나 롤백하는 것을 포함한 세션의 생명주기를 관리

스크린샷, 소프트웨어, 멀티미디어 소프트웨어, 텍스트이(가) 표시된 사진

자동 생성된 설명

Main/java/package경로에 dao/**BoardVO** 작성

VO (DTO)는 Value Object, Data Transfer Object로 DB의 데이터를 객체형태로 Transfer하기 위해 사용됨

VO (DTO)는 로직을 갖고 있지 않는 순수한 데이터 객체이며 getter, setter만을 갖음

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

자동 생성된 설명

application.properties 또는 application.yml에 설정한 mapper location에 **Mapper.xml** 작성

실제 DB를 제어하기 위한 sql문을 작성

태그들에 있는 id에 Dao에서 호출할 이름 설정

Dao에서 namespace.id로 호출

\*Mapper에서는 sql문의 마지막에 ;를 쓰지 않아도 동작

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

자동 생성된 설명

src/main/java 안에 /webapp/WEB-INF/view/ 경로를 만들고 /view/ 경로에 **jsp**파일을 작성

controller에서 return 할 boards, board, addBoard, editBoard .jsp작성

**jsp?**

JavaServer Pages로 HTML에 java 코드를 넣어 동적 웹페이지를 생성함

Spring에서 사용하기 위해 org.apache.tomcat.embed:tomcat-embed-jasper dependency 필요

**jstl?**

JavaServer Pages Standard Tag Library

java EE 기반 웹어플리케이션 개발 플랫폼을 위한 컴포넌트 모음

xml데이터 처리와 조건문, 반복문, 국제/지역화 같은 일을 처리하기위한 JSP태그 라이브러리를 추가함

jsp페이지 내에서 java코드를 바로 사용하지않고 태그 형태로 로직을 내장

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

**DB Table**

꼭 필요한 컬럼들은 Not Null

pk인 게시글 id는 자동 생성

Client(Request)

->Controller

->Service

->Dao(@Repository)

-> Mapper.xml(SQL)

->DB

**<구동 화면>**

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

**Spring MVC 구조**

**Model**

어플리케이션의 데이터이며 모든 데이터 정보를 가공하여 갖고 있는 컴포넌트

View 또는 Controller에 대한 어떤 정보도 알 수 없어야 함

변경이 일어나면 처리 방법을 구현해야 함

**View**

시각적인 UI 요소를 지칭하는 용어

Model이 갖고 있는 데이터를 저장하면 안됨

Model이나 Controller에 대한 정보를 알면 안됨, 단순히 표시만 해주는 역할

변경이 일어나면 처리 방법을 구현해야 함

**Controller**

Model과 View를 연결해주는 역할

Model 또는 View에 대한 정보를 알아야 함

Model 또는 View의 변경을 인지하여 대처해야 함

텍스트, 도표, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

스크린샷, 텍스트, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

**Dispatcher Servlet**

사용자의 모든 요청을 받아 처리함

프론트 컨트롤러에 해당하는 역할을 수행하며 Request를 각 Controller에게 위임

**Handler Mapping**

요청을 직접 처리할 컨트롤러를 탐색함

구체적인 매핑은 xml파일이나 java config 관련 어노테이션 등을 통해 처리

**HandlerAdapter**

매핑된 컨트롤러의 실행을 요청함

**Controller**

직접 요청을 처리하며 처리 결과를 반환

결과가 반환되면 HandlerAdapter가 ModelAndView 객체로 변환되며

여기에는 View Name과 같이 응답을 통해 보여줄 View에 대한 정보와 관련된 데이터가 포함

**View Resolver**

View Name을 확인한 후 컨트롤러로부터 받은 처리 결과를 반영할 View 파일(jsp) 탐색

**View**

처리 결과를 반영한 최종 화면 생성

**동작 순서**

1. 클라이언트가 서버에 요청을 하면 DispatcherServlet이 요청을 받음

2. DispatcherServlet은 프로젝트 파일 내의 servlet-context.xml의 @Controller 인자를 통해 등록한 요청 위임 Controller를 찾아 매핑된 Controller가 존재하면 @RequestMapping을 통해 요청을 처리할 메소드로 이동

3. Controller는 해당 요청을 처리한 Service를 받아 비즈니스 로직을 Service에게 위임

4. Service는 요청에 필요한 작업을 수행하고 요청에 대해 DB에 접근해야한다면 DAO에 요청하여 처리를 위임

5. DAO는 DB정보를 VO(DTO)를 통해 받아 Service에 전달

6. Service는 전달받은 데이터를 Controller에 전달

7. Controller는 Model 객체에게 요청에 맞는 View 정보를 담아 DispatcherServlet에게 전송

8. DispatcherServlet은 전달받은 View 정보를 ViewResolver에게 전달

9. ViewResolver는 응답할 View에 대한 JSP를 찾아 DispatcherServlet에게 전달

10. DispatcherServlet은 응답할 View의 Render를 지시하고 View는 로직을 처리

11. DispatcherServlet은 클라이언트에게 Rendering된 뷰를 응답하여 요청을 마침

**Dao vs Repository**

기능적으로 Data Access Layer(DB 관련 정보를 처리하는)의 구현체인 것은 공통점

Dao는 SQL에 대한 매핑이기 때문에 매핑 레벨이 SQL수준

Repository는 자바 Entity 객체를 테이블에 매핑 할 수 있으므로 매핑 레벨이 객체 수준

Dao는 퍼시스턴스 레이어에 속하여 퍼시스턴스 레이어에 대한 추상화

Repository는 도메인 레이어 객체 컬렉션의 추상화

Repository에서는 Dao를 여러 개 선언해서 사용할 수 있음

**VO vs DTO**

DTO는 Data Transfer Object로 계층(Layer)간 데이터 교환을 위해 사용하는 객체

데이터 교환만을 위해 사용하므로 로직을 갖지 않고 getter, setter 메소드만 갖음

VO는 Value Object로 값 그 자체를 표현하는 객체

로직을 포함할 수 있으며 객체의 불변성을 보장함

서로 다른 이름을 갖는 VO 인스턴스라도 모든 속성 값이 같다면 두 인스턴스는 같은 객체

이를 위해 VO에는 equals()와 hashCode()를 오버라이딩 해야함

**Servlet**

WebProgramming에서 Client의 요청을 처리

그 결과를 다시 Client에게 전송하는 자바 프로그래밍 기술

Servlet 클래스의 구현 규칙을 따름

Javax.servlet package에 정의된 인터페이스

**서블릿의 라이프 사이클을 위한 3가지 필수적인 메소드**를 정의

**-init()**

서블릿 생명 주기 중 초기화 단계에 호출됨

Javax.servlet.ServletConfig 인터페이스를 구현하는 오브젝트가 전달됨

이를 통해 서블릿이 웹어플리케이션에서 초기화 매개변수에 접근할 수 있음

**-service()**

초기화 이후 각 요청들이 들어오면 호출됨

각 요청들은 별도로 나눠진 스레드에서 처리됨

웹 컨테이너는 모든 요청에 대해 서블릿의 service() 메소드를 요청

service() 메소드는 요청의 종류를 판별하고 요청을 처리할 적절한 메소드로 전달

**-destroy()**

서블릿 객체가 파괴되어야 할 때 호출됨

해당 서블릿이 가지고 있던 자원을 release 해줌

**Servlet이 해야하는 일**

http request를 받아서 처리

request에 대한 response

**Servlet의 특징**

Client의 요청에 동적으로 작동

Java Thread를 이용해 동작

HTML 변경시 재컴파일 필요

Java 코드에 HTML이 들어가있음

HTML을 사용해서 요청에 응답