기본적인 웹 게시물 관리

스프링 MVC와 Mybatis를 이용한

CRUD(등록, 수정, 삭제, 조회)와

페이징 처리

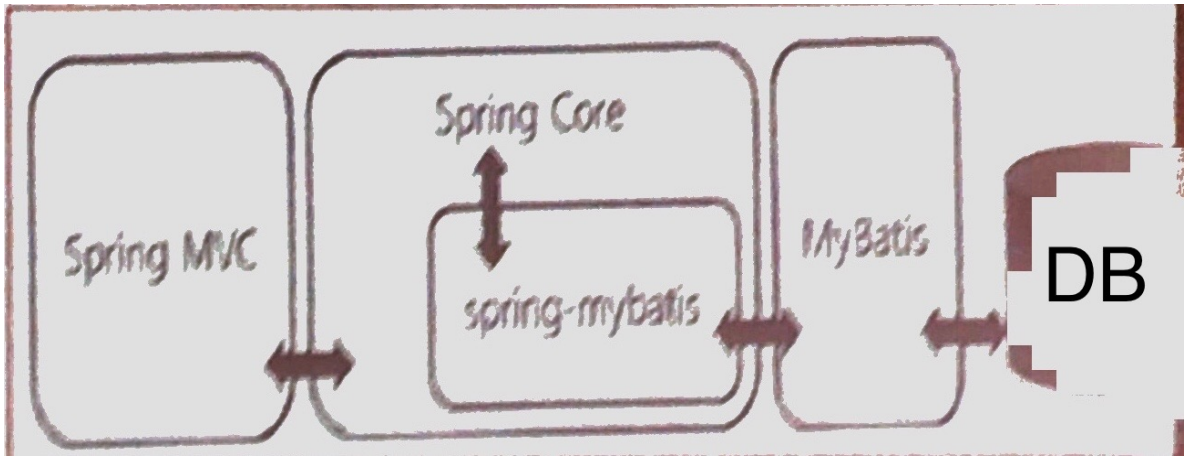
검색기능 의 게시물 관리를 제작.

중요하게 고려해야 할 부분

스프링 MVC를 이용하는 웹 프로젝트 전체 구조에 대한 이해

개발의 각 단계

- Presentation(화면 계층) : 화면에 보여주는 기술을 사용하는 영역
  Servlet/JSP나 스프링 MVC가 담당하는 영역.
  프로젝트의 성격에 맞추어 앱으로 제작하거나,
  CS(Client - Server)로 구성되는 경우도 있다.
  스프링 MVC와 JSP를 이용한 화면 구성이 이에 속한다.
- Business(비즈니스 계층) : 순수한 비느니스 로직을 담고 있는 영역
  이 영역이 중요한 이유
  고객기 원하는 요구사항을 반영하는 계층.
  이 영역의 설계는 고객의 요구 사항과 정확히 일치해야 한ㄷ.
  이 영역은 주로 xxxService와 같은 이름으로 구성하고,
  메서드 이름 역시 고객들이 사용하는 용어를 그대로 사용하는 것이 일반적.
- Persistence(영속 또는 데이터 계층) : 데이터를 어떤 방식으로 보관하고, 사용하는가에 대한 설계가 들어가는 계층.
  일반적인 경우, 데이터베이스를 많이 이용하지만,
  경우에 따라서 네트워크 호출이나 원격 호출 등의 기술이 접목될 수도 있다.
  이 영역은 Mybatis와 mybatis-spring을 이용하여 구성.



- Spring MVC : Presentation Tier 를 구성

root-context.xml, servlet-context.xml 등의 설정 파일이 해당 영역의 설정을 담당.

- Spring Core : POJO(Plain-Old-Java-Object)의 영역.

  스프링의 의존성 주입을 이용해서 객체강의 연관구조를 완성하여 사용

- Mybatis : 현실적으로 mybatis-spring을 이용하여 구성하는 영역

  SQL에 대한 처리를 담당하는 구조

1. 각 영역의 계층

프로젝트를 3-tier 로 구성하는 이유는 유지보수에 대한 필요성 때문

각 영역은 독립적으로 설계되어 추후 특정 기술이 변하더라도 필요한 부분을 전자제품의 부품처럼 쉽게 교환할수 있게 하는 방식

각 영역은 설계당시부터 영역을 구분하고, 해당 연결부위는 인터페이스를 이용하여 설계하는 것이 일반적.

1. 네이밍 규칙

- xxxController :

  스프링 MVC에서 동작하는 Controller 클래스를 설계할때 사용.

- xxxService, xxxServiceImpl :

  비즈니스 영역을 담당하는 인터페이스는 xxxService 방식.

  인터페이스를 구현한 클래스는 xxxServiceImpl이름을 사용

- xxxDAO, xxxRepository :

  DAO(Data - Access - Object) 나 Repository(저장소) 이름으로

  영역을 따로 구성하는 것이 보편적

  별도의 DAO를 구성하는 대신, Mybatis의 Mapper인터페이스를 활용.

- VO, DTO :

  VO나 DTO는 일반적으로 유사한 의미로 사용하는 용어.

  데이터를 담고있는 객체를 의미한다는 공통점이 있다.

  - VO : 주로 ReadOnly 목적이 강하고, 데이터 자체도 immutable(불변)하게 설계하는 것이 정석.
  - DTO : 주로 데이터 수집의 용도가 좀더 강하다.

    예) 웹 화면에서 로그인 하는 정보를 DTO로 처리하는 방식을 사용.

    테이블과 관련된 데이터는 VO라는 이름을 사용.

2. 패키지 구성

패키지의 구성은 프로젝트의 크기나 구성원들의 성향으로 결정.

- 규모가 적은 프로젝트:
  - Controller 영역을 별도의 패키지로 설계

    - Service 영역등을 하나의 패키지로 설계
- 규모가 큰 프로젝트 : 많인 Service 클래스와 Controller 들이 혼재할 경우
    - 비즈니스를 단위 별로 구분하고(즉, 비즈니스 단위 별로 패키지를 작성)

      다시 내부에서 Controller 패키지, Service 패키지 등으로 다시 나누는 방식을 이용.
    - 담당자가 명확해지고, 독립적인 설정을 가지는 형태로 개발

3. 작업 패키지 구성

com.이니셜 : 메인 패키지

com.이니셜.config : 프로젝트와 관련된 설정 클래스들

com.이니셜.controller : 스프링 MVC 의 Controller

com.이니셜.service : 스프링의 Service 인터페이스와 구형 클래스들

com.이니셜.domain : VO, DTO 클래스들

com.이니셜.persistence : Mybatis Mapper 인터페이스

com.이니셜.exception : 웹 관련 예외 처리

com.이니셜.aop : 스프링의 AOP 관련

com.이니셜.security : 스프링의 Security 관련

com.이니셜.util : 각종 유틸리티 클래스 관련

4. 요구사항 분석 설계

고객이 원하는 내용이 무엇이고,

오느정도까지 구현할 것인가에 대한 프로젝트의 범위를 정하는 것이 목적.

5. 요구사항

실제로 방대해 질 수 있으므로 프로젝트에서는 단계를 정확히 구분.

경험이 많은 팀 구성 : 초기 버전에 상당히 많은 기능을 포함시켜서 개발

반대의 경우 : 최대한 단순하고 눈에 보이는 결과를 만들어 내는 형태로 진행

온전한 문장으로 정리.

주어는 "고객"이고, 목적어는 "대상(domain)"이 된다.

예) 게시판의 경우, 게시물이 대상이 된다.

고객은 새로운 게시물을 등록할 수 있어야 한다.

고객은 특정 게시물을 조회할 수 있어야 한다.

고객은 작성한 게시물을 삭제할 수 있어야 한다.

등등.

테이블 : tbl_board

VO클래스 : com.이니셜.domain.BoardVO

게시물과 관련된 로직 : com.이니셜.service.BoardService / com.이니셜.BoardController

6. 요구사항에 따른 화면 설계

예) 고객은 새로운 게시물을 등록할 수 있어야 한다.

세부적인 설계 : '어떤 내용들을 입력하게 될 것인가'

이를 기준으로 테이블이나 클래스의 멤버 변수(인스턴스 변수) 들을 설계.

이러한 화면을 설계할 때는 주로 Mock-up(목업)들을 이용하는 경우가 많다.

대표적인 Mock-up 툴 : PowerPoint 나 Balsamiq studio, Pencil Mockup 등.

각 화면을 설계하는 단계에서는 사용자가 입력해야 하는 값과 함께 전체 페이지의 흐름을 설게.

이 화면의 흐름을 URL로 구성하게 되는데, 이경우 GET/POST 방식에 대하여 언급해둔다.