기말 Term Project 보고서

프로젝트명: Spring Boot와 JPA를 활용한 개인 웹사이트 확장 개발

1. 개요

본 프로젝트는 1학년 때 제작했던 개인 홈페이지를 기반으로, Spring Boot 프레임워크와 JPA(Java Persistence API)를 적용하여 동적인 웹 애플리케이션으로 확장하는 것을 목표로 합니다.

기존 개인 홈페이지는 정적인 HTML과 간단한 자바스크립트만으로 구성되어 있어 방문자 정보 저장이나 게시글 관리와 같은 기능을 제공하지 못했습니다. 이에 이번 텀 프로젝트에서는 **로그인 기능**을 도입하고, **데이터베이스와 연동되는 게시판(또는 앨범) 기능**을 구현하여 실제 서비스에 가까운 구조로 개선하였습니다.

또한 프로젝트 전체를 GitHub 저장소(`testWeb`)로 관리함으로써 버전 관리를 연습하고, 실무에서 많이 사용하는 Spring Boot + JPA + MySQL 조합에 익숙해지는 것을 부가적인 목표로 했습니다.

2. 요구사항 분석

2.1 기능 요구사항

본 프로젝트에서 구현한 주요 기능은 다음과 같습니다.

1. **회원 관리 기능**
    a. 로그인/로그아웃: 가입된 회원 정보를 바탕으로 인증을 수행하고, 세션을 통해 로그인 상태를 유지합니다.
2. **게시판/앨범 기능**
    a. 게시글 목록 조회: 데이터베이스에 저장된 게시글을 리스트 형태로 출력합니다.

b. 게시글 등록: 로그인한 사용자가 새 글을 작성하여 DB에 저장합니다.

c. 게시글 상세보기: 목록에서 특정 게시글을 선택하면 상세 내용을 확인할 수 있습니다.

d. 게시글 수정 및 삭제: 작성자 본인이 자신의 게시글을 수정하거나 삭제할 수 있습니다.

3. **메인 페이지 및 내비게이션**

a. 상단 메뉴를 통해 메인 페이지, 회원가입, 로그인, 게시판 등으로 이동할 수 있도록 구성하였습니다.

b. 1학년 때 제작한 기존 개인 홈페이지의 디자인 요소 일부를 유지하여, 프로젝트 간의 연속성과 완성도를 높였습니다.

2.2 비기능 요구사항

- **개발 언어 및 프레임워크**: Java, Spring Boot
- **데이터베이스**: MySQL
- **ORM 기술**: Spring Data JPA를 사용하여 엔티티와 테이블을 매핑하고, CRUD 기능을 구현한다.
- **템플릿 엔진**: Thymeleaf(또는 사용 중인 뷰 템플릿)
- **개발 환경**: Windows 10, JDK 17, VS Code/IntelliJ, Gradle 또는 Maven
- **버전 관리**: Git 및 GitHub 저장소(`testWeb`) 사용
- **기타 요구사항**: 기본적인 예외 처리 및 오류 페이지(500 에러 등) 제공

3. 시스템 설계

3.1 전체 구조

시스템은 크게 **클라이언트(웹 브라우저)**, **Spring Boot 애플리케이션 서버**, **MySQL 데이터베이스**로 구성됩니다.

1. 사용자는 웹 브라우저를 통해 회원가입, 로그인, 게시글 작성 등의 요청을 보냅니다.
2. 요청은 Spring MVC의 컨트롤러에 도착하며, 컨트롤러는 서비스 계층에 비즈니스 로직 처리를 위임합니다.
3. 서비스 계층에서는 JPA Repository를 통해 데이터베이스와 통신하고, 처리 결과를 다시 컨트롤러로 반환합니다.
4. 컨트롤러는 반환받은 데이터를 Model에 담아 뷰 템플릿(HTML)에 전달하고, 최종적으로 브라우저에 결과를 출력합니다.

이러한 구조를 통해 클라이언트와 서버, 데이터베이스 간의 역할을 명확히 분리하고, 유지보수성을 높였습니다.

3.2 데이터베이스 설계 및 ER 다이어그램

프로젝트에서 사용한 핵심 테이블(엔티티)은 `Member`와 `Post` 두 가지입니다.

1. **Member 테이블**
    a. 주요 컬럼:
        i. `id` (PK, bigint, auto increment)
        ii. `username` (사용자 아이디, unique)
        iii. `password` (비밀번호)
        iv. `name` (이름)
        v. `email` (이메일)
        vi. `reg_date` (가입일)
    b. 역할: 웹사이트에 가입한 회원의 기본 정보를 저장합니다.
2. **Post 테이블**
    a. 주요 컬럼:
        i. `id` (PK, bigint, auto increment)

ii. `title` (제목)

iii. `content` (내용)

iv. `created_date` (작성일)

v. `member_id` (작성자 외래 키, Member.id 참조)

b. 역할: 게시판 또는 앨범의 게시글 정보를 저장합니다.

ER 다이어그램 상에서 `Member`와 `Post`는 **1:N 관계**를 가진다. 즉, 한 명의 회원이 여러 개의 게시글을 작성할 수 있으며, 각 게시글은 정확히 한 명의 회원에 의해 작성된다. `Post` 테이블에는 `member_id` 외래 키를 두어 `Member` 테이블과 연결하였습니다.

3.3 시스템 흐름도

대표적인 시나리오인 로그인 흐름은 다음과 같습니다.

1. **로그인 흐름**

   a. 관리자가 로그인 페이지에서 아이디와 비밀번호를 입력한다.

   b. 컨트롤러는 MemberRepository를 통해 해당 관리자의 로그인 내용을 조회한다.

   c. 비밀번호가 일치하면 세션에 관리자 정보를 저장하고, 메인 페이지 또는 마이페이지로 리다이렉트한다.

   d. 정보가 일치하지 않을 경우, 로그인 페이지로 돌아가 오류 메시지를 출력한다.

이 흐름을 기준으로 순서도(Flow Chart)를 작성하여 보고서에 포함하였다.

4. 구현 내용

4.1 JPA 기반 엔티티 및 Repository 설계

Member 클래스는 @Entity 어노테이션을 통해 JPA 엔티티로 지정하였습니다.
@Id와 @GeneratedValue(strategy = GenerationType.IDENTITY)를 사용하여 기본 키를 자동 증가 값으로 설정하고, username 필드에는 @Column(nullable = false, unique = true)를 지정하여 필수 값이면서 중복될 수 없도록 하였다.

Post 엔티티에서는 @ManyToOne(fetch = FetchType.LAZY)와 @JoinColumn(name = "member_id")를 사용하여 게시글과 작성자 간의 다대일 관계를 매핑하였다. 이를 통해 특정 게시글에서 작성자 정보를 쉽게 조회하거나, 회원이 작성한 게시글 목록을 조회할 수 있습니다.

Repository 계층에서는 JpaRepository<Member, Long>과 JpaRepository<Post, Long>을 상속받는 인터페이스를 정의하였습다.
JPA가 제공하는 기본 CRUD 메서드(save, findById, findAll, deleteById 등)를 활용하였고, 로그인 처리를 위해 findByUsername(String username)과 같은 커스텀 메서드를 추가하여 사용하였습니다.

4.2 회원가입 및 로그인 기능 구현

로그인 기능은 /login 요청을 통해 처리된다. 사용자가 입력한 아이디로 회원 정보를 조회하고, 비밀번호가 일치할 경우 세션에 로그인 정보를 저장하여 이후 요청에서도 로그인 상태를 유지한다. 비밀번호가 일치하지 않으면 로그인 페이지로 되돌아가 "아이디 또는 비밀번호가 올바르지 않습니다"와 같은 안내 메시지를 출력하도록 구현하였습니다.

4.3 게시판 기능 구현

게시판 기능은 /post/list, /post/write, /post/view/{id} 등의 URL 패턴으로 구성하였다.

- **목록 페이지**에서는 PostRepository.findAll()을 통해 조회한 게시글 리스트를 테이블 형태로 출력합니다. 각 행에는 게시글 번호, 제목, 작성자, 작성일을 표시합니다.
- **글 작성 페이지**에서는 제목과 내용 입력 폼을 제공하며, 글 작성 시 현재 로그인한 사용자의 정보와 함께 DB에 저장되도록 하였습니다.

- **상세보기 페이지**에서는 특정 게시글을 ID로 조회하여 제목, 내용, 작성자, 작성일을 보여준다. 작성자 본인으로 로그인된 경우에만 수정/삭제 버튼이 보이도록 템플릿을 구성하였습니다.
- **수정 및 삭제 기능**은 HTTP 메서드(POST 또는 PUT, DELETE)와 매핑되는 컨트롤러 메서드를 통해 구현하였습니다.

5. 화면 구성 및 실행 결과

5.1 메인 페이지

메인 페이지는 기존 1학년 개인 홈페이지의 레이아웃을 유지하면서 상단에 **Home, 회원가입, 로그인, 게시판** 메뉴를 배치하였다.
사용자가 어떤 기능을 이용할 수 있는지 직관적으로 알 수 있도록 간단한 안내 문구를 추가하였습니다.

5.2 로그인 관련 화면

로그인 화면에서는 아이디와 비밀번호를 입력받고, 로그인에 실패하면 적절한 오류 메시지를 보여준다. 로그인에 성공하면 상단 메뉴에 로그인한 관리자의 내용을 출력하여 현재 인증된 상태임을 알 수 있게 하였습니다.

5.3 게시판 화면

게시판 목록 화면에서는 DB에서 조회한 게시글 목록을 표 형식으로 보여준다.
제목을 클릭하면 상세보기 페이지로 이동하며, 상세보기 화면에서는 제목, 내용, 작성자, 작성 시간을 확인할 수 있다. 작성자 본인으로 로그인된 경우, 수정 및 삭제 버튼이 함께 표시되도록 조건문을 사용하여 템플릿을 구성하였습니다.

각 화면은 너무 화려하지 않지만, 기본적인 가독성과 사용성을 고려하여 배치하였습니다.

6. 테스트 및 검증

프로젝트의 주요 기능에 대해 간단한 수동 테스트를 수행하였다. 주요 테스트 항목은 다

음과 같습니다.

테스트 항목	기대 결과	실제 결과
필수 항목 미입력 후 회원가입 시도	오류 메시지 출력, 회원가입 실패	기대 결과와 일치
중복 아이디로 회원가입 시도	"아이디가 이미 존재합니다" 메시지 출력	기대 결과와 일치
올바른 정보로 로그인 시도	메인 페이지 이동, 상단에 사용자 이름 표시	기대 결과와 일치
잘못된 비밀번호로 로그인 시도	로그인 실패 메시지 출력	기대 결과와 일치
로그인 후 게시글 작성	게시글 목록에 새 글이 추가되고 상세 보기 가능	기대 결과와 일치
게시글 작성자와 다른 계정으로 접속	해당 글에 수정/삭제 버튼 미표시	기대 결과와 일치

테스트 결과, 기본적인 회원 관리 및 게시판 기능은 정상적으로 동작함을 확인하였습니다. 다만 예외 상황(데이터베이스 장애, 네트워크 오류 등)에 대한 처리는 충분히 구현하지 못하였으며, 추후 보완이 필요합니다.

7. 결론 및 향후 개선 방향

본 프로젝트를 통해 1학년 때 제작한 정적인 개인 홈페이지를 Spring Boot 기반의 동적인 웹 애플리케이션으로 확장하는 경험을 할 수 있었습니다. 특히 JPA를 활용하여 엔티티를 설계하고, Repository를 통해 데이터베이스와 상호작용하는 과정에서 객체 지향적인 방식으로 데이터를 다루는 방법을 배울 수 있었습니다. 또한 GitHub를 이용해 프로젝트를 관리하면서 버전 관리의 중요성과 협업 도구의 사용법도 익힐 수 있었습니다.

그러나 아직 개선해야 할 점도 많다. 현재 로그인 기능은 기본적인 세션 방식으로만 구현되어 있어, 보안 측면에서 아쉬움이 있습니다.

추후에는 로그인 기능을 넣어 사용자의 로그인 내용을 넣어 사용자가 좋아요를 하는 부분까지 설정 하고 싶습니다. 사용자 DB는 넣었지만 로그인 하는부분을 구현하지 못하였습니다. 로그인 하는 내용을 구현화 하고 , 사용자가 웹에서 할 수 있는 기능을 추가적으

로 만들고 싶습니다.

향후에는 **Spring Security**를 도입하여 인증과 인가를 보다 체계적으로 구성하고, 비밀번호를 암호화하여 저장하도록 개선할 계획입니다. 또한 게시판에 **파일 업로드, 댓글, 페이징 처리, 검색 기능** 등을 추가하여 실제 서비스에 더 가까운 기능을 제공하고자 합니다. 마지막으로, 프론트엔드 디자인을 보완하여 모바일에서도 보기 좋은 반응형 UI로 발전시키는 것이 목표입니다.