

스프링 프레임워크와 오라클DB를 이용한

# 중고차 판매 사이트

PrivacyPolicy

multicampus

소속명

지능형 웹서비스 풀스택 개발 B반

팀명

All바른 중고차

개발기간

2022.01.27. ~ 2022.02.11

팀장

김대용

개발자

김대용(풀스택) 김가령(풀스택)  
김다은(풀스택) 신민종(프론트)

# 목차

1. 서론	
1.1 시장분석 .....	2
1.2 기대효과 .....	2
1.3 주제선정 .....	2
2. 계획	
2.1 시나리오 및 문제 정의 .....	2
2.2 일정계획 .....	2
2.3 노력 추정 .....	3
2.4 조직 계획 .....	4
3. 요구 분석	
3.1 요구 사항 .....	4
3.2 정책 .....	4
3.3 구조적 분석 .....	5
4. 설계	
4.1 시스템 아키텍처 .....	7
4.2 화면설계 .....	7
4.3 다이어그램 .....	10
5. 구현	
5.1 화면 구현 .....	11
5.2 소스 코드 .....	18
6. 추후 개발 방향 및 개발계획 .....	34
7. 프로젝트 업로드 주소 .....	35

## 1. 서론

### 1.1 시장분석

- 구매자가 직접 돌아다니며 중고차를 구매하는 경우가 많다.
- 중고차 시세를 모르는 구매자는 허위매물, 사기의 위험성이 매우 높다.
- 시간과 거리의 제약으로 인해 구매자에게 적합한 차량을 찾기가 힘들다.

### 1.2 기대효과

- 중고차 정보, 유통 투명성을 높여 사용자로부터 신뢰성 증가
- 단순 판매뿐만 아니라 이벤트나 특가를 통한 경쟁력 향상 가능
- 직영점을 통한 차량 배송 서비스를 통한 사용자 편리성

### 1.3 주제선정

직접 돌아다니며 차량을 비교하지 않고, 다양한 정보제공을 통해 사용자에게 적합한 차량을 배송하는 중고차 거래사이트

## 2. 계획

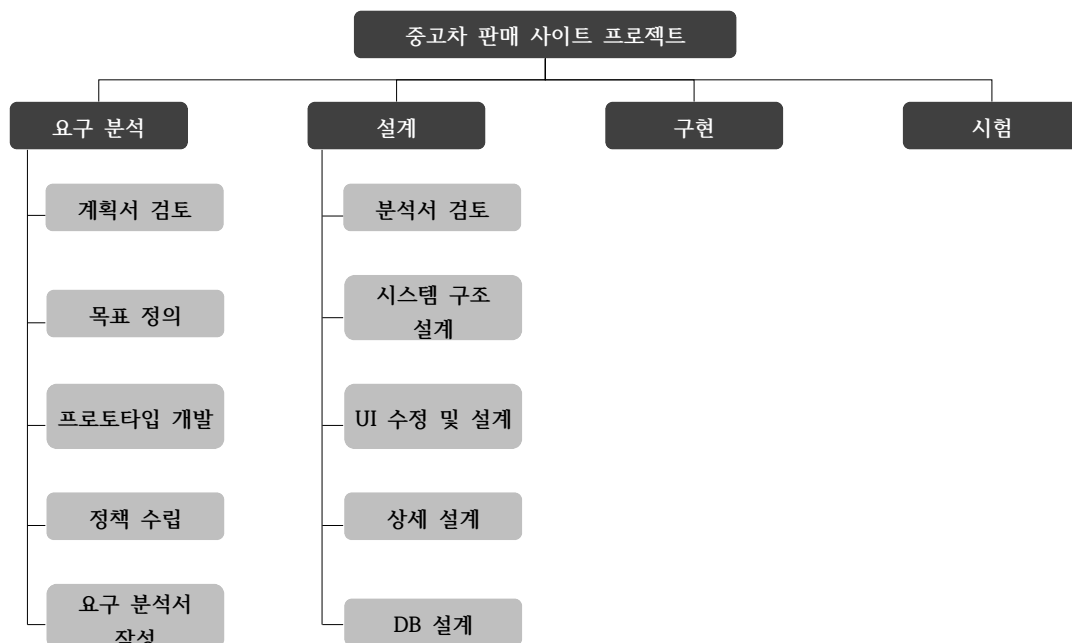
### 2.1 시나리오 및 문제 정의

사용자가 입력한 회원 정보를 저장해서 가입한 회원이 중고차 차량을 구매나 판매를 할 수 있는 중고차 거래사이트를 제작해 줄 것을 의뢰받았다.

### 2.2 일정계획

#### 2.2.1 작업 분해

전체 프로젝트의 일정을 예측하기 위하여 작업 분할 구조도(WBS)를 작성해서 큰 단위의 일을 상대적으로 관리하기 쉬운 작은 일로 나누었다.



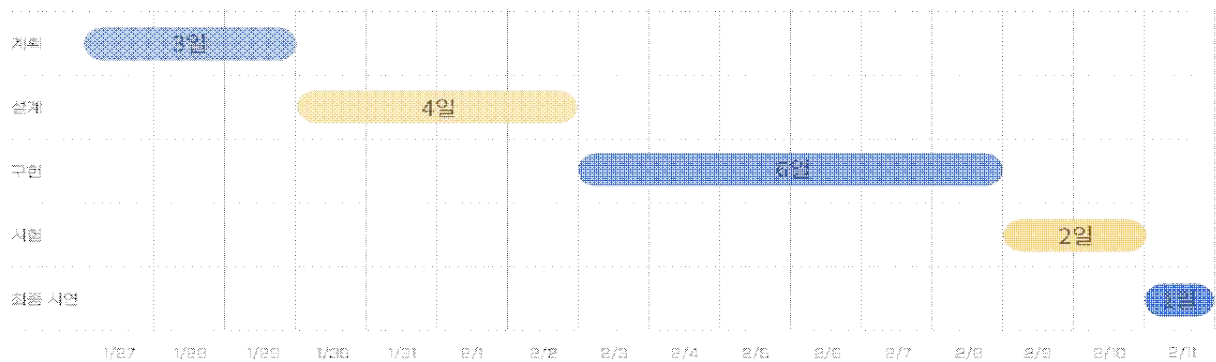
< 그림 1. 작업 분할 구조도(WBS) >

### 2.2.2 간트 차트

프로젝트 일정 관리를 위해 바 형태의 간트 차트를 작성하였다. 프로젝트의 주요 활동을 파악한 후, 각 활동의 일정을 시작하는 시점과 끝나는 시점을 연결할 막대 모양으로 만들어서 전체 일정을 한눈에 볼 수 있도록 하였다.

총개발 기한	2022.01.27. ~ 2022.02.11	
소 작업	중고차 거래사이트 웹 개발	
소 작업별 개발기간	소 작업	개발기간
소 작업별 개발기간	계획 / 요구분석	1월 27일 ~ 1월 29일
	설계	1월 30일 ~ 2월 2일
	구현	2월 3일 ~ 2월 8일
	시험	2월 9일 ~ 2월 10일
	최종 시연	2월 11일
개발 순서	계획 -> 요구분석 -> 설계 -> 구현 -> 시험 -> 최종 시연	
필요 자원	개발용 PC, 개발 공간	

< 표 1. 시스템 개발 일정계획 >



< 그림 2. 간트 차트 >

### 2.3 노력 추정

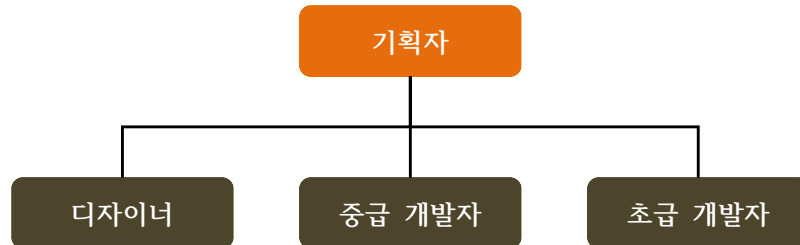
프로젝트를 진행, 완성하기 위하여 드는 비용과 기간을 추정할 필요가 있다. 개발 비용의 대부분은 소요되는 인력 자원이기 때문에 대부분의 비용 예측은 월-인원(Man-Month)으로 표현하는 노력 추정에 초점을 두었다.

구분	비용	기간	합계
기획자(1명)	350,000 원	3 일	1,050,000 원
디자인(1명)	200,000 원	4 일	800,000 원
중급개발자(1명)	300,000 원	5 일	1,500,000 원
초급개발자(1명)	150,000 원	5 일	750,000 원
총 비용			4,100,000 원

< 표2. 인건비 산정 내역 >

## 2.4 조직 계획

짧은 기간 동안 개발이 이루어지기 때문에 빠른 협의와 의사결정이 필요하다고 판단해서 1명의 기획자 아래 디자인 1명, 개발자 2명으로 이루어진 팀으로 구성하였다.



< 그림3. 조직의 구성 >

조직 단위	업무 기능
기획자	조직의 목표를 위한 전략 기획, 운영 정책, 프로세스, 아키텍처 등 분야별 전략 수립
디자이너	UI/UX 기획 및 아키텍처 구축, 프로토타입 검증, 설계 및 구현
중급 개발자	요구 분석 및 설계, 코드 작성, 테스트, 디버깅, 작업 및 지시
초급 개발자	업무 지원, 보조적 분석 및 설계, 구현 담당, 문서작업

< 표3. 조직의 업무 기능 >

## 3. 요구 분석

### 3.1 요구 사항

- 소비자에게 올바른 정보제공을 모토로 삼는 페이지 로고가 필요하다.
- 메인 페이지에 모든 기능을 간결히 배치해야 한다.
- 회원가입을 통해 로그인을 하여 개인정보를 저장할 수 있어야 한다.
- 차량 구매 탭을 통해 차량 정보를 검색하여 비교할 수 있어야 한다.
- 차량 선택 시 차량의 구체적인 정보를 표시해야 한다.
- 차량 구매 시 저장된 회원 정보를 가져와 주문할 수 있어야 한다.
- 마이페이지에서 고객 개인의 정보 및 서비스 사용내역을 확인할 수 있어야 한다.
- 게시판에는 공지사항, Q&A, 후기를 작성할 수 있어야 한다.

### 3.2 정책

NO	정책명	세부 항목	소개	정책 정의
1	회원가입	약관 동의	개인정보 수집 및 환불약관 동의 여부	* 필수 동의
2			마케팅 활용 및 수신 동의 여부	* 선택 동의 * 수신 방법 선택
3		필수정보 입력	아이디 생성 규칙	* 4~12자 영문, 숫자 조합 * 아이디 중복확인 진행 필수
4			비밀번호 생성 규칙	* 8~10자 영문 대/소문자, 숫자, 특수

				문자 조합 * 특수문자는 [~!@#\$\$%^*+=-~]만 허용 * 비밀번호 확인 텍스트란 필수 입력
5		부가 정보 입력	마케팅 활용을 위한 입력정보 수집	* 선택 입력
6	로그인	로그인 입력	로그인 규칙	* ID, PW 필수 입력 후 로그인
7		약관 동의	회원정보 사용을 위한 동의 여부	* 필수 동의
8	차량 구매	결제 방식	결제 방식 선택	* 현금, 할부, 카드 선택
9		구매 방식	구매방식 선택	* 리스로 구매시 상담 후 결제 * 리스가 아닐 시 즉시 결제
10	회원정보 수정	회원정보 수정	회원 정보 수정을 위한 방법	* 가입한 휴대폰번호 인증 후 회원 정보 수정
11	회원탈퇴	사이트 탈퇴	사이트 탈퇴 방법	* 가입한 휴대폰 번호 인증 후 마이페이지에서 회원 탈퇴

< 표 4. 정책(회원,구매) >

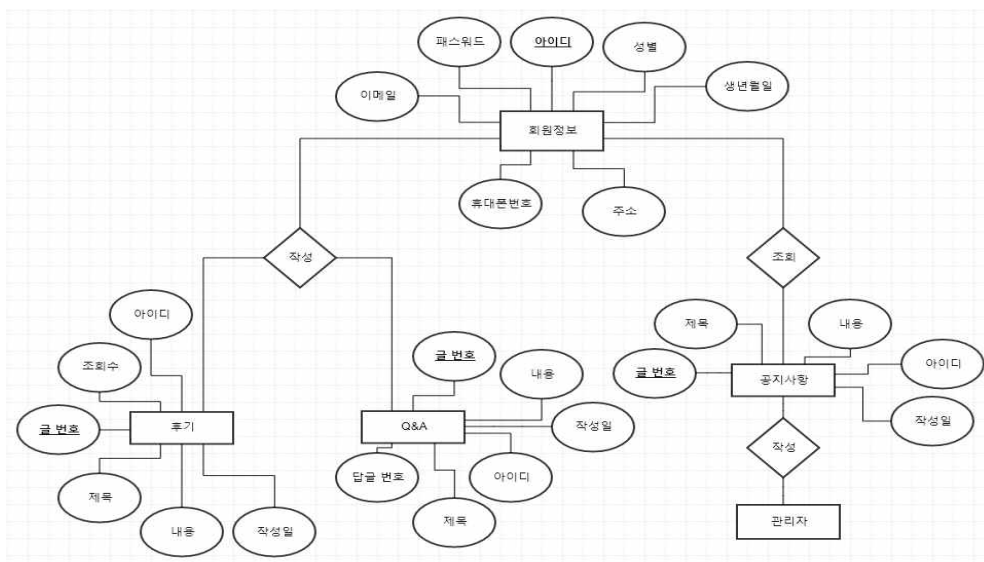
NO	정책명	세부 항목	사용자	보기	수정	삭제	쓰기	비고
12	게시판 정책	공지사항	운영자	O	O	O	O	
			로그인	O	X	X	X	
			비로그인	O	X	X	X	
13		Q&A	운영자	O	O	O	O	* 로그인한 사용자가 글을 작성하면 운영자만 해당 글에 대해 답변 가능 * 작성자 본인과 운영자만 수정,삭제 가능
			로그인	O	O	O	O	
			비로그인	O	X	X	X	
14		후기	운영자	O	O	O	O	* 차량을 구매한 사용자만 작성 가능 * 사진과 함께 업로드 가능 * 글 작성자와 운영자만 수정, 삭제 가능
			로그인	O	O	O	O	
			비로그인	O	X	X	X	

< 표 5. 정책(게시판) >

### 3.3 구조적 분석

#### 3.3.1 개념적설계

개념적설계 단계에서 ERD를 작성하였다. 회원정보, 후기, Q&A, 공지사항 테이블로 구성 되었다. 모든 테이블은 Primary Key를 가지고 있고 회원은 공지사항을 조회할 수 있고 관리자만 공지사항을 작성할 수 있다.



< 그림 4. ERD >

### 3.3.2 DB 설계

NO	컬럼 이름	속성	자료형	크기	NULL 여부 / 비고
1	id	아이디	VARCHAR2	20	PK
2	pwd	패스워드	VARCHAR2	30	NOT NULL
3	name	이름	VARCHAR2	50	NOT NULL
4	emailId	이메일아이디	VARCHAR2	20	NOT NULL
5	emailAddress	이메일주소	VARCHAR2	20	NOT NULL
6	phoneNum	휴대폰번호	VARCHAR2	20	NOT NULL
7	birth_y	생년	VARCHAR2	20	
8	birth_m	생월	VARCHAR2	20	
9	birth_d	생일	VARCHAR2	20	
10	gender	성별	VARCHAR2	10	
11	postcode	우편번호	VARCHAR2	10	
12	address	주소	VARCHAR2	50	
13	detailAddress	상세주소	VARCHAR2	50	
14	marketingOk	마케팅 동의 여부	VARCHAR2	20	
15	noticeKinds	마케팅 수신 여부	VARCHAR2	20	
16	regDate	가입일	DATE		DEFAULT SYSDATE

< 표 6. 회원정보 테이블(site\_member) >

NO	컬럼 이름	속성	자료형	크기	NULL 여부 / 비고
1	bno	글 번호	NUMBER		PK
2	title	제목	VARCHAR2	100	NOT NULL
3	content	내용	VARCHAR2	2000	NOT NULL
4	id	작성자 아이디	VARCHAR2	20	NOT NULL / FK
5	regdate	작성일	DATE		DEFAULT SYSDATE

< 표 7. 게시판 공지사항 테이블(Notice) >

NO	컬럼 이름	속성	자료형	크기	NULL 여부 / 비고
1	articleNO	글 번호	NUMBER		PK
2	parentNO	답글 번호	NUMBER		DEFAULT 0
3	title	제목	VARCHAR2	500	NOT NULL
4	content	내용	VARCHAR2	4000	
5	imageFileName	이미지 파일 이름	VARCHAR2	100	
6	writedate	작성일	DATE		DEFAULT SYSDATE
7	id	작성자 아이디	VARCHAR2	20	NOT NULL / FK

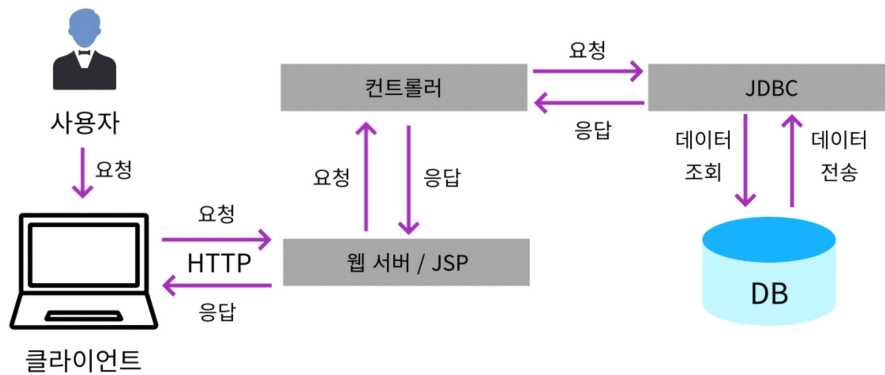
< 표 8. 게시판 Q&A 테이블(board\_QnA) >

NO	컬럼 이름	속성	자료형	크기	NULL 여부 / 비고
1	PSNo	글번호	NUMBER		PK
2	PSTitle	제목	VARCHAR2	100	NOT NULL
3	PSContent	내용	VARCHAR2	4000	NOT NULL
4	PSImageName	이미지 파일 이름	VARCHAR2	50	
5	writeDate	작성일	DATE		DEFAULT SYSDATE
6	hit	조회수	NUMBER		DEFAULT 0
7	ID	작성자 아이디	VARCHAR2	20	NOT NULL / FK

< 표 9. 게시판 구매후기 테이블(Postscripts) >

## 4. 설계

### 4.1 시스템 아키텍처



< 그림 5. 시스템 아키텍처 >

### 4.2 화면설계

#### 4.2.1 디자인 가이드

- 사용 폰트: 영어(Roboto), 한국어(Noto Sans Korean)
- 컬러: 붉은색 계열(#dc232d), 배경색(#cccccc), 버튼색(#5e5e5e)
- 사용 로고



- 페이지 상단: 사이트에서 제공하는 서비스로 갈수 있는 네이게이션 바 및 로고
- 페이지 하단: 사이트 정보 및 회사 정보 등 기타 약관에 관한 정보
- 사용 기술: HTML, CSS, 자바스크립트, 스프링 프레임워크, OracleDB 등

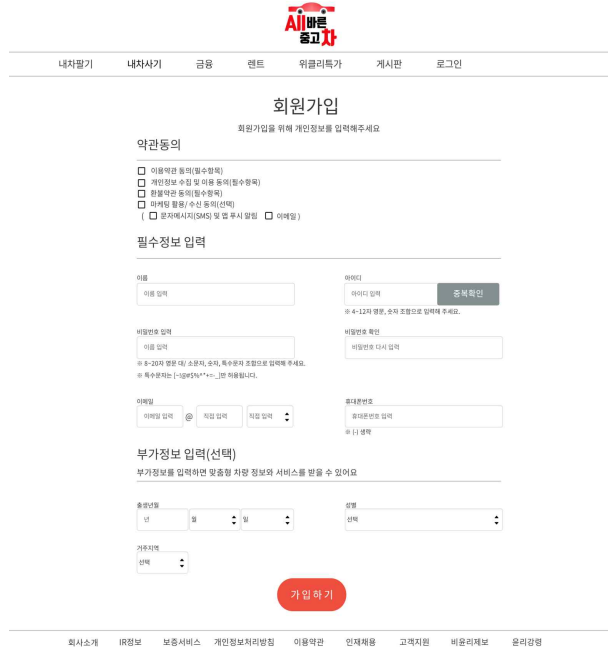


## 4.2.1 디자인 설계

요구사항과 디자인 가이드를 바탕으로 사용자가 이용하기 편하도록 화면을 구성하였다. 페이지는 로그인, 회원가입, 메인페이지, 차량 검색, 차량정보, 차량구매, 마이페이지, 게시판을 설계하였다.

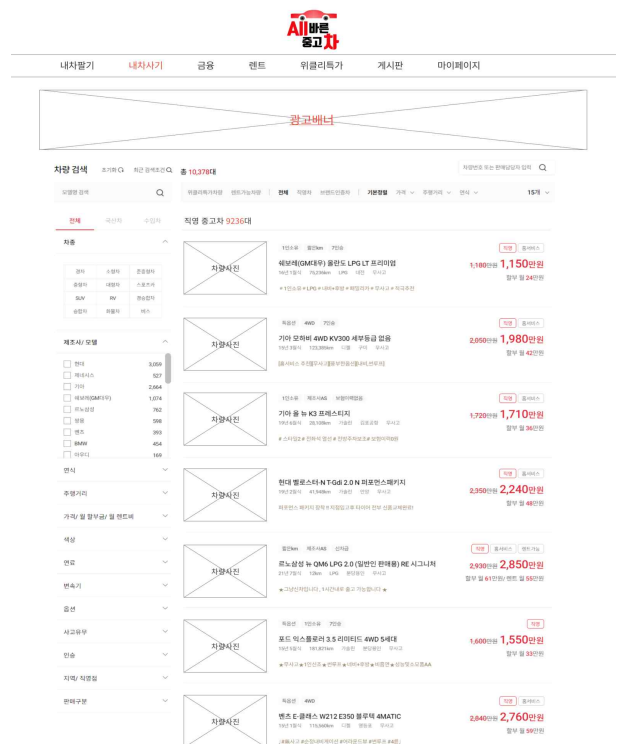
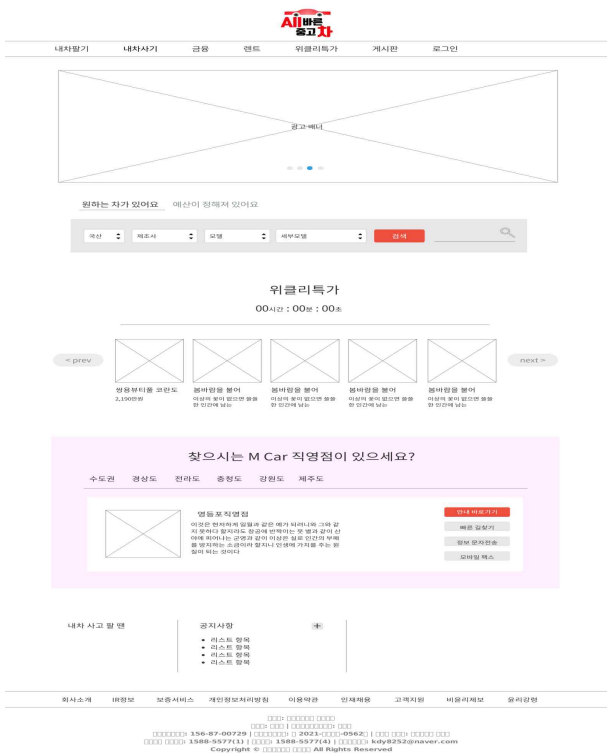


Copyright © 올버론중고차 주식회사 All Rights Reserved



회사소개 | R정보 | 보험서비스 | 개인정보처리방침 | 이용약관 | 인제채용 | 고객센터 | 바울리제보 | 윤리강령

156-87-00729 | 1588-5577(1) | 1588-5577(4) | kdy8252@naver.com  
Copyright © All Rights Reserved



내차팔기

내차사기

금융

렌트

유틸리티가

게시판

마이페이지

광고 배너

김 대용 고객님의

내차가 주유권리

내차가 신장관리

방문예약 신청내역

불량사상 접수내역

내차사가 주유권리

내차말가 신장관리

방문예약 신청내역

불량사상 접수내역

회원정보 수정

이름  
김지하

휴대폰번호  
010-1234-5678

본인인증

새 비밀번호 만들기

비밀번호 다시 입력

이메일  
kimj@naver.com

회원탈퇴

수신한 동적핵심 이벤트도, 할인 등 다양한 혜택 받을 수 있어요.

☐ 전체의 발송 / 수신 동의(선택)

(☐ 마케팅(SMS) 및 앱 푸시 알림    ☒ 마케팅)

취소

승정

회사소개

IR정보

보충서비스

개인정보처리방침

이용약관

인재채용

고객지원

비밀리포트

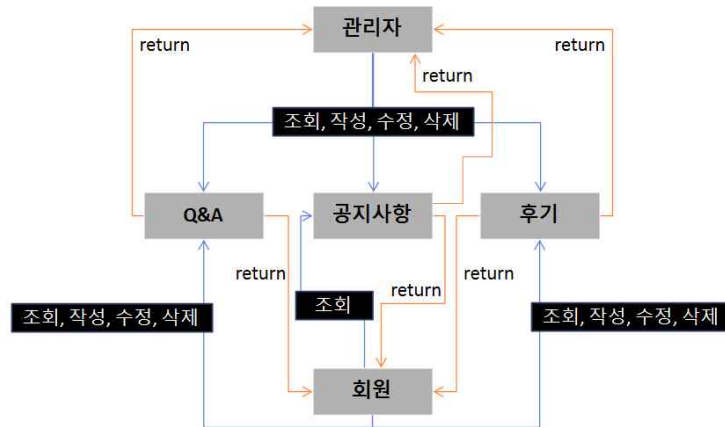
윤리강령

[illegible]

## 4.3 다이어그램

### 4.3.1 순서 다이어그램

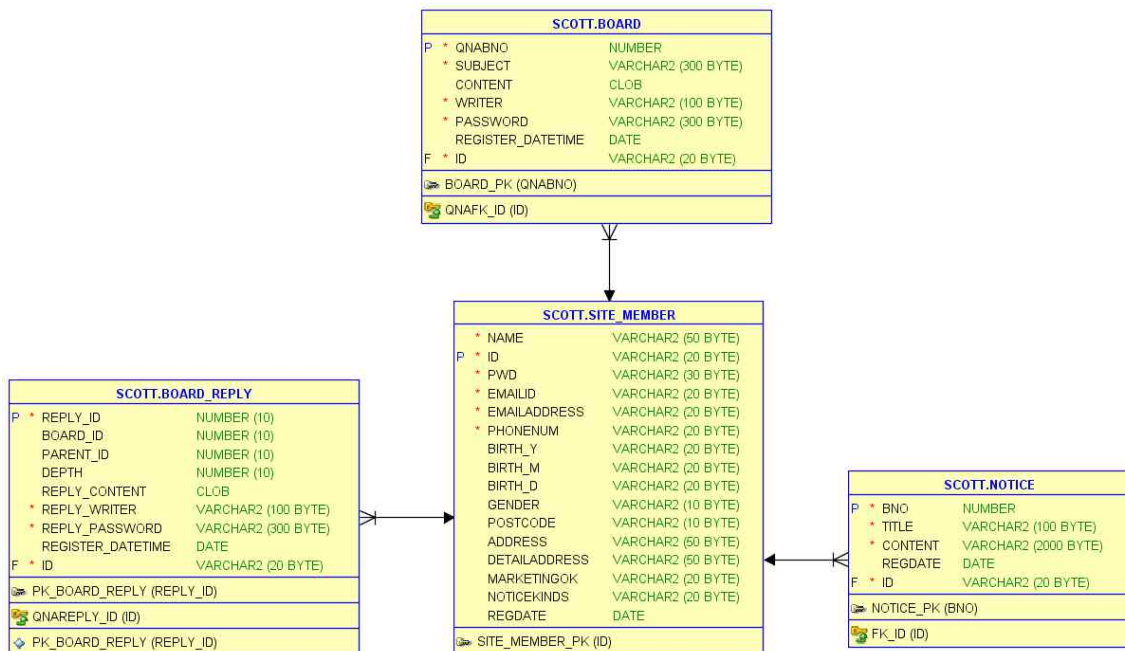
고객은 회원가입을 통해 회원정보를 저장하고 게시판의 Q&A, 후기에 글을 작성, 수정, 삭제, 조회를 할 수 있다. 공지사항은 관리자만 작성, 수정, 삭제가 가능하고 일반회원은 조회만 가능하다.



< 그림 6. 순서 다이어그램 >

### 4.3.2 클래스 다이어그램

아래와 같이 DB 테이블들의 관계를 파악하였다.



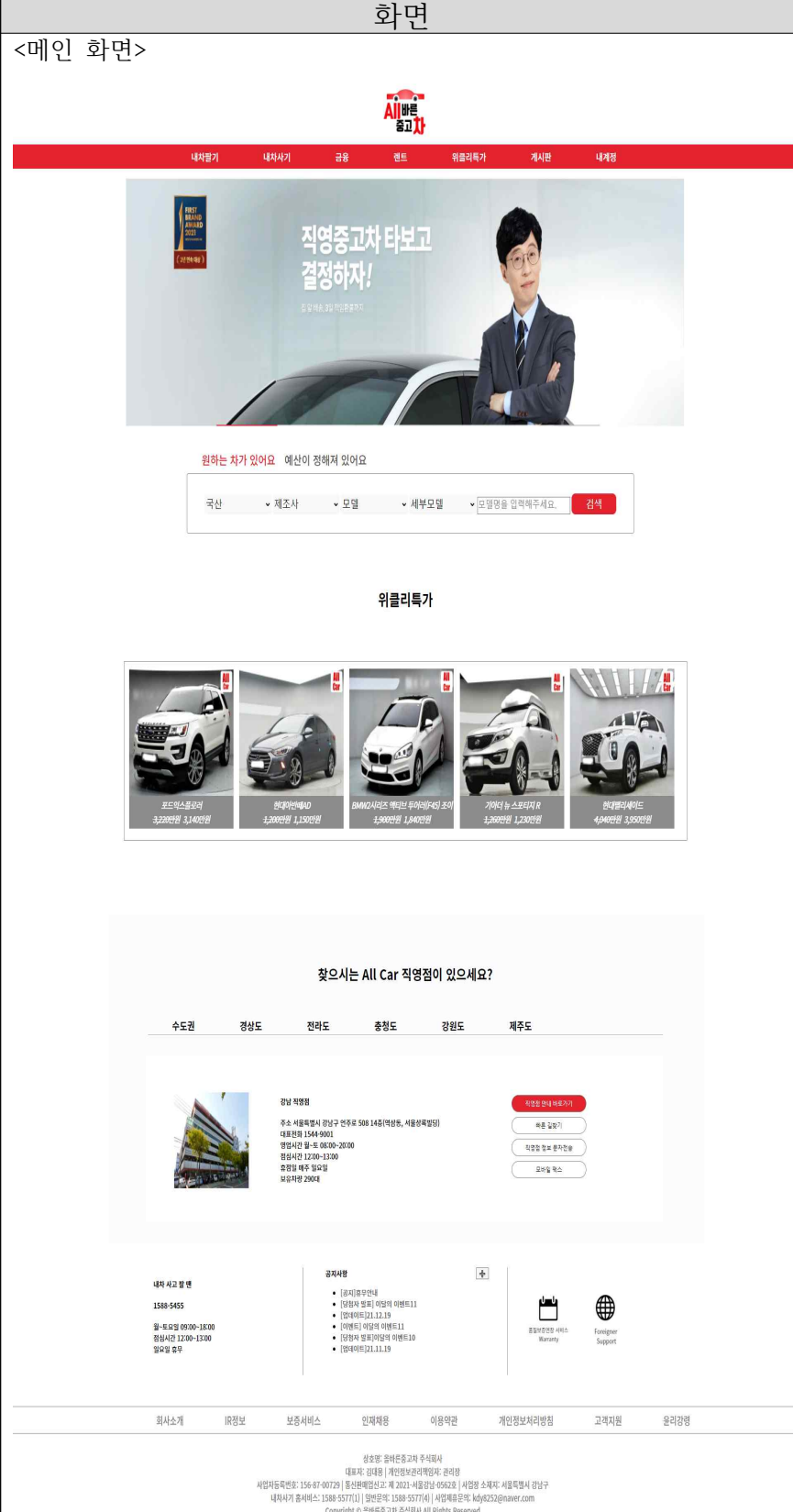
< 그림 7. 클래스 다이어그램 >

## 4.4 프로젝트 수행 도구



- 프론트엔드: HTML, CSS, Javascript
- 백엔드: Spring Framework 3.1.1, OracleDB, Java 1.8

## 5. 구현

### 5.1 화면 구현


화면	기능
<p>&lt;메인 화면&gt;</p> 	<ul style="list-style-type: none"> <li>- 다른 페이지로 넘어갈 수 있도록 헤더 메뉴 구성</li> <li>- 헤더 메뉴 아래 3초 간격으로 광고 배너 이미지 변경</li> <li>- 차량 또는 예산 조건에 따라 간단하게 차량 검색</li> </ul>

화면	기능
<p>&lt;회원가입&gt;</p>  <p>내차팔기 내차사기 금융 렌트 위험리특가 개시판 내결제</p> <p><b>회원가입</b></p> <p>회원가입을 위해 개인정보를 입력해주세요</p> <p><b>약관동의</b></p> <p> <input type="checkbox"/> 이용약관 동의(필수항목)  <input type="checkbox"/> 개인정보 수집 및 이용 동의(필수항목)  <input type="checkbox"/> 청약철회 동의(필수항목)  <input type="checkbox"/> 마케팅 활동 수신 동의(선택)  <input type="checkbox"/> ( <input type="checkbox"/> 문자메시지(SMS) 및 앱 푸시 알림 (1 이해함)         </p> <p><b>필수정보 입력</b></p> <p>           이름 <input type="text"/> 아이디 <input type="text"/> 중복확인 <input type="button" value="중복확인"/>  <small>※ 4~12자 영문, 숫자 조합으로 입력해 주세요.</small>            비밀번호 입력 <input type="password"/> 비밀번호 확인 <input type="password"/>  <small>※ 8~15자 영문 대 소문자, 숫자, 특수문자 조합으로 입력해 주세요.</small>  <small>※ 특수문자는 ~!@#\$%^&amp;*~를 허용합니다.</small>            이메일 <input type="text"/> 휴대전화번호 <input type="text"/>            이메일 입력 <input type="text"/> @ <input type="text"/> 입력         </p> <p><b>부가정보 입력(선택)</b></p> <p>부가정보를 입력하면 맞춤형 차량 정보와 서비스를 받을 수 있어요</p> <p>           출생년월일 <input type="text"/> 성별 <input type="text"/>            주소 <input type="text"/> 주소검색 <input type="button" value="주소검색"/>            주소 <input type="text"/>            상세주소 <input type="text"/> </p> <p><input type="button" value="가입하기"/></p> <p>회사소개   이용약관   개인정보   보증서비스   판매제품   위험약관   개인정보처리방침   고객센터   이용약관   문의</p> <p> <small>           상호명: 미래자동차 주식회사            대표자: 김대환   법인명: 미래자동차 주식회사   공시명: 미래자동차            사업자등록번호: 156-87-00729   통신판매업신고: 제 2021-서울강남-05452호   세무담당: 서울특별시 강남구            내차사기 홈페이지: 1588-5577(1)   고객센터: 1588-5577(4)   이메일: 1588-5577(5)   1588-5577(5)@mcar.com            Copyright © 미래자동차 주식회사 All Rights Reserved         </small> </p>	<ul style="list-style-type: none"> <li>- 필수입력 정보를 바탕으로 회원가입</li> <li>- 아이디 중복체크</li> <li>- 오픈 소스를 이용한 주소 검색 기능</li> <li>- 유효성 검사를 통해 정보 기입 오류 방지</li> </ul>


화면	기능
<p>&lt;로그인&gt;</p>  <p><b>로그인</b></p> <p> <input type="text"/> 아이디 입력  <input type="password"/> 비밀번호 입력  <input type="checkbox"/> 아이디 저장 <a href="#">비밀번호 찾기</a> <a href="#">아이디 찾기</a> </p> <p><input type="button" value="로그인"/></p> <p>           아직 M Car 회원이 아니세요? <a href="#">회원가입</a>            내차팔기 신청내역을 조회하실 수 있습니다 <a href="#">비회원 조회</a> </p>  <p> <b>직영중고차 홈서비스로 타보고 결정하자!</b>          집 앞 배송, 3일 책임완불까지       </p> <p>Copyright © 미래자동차 주식회사 All Rights Reserved</p>	<ul style="list-style-type: none"> <li>- 회원가입을 통해 저장된 정보를 바탕으로 로그인</li> <li>- 로그인 실패 시 알림</li> </ul>

## 화면

### <차량 검색>




[내차팔기](#) [내차사기](#) [금융](#) [렌트](#) [위클리특가](#) [게시판](#) [내계정](#)



검색

구분	<input type="checkbox"/> 국산	<input type="checkbox"/> 수입				
차종	<input type="checkbox"/> 경차	<input type="checkbox"/> 소형차	<input type="checkbox"/> 준중형차	<input type="checkbox"/> 중형차	<input type="checkbox"/> 대형차	더보기+
제조사/모델	<input type="checkbox"/> 현대	<input type="checkbox"/> 제네시스	<input type="checkbox"/> 기아	<input type="checkbox"/> 현대(기아차)	<input type="checkbox"/> 르노삼성	더보기+
연식	<input type="checkbox"/> 2022년	<input type="checkbox"/> 2021년	<input type="checkbox"/> 2020년	<input type="checkbox"/> 2019년	<input type="checkbox"/> 2018년	더보기+
주행거리	<input type="checkbox"/> 0~10000km	<input type="checkbox"/> 10000~20000km	<input type="checkbox"/> 20000~30000km	<input type="checkbox"/> 30000~40000km	<input type="checkbox"/> 40000~50000km	더보기+
연도	<input type="checkbox"/> 가솔린	<input type="checkbox"/> 디젤	<input type="checkbox"/> LPG	<input type="checkbox"/> 전기	<input type="checkbox"/> 가솔린+전기	더보기+
변속기	<input type="checkbox"/> 오토	<input type="checkbox"/> 수동	<input type="checkbox"/> 세미오토	<input type="checkbox"/> CVT		
옵션	<input type="checkbox"/> 내비게이션	<input type="checkbox"/> 선루프	<input type="checkbox"/> 가죽시트	<input type="checkbox"/> 항전시트/운전석	<input type="checkbox"/> 스테트키	더보기+
사고유무	<input type="checkbox"/> 무사고	<input type="checkbox"/> 단순교란	<input type="checkbox"/> 단순사고(합착)	<input type="checkbox"/> 사고		
연승	<input type="checkbox"/> 4인승	<input type="checkbox"/> 5인승	<input type="checkbox"/> 7인승	<input type="checkbox"/> 9인승	<input type="checkbox"/> 11인승	더보기+




**볼보 XC60 2세대 T6 인스크립션**

18년9월식 38,698km 가솔린 오토 무사고

○무사고/가솔린/보통세액K ★ 고급판/배진 / 가죽 시트/서풍

예약 중개서비스  
**5,940만원**  
 할부 월 126만원




**펠리셰이드 3.8 가솔린 AWD VIP**

20년11월식 18,853km 가솔린 오토 무사고

KED,프로그레스 #19인치휠 #내비게이션 #후방주차센서 #블루키로수

예약 중개서비스  
**4,530만원**  
 할부 월 94만원




**k3 1.6 디젤 프레스티지**

17년7월식 62,523km 디젤 오토 단순교란

서해 초년생의 적당한 정차

예약 중개서비스  
**1,495만원**  
 할부 월 31만원



**gv80 3.0 디젤 AWD**

21년2월식 8,976km 디젤 오토 무사고

#국보급 주행거리! #내비게이션/배진! #디젤 50/VI

예약 중개서비스  
**6,572만원**  
 할부 월 137만원

이전
1
2
3
4
5
6
7
8
9
10
다음

[회사소개](#)
[IR정보](#)
[보증서비스](#)
[인재채용](#)
[이용약관](#)
[개인정보처리방침](#)
[고객지원](#)
[윤리강령](#)

## 기능

- 주어진 조건 체크를 통해 차량 검색할 수 있는 페이지
- 차량 썸네일 클릭시 해당 차량 정보 페이지로 이동



화면

기능

<차량 구매>

내차팔기

내차사기

금융

렌트

위클리특가

게시판

내계정

명의자 정보를 입력해주세요

☐ 차량 계약 및 배송을 위해 회원정보를 사용하겠습니다.

명의자

휴대폰 번호

이름

· 숫자만 입력해주세요

본인인증

※ 다른 사람 명의로 계약을 원하시면 휴대폰인증 버튼을 눌러 명의자 변경 후 진행해 주세요

주민번호

주민번호 앞자리

주민번호 뒷자리

주소지(주민등록지)

우편번호

우편번호찾기

기본주소

상세주소를 입력하세요

※ 주소 정보가 잘못되었을 시 다시 입력하여 주세요

이메일

이메일주소

@ 도메인명

차량을 어디로 배송해 드릴까요?

차량 배송지 주소가 형식과 정보를 동일합니까?

동일합니다

다릅니다

받는 분

이름

휴대폰 번호

· 숫자만 입력해주세요

배송지 주소

우편번호

우편번호찾기

기본주소

상세주소를 입력하세요

※ 배송 거리에 따라 배송비가 산정되며, 정확한 배송비는 밑에서 확인 가능합니다.

최종 배송물

연도-용량

□

※ 최종 배송물을 정확히 없으면 탈주물 후에 배송됩니다.

고객님 어떻게 결제 하시겠어요?

결제방법을 선택하여 주세요(복수 선택 가능합니다)

현금

입금

카드

매각확실 차량이 있으신가요?

☐ 예

리스로 구매하시겠습니까?

☐ 예

※ 리스 선택 시에는 "보통 주 결제"만 가능합니다.

즉시 결제

상당 후 결제

회사소개

이용정보

보통서비스

안전제품

이용약관

개인정보처리방침

고객지원

알리상담

상호명: 알바몬코리아 주식회사  
대표자: 김대환 | 개인정보관리책임자: 권미영  
사업자등록번호: 156-87-00729 | 통신판매업신고: 제 2021-서울강남-0562호 | 사업자명: 서울특별시 강남구  
내사지점: 서울특별시 강남구 테헤란로 156-87-00729 | 고객센터: 1588-5577(11) | 이메일: info@albamon.com  
Copyright © 알바몬코리아 주식회사. All Rights Reserved

- 바로구매 버튼 클릭후 나타나는 페이지
- 구매자 정보를 입력한 후 차량 구매
- 즉시 결제와 상담 후 결제 버튼을 통해 구매 방법 분리

차량을 어디로 배송해 드릴까요?

차량 배송지 주소가 명세서 정보와 동일한가요?

동일합니다

다릅니다

받는 분

휴대폰 번호

배송지 주소

우편번호찾기

기본주소

상세주소를 입력하세요

※ 배송 거리에 따라 배송비가 산출되며, 정확한 배송비는 밑에서 확인 가능합니다.

최종 배송일

※ 최종 배송일을 정확히 알려면 일주일 후에 배송합니다.

고객님 어떻게 결제 하시겠어요?

결제방법을 선택하여 주세요(복수 선택 가능합니다)

현금

영부

카드

매각하실 차량이 있으신가요?

☐ 예

리스로 구매하시겠어요?

☐ 예

※ 리스 선택 시에는 "상당 후 결제"만 가능합니다.

즉시 결제

상당 후 결제

회사소개

IR정보

보증서비스

인재채용

이음역권

개인정보처리방침

고객지원

윤리강령

상용명: 엘베론중고차 주식회사  
 대표자: 김대웅 | 개인정보관리책임자: 권영랑  
 사업자등록번호: 156-87-00729 | 통산번호: 4592호 | 사업장 소재지: 서울특별시 강남구  
 내지서기 홈페이지: 1588-5577(1) | 영업문의: 1588-5577(6) | 사업제휴문의: kdy8252@naver.com  
 Copyright © 엘베론중고차 주식회사 All Rights Reserved



기능

- 회원정보 수정과 사이트 서비스 이용내역을 조회할 수 있는 마이페이지
- 회원탈퇴 기능

< 마이페이지 - 차량구매 탭 >

- 16 -

화면	기능
<p>&lt; 게시판 &gt;</p> 	<ul style="list-style-type: none"> <li>- 공지사항, Q&amp;A, 후기 글 작성, 조회, 수정, 삭제 가능</li> <li>- 공지사항은 관리자만 등록, 수정, 삭제 가능</li> </ul>

화면	기능
<p>&lt; 공지사항 작성글 보기&gt;</p> 	<ul style="list-style-type: none"> <li>- 공지사항 글 제목 클릭시 작성글 보기 페이지로 이동</li> <li>- 수정 버튼을 통해 수정 가능</li> <li>- 삭제 버튼을 통해 글 삭제 가능</li> </ul>

## 5.2 소스 코드

### 5.2.1 회원 정보 관리 코드

웹에서 요청한 로그인, 회원가입, 회원정보 수정 및 탈퇴를 오라클DB와 연동해서 로직을 수행한다. 추가적으로 메인, 로그인, 회원가입, 마이페이지 등의 페이지 요청에도 응답하는 로직을 수행한다.

#### < DB 쿼리문 수행 - member.xml >

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="mapper.member">
    <!-- 테이블 변수 -->
    <resultMap id="memResult" type="memberVO">
        <result property="name" column="name" />
        <result property="id" column="id" />
        <result property="pwd" column="pwd" />
        <result property="emailId" column="emailId" />
        <result property="emailAddress" column="emailAddress" />
        <result property="phoneNum" column="phoneNum" />
        <result property="birth_y" column="birth_y" />
        <result property="birth_m" column="birth_m" />
    </resultMap>
</mapper>
```

```

<result property="birth_d" column="birth_d" />
<result property="gender" column="gender" />
<result property="postcode" column="postcode" />
<result property="address" column="address" />
<result property="detailAddress" column="detailAddress" />
<result property="marketingOk" column="marketingOk" />
<result property="noticeKinds" column="noticeKinds" />
<result property="regdate" column="regdate" />
</resultMap>

<!-- 회원정보 출력 -->
<select id="selectAllMemberList" resultMap="memResult">
<![CDATA[
    select * from site_member order by regdate desc
]]>
</select>

<!-- 회원가입 정보 insert문 -->
<insert id="insertMember" parameterType="memberVO">
<![CDATA[
    insert into site_member(name, id, pwd, emailId, emailAddress, phoneNum,
    birth_y, birth_m, birth_d, gender, postcode, address, detailAddress, marketingOk, noticeKinds)
    values(#{name}, #{id}, #{pwd}, #{emailId}, #{emailAddress}, #{phoneNum},
    #{birth_y}, #{birth_m}, #{birth_d}, #{gender}, #{postcode}, #{address}, #{detailAddress}, #{marketingOk},
    #{noticeKinds})
]]>
</insert>

<!-- 회원정보 수정 update문 -->
<update id="updateMember" parameterType="memberVO">
<![CDATA[
    update site_member
    set pwd=#{pwd}, emailId=#{emailId}, emailAddress=#{emailAddress},
    phoneNum=#{phoneNum}
    where
    id=#{id}
]]>
</update>

<!-- 회원탈퇴 delete문 -->
<delete id="deleteMember" parameterType="String">
<![CDATA[
    delete from site_member
    where
    id=#{id}
]]>
</delete>

<!-- 로그인 정보 보내기 -->
<select id="loginById" resultType="memberVO" parameterType="memberVO">
<![CDATA[
    select * from site_member
    where id=#{id} and pwd=#{pwd}
]]>
</select>

<!-- 아이디 중복체크 쿼리문, 아이디 중복=true, 아이디 사용 가능=false -->
<select id="checkID" parameterType="String" resultType="String">
    select decode(count(*),1, 'true', 0, 'false') from site_member where id = #{id}
</select>
</mapper>

```

#### < 회원 정보 getter/setter - MemberVO.java >

```

package com.myspring.allCar.member.vo;

import java.sql.Date;

import org.springframework.stereotype.Component;

@Component("memberVO")
public class MemberVO {
    private String name;
    private String id;

```

```

private String pwd;
private String emailId;
private String emailAddress;
private String phoneNum;
private String birth_y;
private String birth_m;
private String birth_d;
private String gender;
private String postcode;
private String address;
private String detailAddress;
private String marketingOk;
private String noticeKinds;
private Date regdate;

public MemberVO() {
}

public MemberVO(String name, String id, String pwd, String emailId, String emailAddress,
String phoneNum,
String birth_y, String birth_m, String birth_d, String gender, String
postcode, String address,
String detailAddress, String marketingOk, String noticeKinds) {
    super();
    this.name = name;
    this.id = id;
    this.pwd = pwd;
    this.emailId = emailId;
    this.emailAddress = emailAddress;
    this.phoneNum = phoneNum;
    this.birth_y = birth_y;
    this.birth_m = birth_m;
    this.birth_d = birth_d;
    this.gender = gender;
    this.postcode = postcode;
    this.address = address;
    this.detailAddress = detailAddress;
    this.marketingOk = marketingOk;
    this.noticeKinds = noticeKinds;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getId() {
    return id;
}

public void setId(String id) {
    this.id = id;
}

public String getPwd() {
    return pwd;
}

public void setPwd(String pwd) {
    this.pwd = pwd;
}

public String getEmailId() {
    return emailId;
}

public void setEmailId(String emailId) {
    this.emailId = emailId;
}

public String getEmailAddress() {
    return emailAddress;
}

```

```

    }

    public void setEmailAddress(String emailAddress) {
        this.emailAddress = emailAddress;
    }

    public String getPhoneNum() {
        return phoneNum;
    }

    public void setPhoneNum(String phoneNum) {
        this.phoneNum = phoneNum;
    }

    public String getBirth_y() {
        return birth_y;
    }

    public void setBirth_y(String birth_y) {
        this.birth_y = birth_y;
    }

    public String getBirth_m() {
        return birth_m;
    }

    public void setBirth_m(String birth_m) {
        this.birth_m = birth_m;
    }

    public String getBirth_d() {
        return birth_d;
    }

    public void setBirth_d(String birth_d) {
        this.birth_d = birth_d;
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public String getPostcode() {
        return postcode;
    }

    public void setPostcode(String postcode) {
        this.postcode = postcode;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getDetailAddress() {
        return detailAddress;
    }

    public void setDetailAddress(String detailAddress) {
        this.detailAddress = detailAddress;
    }

    public String getMarketingOk() {
        return marketingOk;
    }

    public void setMarketingOk(String marketingOk) {
        this.marketingOk = marketingOk;
    }

```

```

    }

    public String getNoticeKinds() {
        return noticeKinds;
    }

    public void setNoticeKinds(String noticeKinds) {
        this.noticeKinds = noticeKinds;
    }

    public Date getRegdate() {
        return regdate;
    }

    public void setRegdate(Date regdate) {
        this.regdate = regdate;
    }
}

```

< member.xml을 이용해 DB I/O 처리 코드 - MemberDAOImpl.java >

```

package com.myspring.allCar.member.dao;

import java.util.List;

import org.apache.ibatis.session.SqlSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.dao.DataAccessException;
import org.springframework.stereotype.Repository;

import com.myspring.allCar.member.vo.MemberVO;

@Repository("memberDAO")
public class MemberDAOImpl implements MemberDAO {
    @Autowired
    private SqlSession sqlSession;

    // member.xml의 쿼리문을 이용해 db에 저장된 회원정보 불러오기
    @Override
    public List selectAllMemberList() throws DataAccessException {
        List<MemberVO> membersList = null;
        membersList = sqlSession.selectList("mapper.member.selectAllMemberList");
        return membersList;
    }

    // 회원정보 insert
    @Override
    public int insertMember(MemberVO memberVO) throws DataAccessException {
        int result = sqlSession.insert("mapper.member.insertMember", memberVO);
        return result;
    }

    // 회원탈퇴 delete
    @Override
    public int deleteMember(MemberVO memberVO) throws DataAccessException {
        int result = sqlSession.delete("mapper.member.deleteMember", memberVO);
        return result;
    }

    // 회원정보 수정
    @Override
    public int updateMember(MemberVO memberVO) throws DataAccessException {
        int result = sqlSession.update("mapper.member.updateMember", memberVO);
        return result;
    }

    // 로그인한 ID,PW로 회원정보 조회
    @Override
    public MemberVO loginById(MemberVO memberVO) throws DataAccessException {
        MemberVO vo = sqlSession.selectOne("mapper.member.loginById", memberVO);
        return vo;
    }
}

```

```

    }

    // 회원가입 품의 아이디 중복체크 "true" / "false" 문자열로 리턴
    @Override
    public String checkID(String id) throws DataAccessException {
        String result = sqlSession.selectOne("mapper.member.checkID", id);
        return result;
    }
}

```

#### < @Service를 이용하여 자바 내부 로직 처리 - MemberServiceImpl.java >

```

package com.myspring.allCar.member.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.dao.DataAccessException;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

import com.myspring.allCar.member.dao.MemberDAO;
import com.myspring.allCar.member.vo.MemberVO;

@Service("memberService")
@Transactional(propagation = Propagation.REQUIRED) //
public class MemberServiceImpl implements MemberService{
    @Autowired
    private MemberDAO memberDAO;

    @Override
    public List listMembers() throws DataAccessException {
        List membersList = null;
        membersList = memberDAO.selectAllMemberList();
        return membersList;
    }

    @Override
    public int addMember(MemberVO member) throws DataAccessException {
        return memberDAO.insertMember(member);
    }

    @Override
    public int deleteMember(MemberVO member) throws DataAccessException {
        return memberDAO.deleteMember(member);
    }

    @Override
    public int updateMember(MemberVO member) throws DataAccessException {
        return memberDAO.updateMember(member);
    }

    @Override
    public MemberVO login(MemberVO memberVO) throws Exception {
        return memberDAO.loginByld(memberVO);
    }

    @Override
    public String idcheck(String id) throws Exception {
        return memberDAO.checkID(id);
    }
}

```

#### < 웹에서의 요청과 응답 처리 - MemberControllerImpl.java >

```

package com.myspring.allCar.member.controller;

import java.io.PrintWriter;
import java.util.List;

```



```

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.myspring.allCar.member.service.MemberService;
import com.myspring.allCar.member.vo.MemberVO;

@Controller("memberController")
public class MemberControllerImpl implements MemberController {
    @Autowired
    private MemberService memberService;
    @Autowired
    private MemberVO memberVO;

    // 회원정보 리스트 요청, 응답
    @Override
    @RequestMapping(value="/member/listMembers.do", method = RequestMethod.GET)
    public ModelAndView listMembers(HttpServletRequest request, HttpServletResponse response)
throws Exception {
        String viewName = getViewName(request);
        List membersList = memberService.listMembers();
        ModelAndView mav = new ModelAndView(viewName);
        mav.addObject("membersList", membersList);
        return mav;
    }

    // 회원가입 수행
    @Override
    @RequestMapping(value = "/member/addMember.do", method = RequestMethod.POST)
    public ModelAndView addMember(@ModelAttribute("member") MemberVO member,
                                   HttpServletRequest request, HttpServletResponse response) throws Exception {
        request.setCharacterEncoding("utf-8");
        int result = 0;
        result = memberService.addMember(member);
        ModelAndView mav = new ModelAndView("redirect:/main.do");
        return mav;
    }

    // 로그인 성공, 실패 정보 세션에 저장
    @Override
    @RequestMapping(value = "/member/login.do", method = RequestMethod.POST)
    public ModelAndView login(@ModelAttribute("member") MemberVO member,
                              RedirectAttributes rAttr,
                              HttpServletRequest request, HttpServletResponse response)
throws Exception {
        ModelAndView mav = new ModelAndView();
        memberVO = memberService.login(member);
        if(memberVO != null) {
            HttpSession session = request.getSession();
            session.setAttribute("member", memberVO);
            session.setAttribute("isLogOn", true);
            mav.setViewName("redirect:/main.do");
        }else {
            response.setContentType("text/html; charset=UTF-8");
            PrintWriter out = response.getWriter();
            out.println("<script>"
                + "alert('입력한 정보가 올바르지 않습니다.');"
                + "location.href='/allCar/member/loginForm.do';"
                + "</script>");
            out.flush();
        }
        return mav;
    }
}

```

```

    }

    // 로그아웃 정보 세션에 저장
    @Override
    @RequestMapping(value = "/member/logout.do", method = RequestMethod.GET)
    public ModelAndView logout(HttpServletRequest request, HttpServletResponse response)
    throws Exception {
        HttpSession session = request.getSession();
        session.removeAttribute("member");
        session.removeAttribute("isLogOn");
        ModelAndView mav = new ModelAndView();
        mav.setViewName("redirect:/main.do");
        return mav;
    }

    // 회원탈퇴 후 로그아웃
    @Override
    @RequestMapping(value = "/member/deleteMember.do", method = RequestMethod.POST)
    public ModelAndView deleteMember(@ModelAttribute("member") MemberVO member,
        HttpServletRequest request, HttpServletResponse response) throws Exception
    {
        request.setCharacterEncoding("utf-8");
        HttpSession session = request.getSession();
        session.removeAttribute("member");
        session.removeAttribute("isLogOn");
        int result = memberService.deleteMember(member);
        ModelAndView mav = new ModelAndView();
        mav.setViewName("redirect:/main.do");
        return mav;
    }

    // 회원정보 수정 요청, 응답
    @Override
    @RequestMapping(value = "/member/updateMember.do", method = RequestMethod.POST)
    public ModelAndView updateMember(@ModelAttribute("member") MemberVO member,
        HttpServletRequest request, HttpServletResponse response) throws Exception
    {
        request.setCharacterEncoding("utf-8");
        int result = memberService.updateMember(member);
        ModelAndView mav = new ModelAndView("redirect:/main.do");
        return mav;
    }

    // 아이디 중복 체크
    @Override
    @RequestMapping(value="/member/idcheck.do", method = RequestMethod.POST)
    public ResponseEntity idcheck(@RequestParam("id") String id, HttpServletRequest request,
        HttpServletResponse response) throws Exception{
        ResponseEntity resEntity = null;
        String result = memberService.idcheck(id); // 아이디 중복 결과 'true' 또는 'false' 반환
        resEntity =new ResponseEntity(result, HttpStatus.OK);
        return resEntity;
    }

    // 메인 화면 view
    @Override
    @RequestMapping(value="/main.do", method = RequestMethod.GET)
    public ModelAndView main(HttpServletRequest request, HttpServletResponse response) throws
    Exception {
        String viewName = getViewName(request);
        ModelAndView mav = new ModelAndView(viewName);
        return mav;
    }

    // 마이페이지 화면 view
    @Override
    @RequestMapping(value="/member/mypage.do", method = RequestMethod.GET)
    public ModelAndView mypage(HttpServletRequest request, HttpServletResponse response)
    throws Exception {
        String viewName = getViewName(request);
        ModelAndView mav = new ModelAndView(viewName);
        return mav;
    }

    // *Form.do 요청에 응답
    @RequestMapping(value = "/member/*Form.do", method= RequestMethod.GET)

```

```

    public ModelAndView form(HttpServletRequest request, HttpServletResponse response) throws
Exception {
        String viewName = getViewName(request);
        ModelAndView mav = new ModelAndView(viewName);
        mav.setViewName(viewName);
        return mav;
    }

    // 요청 받은 주소 값 자르기
    private String getViewName(HttpServletRequest request) throws Exception {
        String contextPath = request.getContextPath();
        String uri = (String) request.getAttribute("javax.servlet.include.request_uri");
        if (uri == null || uri.trim().equals("")) {
            uri = request.getRequestURI();
        }

        int begin = 0;
        if (((contextPath == null) || "".equals(contextPath))) {
            begin = contextPath.length();
        }

        int end;
        if (uri.indexOf(";") != -1) {
            end = uri.indexOf(";");
        } else if (uri.indexOf("?") != -1) {
            end = uri.indexOf("?");
        } else {
            end = uri.length();
        }

        String viewName = uri.substring(begin, end);
        if (viewName.indexOf(".") != -1) {
            viewName = viewName.substring(0, viewName.lastIndexOf("."));
        }
        if (viewName.lastIndexOf("/") != -1) {
            viewName = viewName.substring(viewName.lastIndexOf("/" + 1),
viewName.length());
        }
        return viewName;
    }
}

```

### 5.2.1 공지사항 관리 코드

웹에서 요청한 공지사항 페이지 불러오기 글 작성, 수정, 삭제 및 검색을 오라클DB와 연동해서 로직을 수행한다.

#### < DB 쿼리문 수행 - noticeMapper.xml >

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="noticeMapper">

    <!-- 게시판 글 작성 -->
    <insert id="insert">
        INSERT INTO NOTICE(      BNO
                                , TITLE
                                , CONTENT
                                , ID )
        VALUES(      NOTICE_SEQ.NEXTVAL
                    , #{title}
                    , #{content}
                    , #{id} )

    </insert>

    <select id="listPage" resultType="com.myspring.allCar.board.notice.vo.NoticeVO">

```

```

parameterType="com.myspring.allCar.board.notice.vo.SearchCriteria">
    SELECT      BNO
                , TITLE
                , CONTENT
                , ID
                , REGDATE
    FROM (
        SELECT BNO
                , TITLE
                , CONTENT
                , ID
                , REGDATE
                , ROW_NUMBER() OVER(ORDER BY BNO DESC) AS RNUM
        FROM NOTICE
        WHERE 1=1
        <include refid="search"></include> )
    WHERE RNUM BETWEEN #{rowStart} AND #{rowEnd}
    ORDER BY BNO DESC
</select>

<select id="listCount" parameterType="com.myspring.allCar.board.notice.vo.SearchCriteria"
resultType="int">
    SELECT COUNT(BNO)
    FROM NOTICE
    WHERE 1=1
    <include refid="search"></include>
    AND BNO>0
</select>

<select id="read" parameterType="int"
resultType="com.myspring.allCar.board.notice.vo.NoticeVO">
    SELECT      BNO
                , TITLE
                ,CONTENT
                ,ID
                ,REGDATE
    FROM NOTICE
    WHERE BNO = #{bno}

</select>

<sql id="search">
    <if test="searchType != null">
        <if test="searchType == 't'.toString()">AND TITLE LIKE '%' || #{keyword} ||
    '%</if>
        <if test="searchType == 'c'.toString()">AND CONTENT LIKE '%' || #{keyword}
    || '%</if>
        <if test="searchType == 'w'.toString()">AND ID LIKE '%' || #{keyword} ||
    '%</if>
        <if test="searchType == 'tc'.toString()">AND (TITLE LIKE '%' || #{keyword} ||
    '%') or (CONTENT LIKE '%' || #{keyword} || '%')</if>
    </if>
</sql>

<update id="update" parameterType="com.myspring.allCar.board.notice.vo.NoticeVO">
    UPDATE NOTICE
    SET TITLE    = #{title},
        CONTENT  = #{content}
    WHERE BNO = #{bno}

</update>

<delete id="delete" parameterType = "int">
    DELETE
    FROM NOTICE
    WHERE BNO = #{bno}

</delete>
</mapper>

```

< 게시판 속성 getter/setter - NoticeVO.java >

```
package com.myspring.allCar.board.notice.vo;
```

```

import java.util.Date;

public class NoticeVO {

    private int bno;
    private String title;
    private String content;
    private String id;
    private Date regdate;

    public int getBno() {
        return bno;
    }
    public void setBno(int bno) {
        this.bno = bno;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }
    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public Date getRegdate() {
        return regdate;
    }
    public void setRegdate(Date regdate) {
        this.regdate = regdate;
    }
}

```

#### < 게시판 속성 getter/setter - Criteria.java >

```

package com.myspring.allCar.board.notice.vo;

public class Criteria {

    private int page;
    private int perPageNum;
    private int rowStart;
    private int rowEnd;

    public Criteria() {
        this.page = 1;
        this.perPageNum = 10;
    }

    public void setPage(int page) {
        if (page <= 0) {
            this.page = 1;
            return;
        }
        this.page = page;
    }

    public void setPerPageNum(int perPageNum) {
        if (perPageNum <= 0 || perPageNum > 100) {
            this.perPageNum = 10;
            return;
        }
        this.perPageNum = perPageNum;
    }
}

```

```

    public int getPage() {
        return page;
    }

    public int getPageStart() {
        return (this.page - 1) * perPageNum;
    }

    public int getPerPageNum() {
        return this.perPageNum;
    }

    public int getRowStart() {
        rowStart = ((page - 1) * perPageNum) + 1;
        return rowStart;
    }

    public int getRowEnd() {
        rowEnd = rowStart + perPageNum - 1;
        return rowEnd;
    }

    @Override
    public String toString() {
        return "Criteria [page=" + page + ", perPageNum=" + perPageNum + ", rowStart=" +
rowStart + ", rowEnd=" + rowEnd
        + "];"
    }
}

```

#### < 게시판 속성 getter/setter - PageMaker.java >

```

package com.myspring.allCar.board.notice.vo;

import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;

import org.springframework.web.util.UriComponents;
import org.springframework.web.util.UriComponentsBuilder;

public class PageMaker {

    private int totalCount;
    private int startPage;
    private int endPage;
    private boolean prev;
    private boolean next;
    private int displayPageNum = 10;
    private Criteria cri;

    public void setCri(Criteria cri) {
        this.cri = cri;
    }

    public void setTotalCount(int totalCount) {
        this.totalCount = totalCount;
        calcData();
    }

    public int getTotalCount() {
        return totalCount;
    }

    public int getStartPage() {
        return startPage;
    }

    public int getEndPage() {
        return endPage;
    }
}

```

```

    public boolean isPrev() {
        return prev;
    }

    public boolean isNext() {
        return next;
    }

    public int getDisplayPageNum() {
        return displayPageNum;
    }

    public Criteria getCri() {
        return cri;
    }

    private void calcData() {
        endPage = (int) (Math.ceil(cri.getPage() / (double)displayPageNum) * displayPageNum);
        startPage = (endPage - displayPageNum) + 1;

        int tempEndPage = (int) (Math.ceil(totalCount / (double)cri.getPerPageNum()));
        if (endPage > tempEndPage) {
            endPage = tempEndPage;
        }
        prev = startPage == 1 ? false : true;
        next = endPage * cri.getPerPageNum() >= totalCount ? false : true;
    }

    public String makeQuery(int page) {
        UriComponents uriComponents =
            UriComponentsBuilder.newInstance()
                                .queryParam("page", page)
                                .queryParam("perPageNum",
cri.getPerPageNum())
                                .build();

        return uriComponents.toUriString();
    }

    public String makeSearch(int page)
    {
        UriComponents uriComponents =
            UriComponentsBuilder.newInstance()
                                .queryParam("page", page)
                                .queryParam("perPageNum", cri.getPerPageNum())
                                .queryParam("searchType", ((SearchCriteria)cri).getSearchType())
                                .queryParam("keyword", encoding(((SearchCriteria)cri).getKeyword()))
                                .build();
        return uriComponents.toUriString();
    }

    private String encoding(String keyword) {
        if(keyword == null || keyword.trim().length() == 0) {
            return "";
        }

        try {
            return URLEncoder.encode(keyword, "UTF-8");
        } catch (UnsupportedEncodingException e) {
            return "";
        }
    }
}

```

#### < 게시판 속성 getter/setter - SearchCriteria.java >

```

package com.myspring.allCar.board.notice.vo;

public class SearchCriteria extends Criteria {

    private String searchType = "";
    private String keyword = "";
}

```

```

    public String getSearchType() {
        return searchType;
    }
    public void setSearchType(String searchType) {
        this.searchType = searchType;
    }
    public String getKeyword() {
        return keyword;
    }
    public void setKeyword(String keyword) {
        this.keyword = keyword;
    }
    @Override
    public String toString() {
        return "SearchCriteria [searchType=" + searchType + ", keyword=" + keyword + "]";
    }
}

```

< noticeMapper.xml을 이용해 DB I/O 처리 코드 - NoticeDAOImpl.java >

```

package com.myspring.allCar.board.notice.dao;

import java.util.List;

import javax.inject.Inject;

import org.apache.ibatis.session.SqlSession;
import org.springframework.stereotype.Repository;

import com.myspring.allCar.board.notice.vo.NoticeVO;
import com.myspring.allCar.board.notice.vo.SearchCriteria;

@Repository
public class NoticeDAOImpl implements NoticeDAO {

    @Inject
    private SqlSession sqlSession;

    //게시글 작성
    @Override
    public void write(NoticeVO noticeVO) throws Exception {
        sqlSession.insert("noticeMapper.insert", noticeVO);
    }

    //게시물 목록 조회
    @Override
    public List<NoticeVO> list(SearchCriteria scri) throws Exception {
        return sqlSession.selectList("noticeMapper.listPage", scri);
    }

    //게시물 총 갯수
    @Override
    public int listCount(SearchCriteria scri) throws Exception {
        return sqlSession.selectOne("noticeMapper.listCount", scri);
    }

    //게시물 조회
    @Override
    public NoticeVO read(int bno) throws Exception {
        return sqlSession.selectOne("noticeMapper.read", bno);
    }

    //게시물 수정
    @Override
    public void update(NoticeVO noticeVO) throws Exception {
        sqlSession.update("noticeMapper.update", noticeVO);
    }
}

```



```

    }

    //게시물 삭제
    @Override
    public void delete(int bno) throws Exception {

        sqlSession.delete("noticeMapper.delete", bno);

    }

}

```

### < @Service를 이용하여 자바 내부 로직 처리 - NoticeServiceImpl.java >

```

package com.myspring.allCar.board.notice.service;

import java.util.List;

import javax.inject.Inject;

import org.springframework.stereotype.Service;

import com.myspring.allCar.board.notice.dao.NoticeDAO;
import com.myspring.allCar.board.notice.vo.NoticeVO;
import com.myspring.allCar.board.notice.vo.SearchCriteria;

@Service
public class NoticeServiceImpl implements NoticeService {

    @Inject
    private NoticeDAO dao;

    //게시글 작성
    @Override
    public void write(NoticeVO noticeVO) throws Exception {

        dao.write(noticeVO);

    }

    //게시물 목록 조회
    @Override
    public List<NoticeVO> list(SearchCriteria scri) throws Exception {

        return dao.list(scri);

    }

    //게시물 총 갯수
    @Override
    public int listCount(SearchCriteria scri) throws Exception {

        return dao.listCount(scri);

    }

    //게시물 조회
    @Override
    public NoticeVO read(int bno) throws Exception {

        return dao.read(bno);

    }

    //게시물 수정
    @Override
    public void update(NoticeVO noticeVO) throws Exception {

        dao.update(noticeVO);

    }

    //게시물 삭제
    @Override
    public void delete(int bno) throws Exception {

        dao.delete(bno);

    }

}

```

```
}
```

#### < 웹에서의 요청과 응답 처리 - NoticeController.java >

```
package com.myspring.allCar.board.notice.controller;

import javax.inject.Inject;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.myspring.allCar.board.notice.service.NoticeService;
import com.myspring.allCar.board.notice.vo.NoticeVO;
import com.myspring.allCar.board.notice.vo.PageMaker;
import com.myspring.allCar.board.notice.vo.SearchCriteria;

@Controller
@RequestMapping("/notice/*")
public class NoticeController {

    private static final Logger logger = LoggerFactory.getLogger(NoticeController.class);

    @Inject
    NoticeService service;

    //게시판 글 작성 화면
    @RequestMapping(value = "/notice/writeView", method = RequestMethod.GET)
    public void writeView() throws Exception {

        logger.info("writeView");

    }

    //게시판 글 작성
    @RequestMapping(value = "/notice/write", method = RequestMethod.POST)
    public String write(NoticeVO noticeVO) throws Exception {

        logger.info("write");

        service.write(noticeVO);

        return "redirect:/notice/list";

    }

    //게시판 목록 조회
    @RequestMapping(value = "/notice/list", method = RequestMethod.GET)
    public String list(Model model, @ModelAttribute("scri") SearchCriteria scri) throws Exception {

        logger.info("list");

        model.addAttribute("list", service.list(scri));

        PageMaker pageMaker = new PageMaker();
        pageMaker.setCri(scri);
        pageMaker.setTotalCount(service.listCount(scri));

        model.addAttribute("pageMaker", pageMaker);

        return "notice/board";

    }

    //게시판 조회
    @RequestMapping(value = "/readView", method = RequestMethod.GET)
    public String read(NoticeVO noticeVO, @ModelAttribute("scri") SearchCriteria scri, Model
```

```

model) throws Exception {

    logger.info("read");

    model.addAttribute("read", service.read(noticeVO.getBno()));
    model.addAttribute("scri", scri);

    return "notice/readView";

}

// 게시판 수정뷰
@RequestMapping(value = "/updateView", method = RequestMethod.GET)
public String updateView(NoticeVO noticeVO, @ModelAttribute("scri") SearchCriteria scri,
Model model) throws Exception{
    logger.info("updateView");

    model.addAttribute("update", service.read(noticeVO.getBno()));
    model.addAttribute("scri", scri);

    return "notice/updateView";
}

//게시판 수정
@RequestMapping(value = "/update", method = RequestMethod.POST)
public String update(NoticeVO noticeVO, @ModelAttribute("scri") SearchCriteria scri,
RedirectAttributes rttr) throws Exception {

    logger.info("update");

    service.update(noticeVO);

    rttr.addAttribute("page", scri.getPage());
    rttr.addAttribute("perPageNum", scri.getPerPageNum());
    rttr.addAttribute("searchType", scri.getSearchType());
    rttr.addAttribute("keyword", scri.getKeyword());

    return "redirect:/notice/list";

}

//게시판 삭제
@RequestMapping(value = "/delete", method = RequestMethod.POST)
public String delete(NoticeVO noticeVO, @ModelAttribute("scri") SearchCriteria scri,
RedirectAttributes rttr) throws Exception {

    logger.info("delete");

    service.delete(noticeVO.getBno());

    rttr.addAttribute("page", scri.getPage());
    rttr.addAttribute("perPageNum", scri.getPerPageNum());
    rttr.addAttribute("searchType", scri.getSearchType());
    rttr.addAttribute("keyword", scri.getKeyword());

    return "redirect:/notice/list";

}

}

```

## 6. 추후 개발 방향 및 개발계획

- 로그인 페이지 아이디 저장 기능 추가
- 아이디, 비밀번호 찾기 기능 추가
- 휴대폰 본인인증 기능 추가
- 게시판 Q&A, 후기 글 작성 추가
- 공지사항 메인에 보여주는 기능 추가
- 향후 앱페이지로도 개발할 계획

## 7. 프로젝트 업로드 주소

### 7.1 Git 주소

김대용: [https://github.com/cheesecup/Semi\\_Project.git](https://github.com/cheesecup/Semi_Project.git)

김가령: [https://github.com/LMM07/all\\_the\\_car3.git](https://github.com/LMM07/all_the_car3.git)

김다은: <https://github.com/olssu/Back.git>

신민종: [https://github.com/MinJongS/All\\_my\\_Car.git](https://github.com/MinJongS/All_my_Car.git)

### 7.2 유튜브 주소

김대용: <https://www.youtube.com/watch?v=jVmJUmWQKLc>

김가령: [https://www.youtube.com/watch?v=Wu\\_r7GtEnxY](https://www.youtube.com/watch?v=Wu_r7GtEnxY)