

SPRING의 MVC2 패턴을 적용한 부동산 홈페이지 구현

최선종

— 목차

01. 프로젝트 개요

02. 시스템 구성

03. 기능 설명

1. 프로젝트 개요

○ 목 적 : MVC2 등을 적용하여 부동산 거래 웹사이트 생성
(회원가입, 부동산 매물 정보 추천 기능 구현)

○ 기 간 : 2021.03.09. ~ 04.15.(5주간)

○ 기대효과

▲ 프로그래밍 언어 숙달

▲ MVC2 구조 습득

▲ 정보 추천 알고리즘 방식 이해

○ 결과 : 부동산 거래 홈페이지 구축 완료

1. 프로젝트 개요

프로젝트 개발 기간 (WBS)

SPRING 의 MVC2 패턴을 적용한 부동산 홈페이지 구현

사전 준비

03.09~03.10

팀 구성 및 프로젝트
아이디어 공유

기획

03.15~03.16

주제 선정 및 구체화

요구 분석

03.17~03.18

요구사항 정의서 및
테이블 흐름도 작성

설계

03.19~03.22

각 페이지 초안 및 DB
테이블/변수명 선정

구현 시작

03.23~04.06

기본기능 중심 코드 구현
및 DB 연동

구현 완료

04.07~04.09

디자인 적용(부트스트랩,
CSS)
상세 주석 수정

통합 및 테스트

04.12~04.14

각 파트 통합 및 오류 개선

마무리

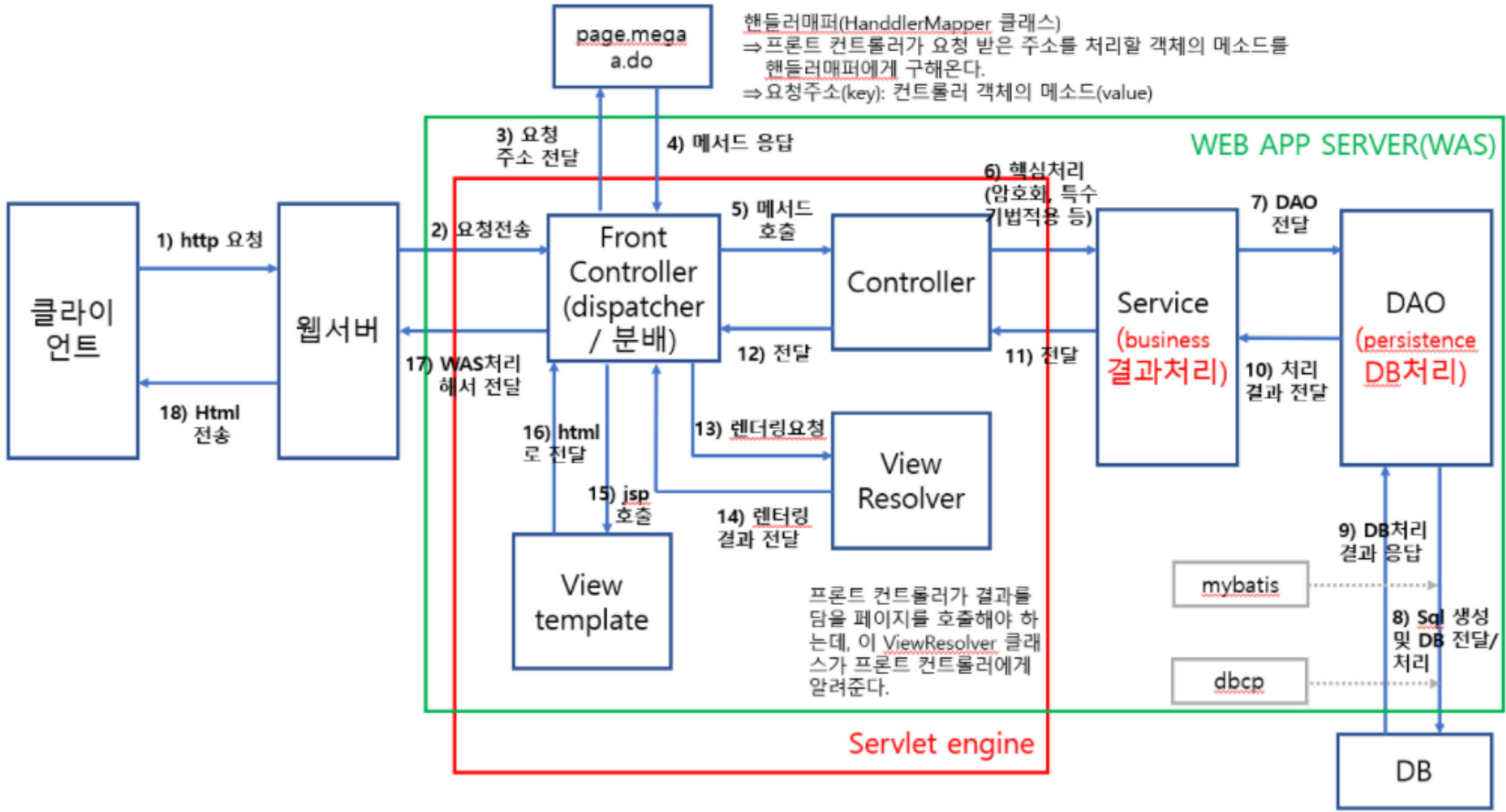
04.14~04.15

발표자료 작성 및 제출

2. 시스템 구성도 - 사용 프로그램



2. 시스템 구성도 - MVC2 흐름도



2. 시스템 구성도 - 요구사항 정의서 및 테이블 항목 구성

요구사항 정의서

프로젝트명	Spring을 활용한 신촌 지역 부동산 매물 추천 홈페이지 구축		
작성자	최선종	작성일	2021.03.16
버전	1.0.0	최종수정일	2021.03.16

구분	요구ID	내용	비고
User Requirement -Service	CSJ-001	사용자는 원하는 거래 유형이 포함된 회원가입 가능	사용자 회원가입
	CSJ-002	사용자는 가입한 ID, PW로 로그인 가능	사용자 회원가입
	CSJ-003	사용자의 개인정보 수정 가능	사용자 정보수정
	CSJ-004	로그인 후 메인페이지에서 사용자에게 적합한 물건을 추천	
Functional Requirement -Technology	CSJ-005	중복된 아이디 사용이 불가능 하도록 해야함	아이디 중복 체크
	CSJ-006	비밀번호는 대문자, 특수문자를 반드시 포함	비밀번호 체크
	CSJ-007	카카오 연동하여 카카오 아이디로 가입이 가능	카카오연동
	CSJ-008	기존 웹사이트와 연동하여 기존 이메일 주소로 회원가입이 가능	이메일연동
	CSJ-009	가입 유형(일반, 관리자)에 따라 회원가입 가능	유형별 회원가입
	CSJ-010	거주지역, 타입, 유형 등에 따라 사용자 선택정보에 적합한 물건 추천	물건추천
Non Functional Requirement -Protection &Change	CSJ-011	일반 회원은 관리자 페이지에 접근할 수 없어야 함	
	CSJ-012	회원가입시 관심 거래 유형을 최소 1개 이상 선택하여야 함	

Table_CSJ_01						
테이블ID		RUSER		작성자	최선종	
테이블설명		회원정보				
번호	칼럼ID	형식	NULL	KEY	내용	비고
1	RID	varchar2(20)	NN	PK	아이디	
2	RPW	varchar2(30)	NN		비밀번호	
3	RNAME	varchar2(10)	NN		이름	
4	RTEL	varchar2(15)	NN		전화번호	
5	REMAIL	varchar2(30)	NN		이메일	
6	RBIRTH	varchar2(10)	NN		생년월일	
7	RADDRESS	varchar2(50)	NN		주소	
8	RGENDER	varchar2(2)	NN		성별	
9	RTYPE	varchar2(5)	NN		거래타입	전세, 월세
10	RROOM	varchar2(5)	NN		거래형태	원룸, 투룸
11	RINTEREST	varchar2(5)	NN		기타조건	펍세권

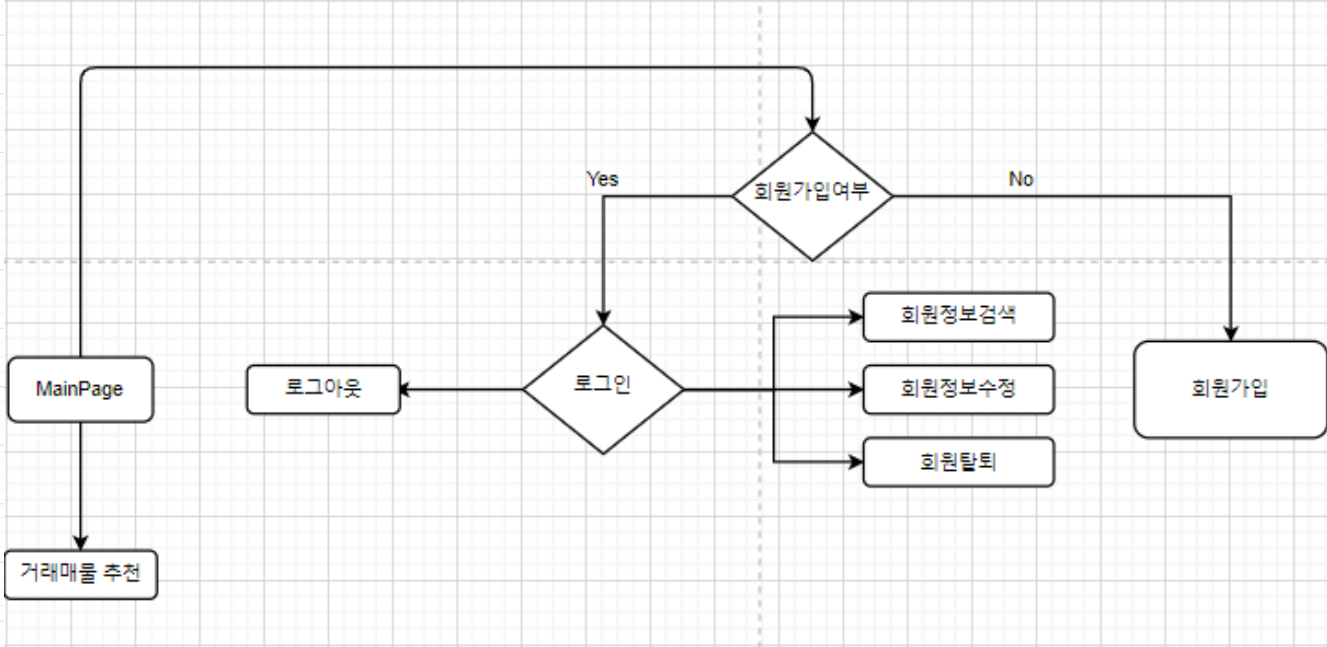
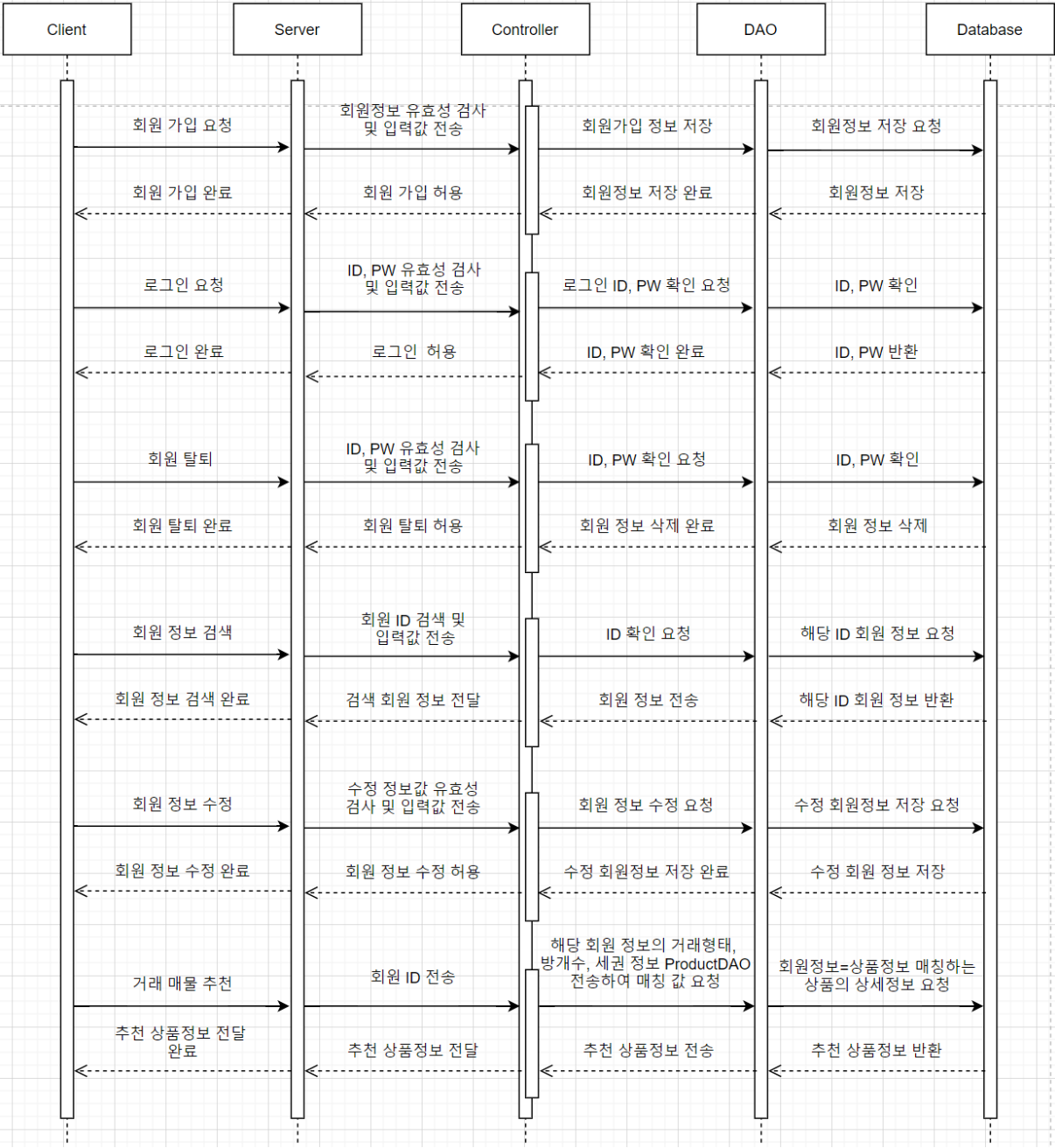
2. 시스템 구성도 - ERD & 테이블 항목

RUSER
ABC USERID
ABC USERPW
ABC USERNAME
ABC USETEL
ABC USEREMAIL
ABC USERBIRTH
ABC USERGENDER
ABC USERTYPE
ABC USERROOM
ABC USERINTEREST

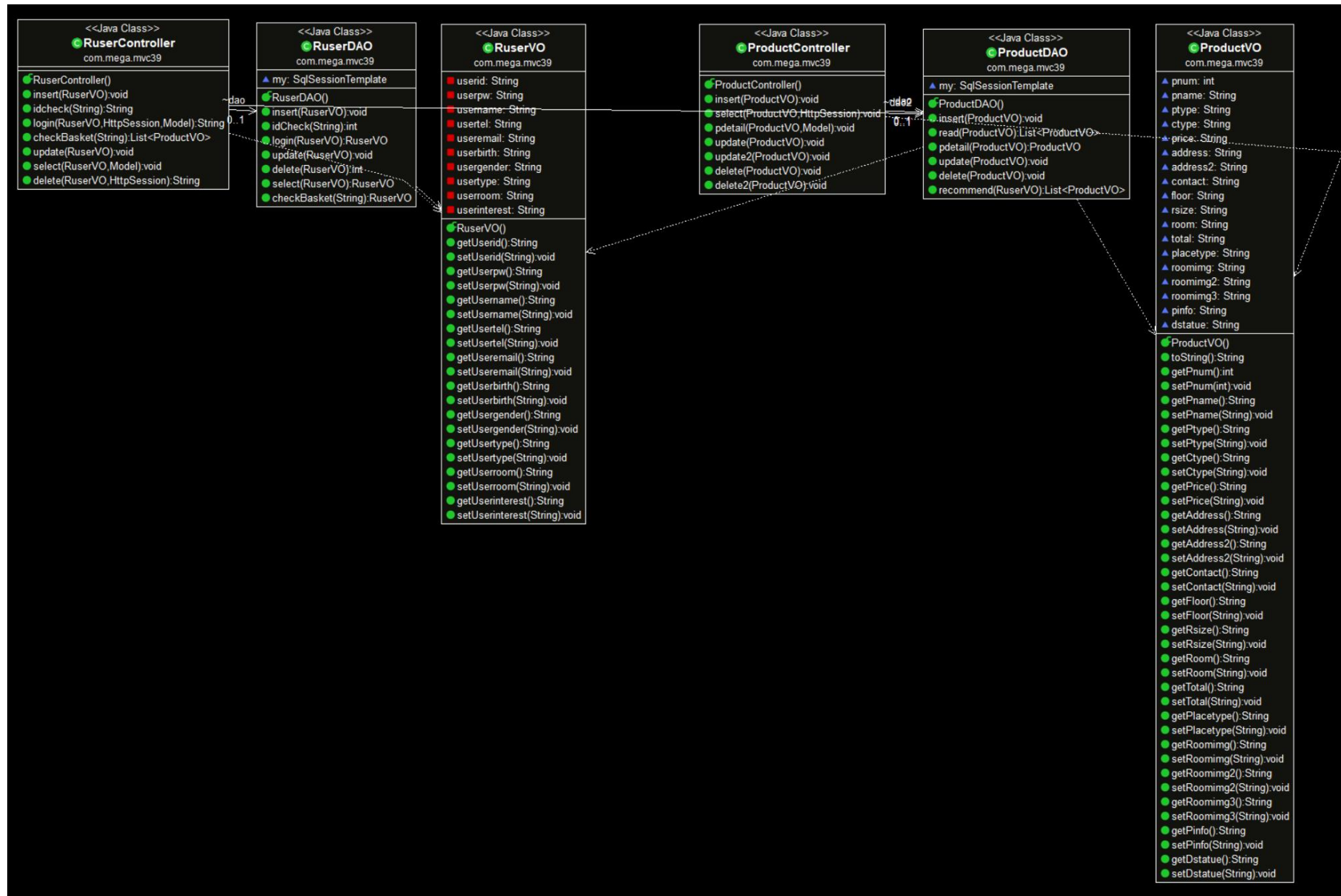
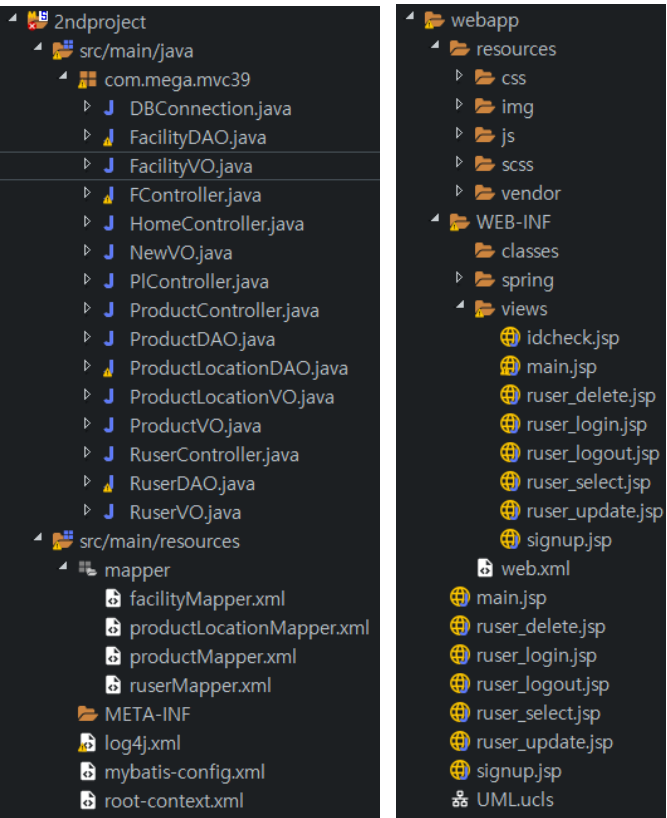
PRODUCT
ABC PNUM
ABC PNAME
ABC PTYPE
ABC CTYPE
ABC PRICE
ABC ADDRESS
ABC ADDRESS2
123 CONTACT
ABC FLOOR
123 RSIZE
ABC ROOM
123 TOTAL
ABC PLACETYPE
ABC ROOMIMG
ABC ROOMIMG2
ABC ROOMIMG3
ABC PINFO
ABC DSTATUE

컬럼명	번호	DataType	제약조건	내용
USERID	1	VARCHAR(20)	Primary key Not Null	회원 아이디
USERPW	2	VARCHAR(30)	Not Null	회원 비밀번호
USERNAME	3	VARCHAR(10)	Not Null	회원 이름
USETEL	4	VARCHAR(15)	Not Null	회원 전화번호
USEREMAIL	5	VARCHAR(30)	Not Null	회원 이메일 주소
USERBIRTH	6	VARCHAR(10)	Not Null	회원 생년월일
USERGENDER	7	VARCHAR(10)	Not Null	회원 성별
USERTYPE	8	VARCHAR(10)	Not Null	회원 거래타입
USERROOM	9	VARCHAR(10)	Not Null	회원 거래매물형태
USERINTEREST	10	VARCHAR(10)	Not Null	회원 희망 세권

2. 시스템 구성도 - Sequence Diagram & 흐름도



2. 시스템 구성도 - 프로젝트 트리 & Class Diagram



3. 회원가입 - 기술설명(회원가입)

```
@Autowired
RuserDAO dao;

//@Autowired 사용하여 RuserDAO객체 사용하기 위해 의존성 주입
//객체 생성이(new) 필요없이 객체 자동으로 생성

@RequestMapping("ruser_login")
public void insert(RuserVO bag) throws Exception {
    dao.insert(bag);
}
```

Controller

```
public void insert(RuserVO bag) throws Exception {
    // 회원가입시 브라우저에서 입력한 정보를 서버로 보내기 위한 메서드
    my.insert("ruser.insert", bag);
}
```

DAO

```
<!-- 회원가입 -->
<insert id="insert" parameterType="ruserVO">
    insert into RUSER values ({userid}, #{userpw}, #{username}, #{usertel},
    #{useremail}, #{userbirth}, #{usergender}, #{usertype}, #{userroom}, #{userinterest})
</insert>
```

mapper

○ 회원가입

Controller - @autowired 사용하여 RuserDAO 객체 사용하기 위한 의존성 주입, insert 메서드 사용하여 회원가입 정보가 담긴 bag변수를 dao로 전달

DAO - 전달받은 VO의 bag을 DB에 저장하기 위해 mapper로 전송

mapper - VO에 설정한 값들을 DB의 RUSER 테이블에 저장

회원 가입

root

.....

.....

루트

01011112222

root@root.com

19890528

남자

매매

아파트

역세권

> ANONYMOUS

> APEX_040000

> APEX_PUBLIC_USER

> APPLE

> APPQOSSYS

> CJ

> CTXSYS

> DBSNMP

> DIP

> FLOWS_FILES

> GOOD

> HR

> MDSYS

> ORACLE_OCM

> OUTLN

> ROOT

> Tables

> BBS 64K

> MEMBER 64K

> MEMBER1 64K

> MEMBER2 64K

> MEMBER3 64K

> PERSON 128K

> PRODUCT 64K

> RUSER 64K

> Views

ABC USERID	ABC USERPW	ABC USERNAME	ABC USERTEL	ABC USEREMAIL
root	Asd1234!!	루트	01011112222	root@root.com

ABC USERBIRTH	ABC USERGENDER	ABC USERTYPE	ABC USERROOM	ABC USERINTEREST
19890528	남자	매매	아파트	역세권

○ 회원가입 완료 및 DB 저장

회원가입 완료 후, 입력된 정보가 Oracle 데이터베이스의 RUSER테이블에 저장되었는지 확인

3. 회원가입 - 기술설명(회원가입-ID 유효성 검사 및 중복확인)

회원 가입

아이디

영문 대소문자, 숫자 4~12자리로 구성되어야 합니다.

○ ID 유효성 검사

(Success) jsp에서 ID 정규식 변수 regex설정,
입력 값이 null이 아닐 경우 회원가입 허용
(Fail) id 값, 정규식을 벗어날 경우 에러 메시지 표출

```
// id중복확인 및 유효성 검사
$('#id').blur(function() {
    id = $('#id').val()
    console.log(id)
    // id정규식: 영문 소문자 4~11자
    regex = /^[a-z0-9]{4,12}$/
    // 정규식 결과와 매칭하여 실행
    result = regex.exec(id)
    // 입력 id 값이 정규식에 포함될 경우 성공
    if (result != null) {
        $('#id2check').html('')
    }
    // 입력 id 값이 정규식에 벗어날 경우
    } else {
        $('#id2check').html('<h6 style=color:red>영문 대소문자, 숫자 4~12자리로
    }
}
```

회원 가입

apple

중복된 아이디입니다.

○ ID 중복 검사

(Success) ajax로 브라우저에서 중복 확인하도록
설정, 컨트롤러로 id 값을 넘겨 확인 후 실패가 아닐
경우 사용 허용
(Fail) 실패할 경우 에러 메시지 화면에 표출

```
// id 중복값 체크 기능 구현(ajax)
$.ajax({
    type: "post", // post 방식의 경우 패킷안에 숨겨져서 전송됨
                  // CUD에 사용되며, 캐시가 남지않아 보안적인 면에서 우수
    url: "idcheck",
    data : {id : id}, //컨트롤에 넘길 데이터 (userid)
    success: function(result) {
        // id 중복이 아닐 경우(성공할 경우)
        if (result != 'fail') {
            $('#idcheck').html('')
        }
        // 중복일 경우
        } else {
            $('#idcheck').html('<h6 style=color:red>중복된 아이디입니다.</h6>')
        }
    } //success
}) //ajax
```

3. 회원가입 - 기술설명(회원가입-ID중복확인)

//get은 Select기능에 사용, 캐시가 남음, 보안측면이 떨어지나
//전송속도가 우수하고 파라미터가 url에 노출됨

//post 방식의 경우 패킷안에 숨겨져서 전송됨
//CUD에 사용되며, 캐시가 남지않아 보안적인 면에서 우수
//Request body에 데이터가 들어가기 때문에
//파라미터가 노출되지 않음.

```
@RequestMapping(value="idcheck", method = RequestMethod.POST)
//http 전달 방식 post, 가상주소명 idcheck
@ResponseBody
//메서드의 return값을 HTTP reponse의 body에 담는 역할
public String idcheck(String id) throws Exception {
```

```
// int타입으로 캐스팅하여 id가 존재하면 1, 존재하지 않으면 0
int result = dao.idCheck(id);
```

```
if (result != 0) { // 0 = 아이디 중복 x, 1 = 아이디 중복 yes
    return "fail"; // 중복아이디 존재
} else {
    return "success"; //중복아이디 없음
}
```

Controller

○ ID 중복체크

Controller - @ResponseBody 사용(return 값 body에 담음), id 중복 결과값을 int 타입으로 (존재하면 1, 존재하지 않으면 0) 캐스팅하여 확인 (Binary 형태로 변경할 경우 RAM 효율성 ↑)

개인정보의 경우 method = POST 타입으로 하여 보안성 강화

```
public int idCheck(String id) {
// 회원가입시 id 중복 체크를 하기 위한 메서드
// int타입으로 캐스팅(중복 0 = 1, 중복 X = 0)
    return my.selectOne("ruser.idcheck", id);
}
```

DAO

```
<!-- idcheck 메서드의 파라미터 타입은 String, 결과값은 int -->
<select id="idcheck" parameterType="String" resultType="int">
    select count(*) from RUSER where userid = #{userid}
</select>
```

mapper

○ ID 중복체크

DAO - String 타입의 id 값이 DB에 존재하는지 확인 후,
확인된 결과를 int 타입으로 return

mapper - userid 중복여부를 int 타입으로 카운트, DAO에서 전달받은 파라미터 타입은 String, 전달할 결과값은 int로 리턴

3. 회원가입 - 기술설명(회원가입-NAME/TEL 유효성검사)

2자 이상의 한글만 입력 가능합니다.

한 자리만 입력할 경우

2자 이상의 한글만 입력 가능합니다.

영문 입력할 경우

2자 이상의 한글만 입력 가능합니다.

특수문자 입력할 경우

```
//이름 유효성검사
$('#name').blur(function() {

    //한글 2자 이상
    regex = /[가-힣]{2,}/
    //exec 함수를 사용하면 인자로 주어진 문자열 안에서 pattern을 찾음
    result = regex.exec($('#name').val())

    if (result != null) {
        $('#namecheck').html('')
    } else {
        $('#namecheck').html('<h6 style=color:red;>2자 이상의 한글만 입력
    }
})// name
```

○ NAME 유효성 검사

NAME 정규식 변수(한글 2자 이상) 설정, 입력 값이 변수에 벗어날 경우 에러 메시지 표출

올바른 번호가 아닙니다.

01(016789)아닐 경우

올바른 번호가 아닙니다.

10자리 미만일 경우

올바른 번호가 아닙니다.

11자리 초과일 경우

```
//전화번호 유효성 검사
$('#tel').blur(function() {

    // ^문자열 시작, $문자열 종료, ?앞 문자가 없거나 하나있음
    // 01x(3자로 시작), 0-9까지중 3, 4자리, 0-9까지중 4자리
    regex = /^(01[016789]{1})?([0-9]{3,4})?([0-9]{4})$/
    result = regex.exec($('#tel').val());

    if (result != null) {
        $('#telcheck').html('')
    } else {
        $('#telcheck').html('<h6 style=color:red;>올바른 번호가 아닙니다
    }
})
```

○ TEL 유효성 검사

TEL 정규식 변수(01X으로 시작, 뒷자리 7자 이상) 설정, 입력 값이 변수에서 벗어날 경우 에러 메시지 표출

3. 회원가입 - 기술설명(회원가입-PW/EMAIL)

비밀번호는 8자 이상, 숫자/대소문자/특수문자 모두 포함해야 합니다.

비밀번호가 일치하지 않습니다.

- PW 유효성 검사 및 재확인
- 1. 유효성검사
 - PW 정규식 변수(영문 대소문자, 숫자, 특수문자 포함) 설정, 입력 값이 변수에 벗어날 경우 에러 메시지 표출
- 2. 재확인
 - PW1, PW2 값이 동일하지 않을 경우 에러 메시지 표출

```
//비밀번호 유효성검사
$('#pw1').blur(function() {
    pw1 = $('#pw1').val()
    //최소 8자이상, 대소문자, 숫자 및 특수문자 각 하나씩 이상
    regex = /^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*?[!@#$%^&*~]).{8,}$/
    console.log(pw1)
    // 입력된 pw1이 비밀번호 정규식에 벗어날 경우
    if (false == regex.test(pw1)) {
        $('#pw1check').html('<h6 style=color:red>비밀번호는 8자 이상, 숫자/대소문자/특수문자 포함될 경우')
    } else {
        $('#pw1check').html('')
    }
}) //pw1

//비밀번호 확인
$('#pw2').blur(function() {
    pw1 = $('#pw1').val()
    pw2 = $('#pw2').val()
    console.log(pw1)
    console.log(pw2)
    // pw1(비밀번호 최초입력값) = pw2(비밀번호 재확인) 같은 경우
    if (pw1 == pw2) {
        $('#pw2check').html('')
    } // pw1(비밀번호 최초입력값) = pw2(비밀번호 재확인) 다를 경우
    } else {
        $('#pw2check').html('<h6 style=color:red>비밀번호가 일치하지 않습니다.</h6>')
    }
})
```

올바른 이메일 주소가 아닙니다.

올바른 이메일 주소가 아닙니다.

```
//이메일주소 유효성 검사
$('#email').blur(function() {
    //숫자, 영문대소문자 및 특수문자 사용 가능하며 중앙에 @ 필수, .뒤에
    regex = /^[0-9a-zA-Z]([-_.]?[0-9a-zA-Z])*@[0-9a-zA-Z]([-_.]?[0-9a-zA-Z])$/
    result = regex.exec($('#email').val());

    if (result != null) {
        $('#emailcheck').html('')
    } else {
        $('#emailcheck').html('<h6 style=color:red>올바른 이메일 주소가')
    }
})
```

- EMAIL 유효성 검사
- EMAIL 정규식 변수(숫자, 영문 대소문자, 특수문자) 설정, 입력 값이 변수에 벗어날 경우 에러 메시지 표출

3. 회원가입 - 기술설명(회원가입-BIRTH)

18000528

생년월일을 확인해주세요.

1900년 미만 입력할 경우

19891328

생년월일을 확인해주세요.

12월 초과일 경우

19890532

생년월일을 확인해주세요.

31일 초과일 경우

19890431

생년월일을 확인해주세요.

4, 6, 9, 11월이 31일 경우

```
// 생년월일 유효성 검사 초기화
birth = false;
//생년월일 유효성 검사
$('#birth').blur(function(){
    dateStr = $(this).val();
    year = Number(dateStr.substr(0,4)); // 입력한 값의 0~4자리까지 (연)
    month = Number(dateStr.substr(4,2)); // 입력한 값의 4번째 자리부터 2자리 숫자 (월)
    day = Number(dateStr.substr(6,2)); // 입력한 값 6번째 자리부터 2자리 숫자 (일)
    today = new Date(); // 날짜 변수 선언
    yearNow = today.getFullYear(); // 올해 연도 가져옴

    //생년월일 8자리 이하
    if (dateStr.length == 8) {
        // 연도의 경우 1900 보다 작거나 yearNow 보다 크다면 false를 반환
        if (1900 > year || year > yearNow){
            $('#birthcheck').html('<h6 style=color:red>생년월일을 확인해주세요.</h6>');
            //1월 미만 또는 12월 초과일 경우
        }else if (month < 1 || month > 12) {
            $('#birthcheck').html('<h6 style=color:red>생년월일을 확인해주세요.</h6>');
            //1일 미만 또는 31일 초과일 경우
        }else if (day < 1 || day > 31) {
            $('#birthcheck').html('<h6 style=color:red>생년월일을 확인해주세요.</h6>');
            //4월, 6월, 9월, 11월이 31일 경우
        }else if ((month==4 || month==6 || month==9 || month==11) && day==31) {
            $('#birthcheck').html('<h6 style=color:red>생년월일을 확인해주세요.</h6>');
        }
    }
});
```

○ BIRTH 유효성 검사

BIRTH 정규식 변수(연도, 월, 일자) 설정, today 변수 선언하고 올해 연도 확인을 위해 yearNow 변수 설정
Plus, subStr으로 문자열을 나누고 각각의 입력 값이 변수에 벗어날 경우 에러 메시지 표출

3. 회원가입 - 기술설명(회원가입-BIRTH)

19930229

윤년이지만 29일인 경우

생년월일을 확인해주세요.

198905282

생년월일 8자리 초과일 경우

생년월일을 확인해주세요.

```
//2월일 경우
}else if (month == 2) {
    //윤년설정
    //연도를 4로 나누었을 때 나머지가 0일 때와 100 or 400으로 나누었을 때 나머지 0일 때
    isleap = (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0))
    //29일 초과일 때 또는 29일 이면서 윤년이 아닐 때
    if (day>29 || (day==29 && !isleap)) {
        $('#birthcheck').html('<h6 style=color:red>생년월일을 확인해주세요.</h6>')
    }else{
        $('#birthcheck').html('');
        birth = true;
    }//end of if (day>29 || (day==29 && !isleap))
    }else{
        $('#birthcheck').html('');
        birth = true;
    }//end of if
    }else{
        //입력된 생년월일이 8자 초과할때 : false
        $('#birthcheck').html('<h6 style=color:red>생년월일을 확인해주세요.</h6>')
    }
}
}) //End of method

//생년월일 8자리 일때
if (dateStr.length == 8) {
```

○ BIRTH 유효성 검사

윤년일 경우 조건문 설정(연도를 4로 나누었을 때 나머지가 0 이거나 100 또는 400으로 나누었을 때 나머지가 0일 때)
윤년의 2월은 29일까지 있으므로 20일 초과이거나, 29일 이지만 윤년이 아닐 때 에러 메시지 표출
Last, 생년월일 숫자 길이가 8자리가 아닐 경우 에러 메시지 표출

3. 회원가입 - 기술설명(회원정보 검색)

```
// 회원 정보 검색
@RequestMapping("ruser_select")
public void select(RuserVO ruservo, Model model) {
    RuserVO vo = dao.select(ruservo);
    model.addAttribute("vo2", vo); // 검색결과 views에 데이터 리턴
}
```

Controller

```
// 회원 정보 검색
public RuserVO select(RuserVO bag) {
    return my.selectOne("ruser.select", bag);
}
```

DAO

```
<!-- 회원정보 검색 -->
<select id="select" parameterType="ruserVO" resultType="ruserVO">
    select * from RUSER where userid = #{userid} and userpw = #{userpw}
</select>
```

mapper

○ 회원정보 검색

Controller - update 메서드를 사용, 브라우저에서 입력 받은 id, pw
해당하는 회원 정보를(vo) views page에 리턴

DAO - 컨트롤러로 부터 전달받은 값에 해당하는 값을 DB에서 가져
오기 위해 mapper로 전달, 쿼리 결과로 받는 값이 하나
이므로 selectOne 사용

mapper - id와 pw값이 동일할 경우 Parameter를 view 페이지에
DB의 회원 정보 ruserVO값으로 리턴

검색한 회원정보

루트

01011112222

root@root.com

views

19890528

매매

아파트

역세권

```
<div class="register">
    <input type="text" id="name" placeholder="${vo2.username}" name="name">
    <input type="text" id="tel" placeholder="${vo2.usertel}" name="tel">
    <input type="text" id="email" placeholder="${vo2.useremail}" name="email">
    <input type="text" id="birth" placeholder="${vo2.userbirth}" name="birth">
    <input type="text" id="type" placeholder="${vo2.usertype}" name="type">
    <input type="text" id="room" placeholder="${vo2.userroom}" name="room">
    <input type="text" id="interest" placeholder="${vo2.userinterest}" name="interest">
</div>
```

○ 회원정보 검색

views - Controller에서 views 페이지로 전달받은
vo2 리턴 값으로 받아 ID에 해당하는 회원정보 값들
을 표출

3. 회원가입 - 기술설명(회원정보 수정)

```
// 회원 정보 업데이트
@RequestMapping("ruser_update")
public void update(RuserVO bag) throws Exception {
    dao.update(bag);
}
```

Controller

```
// 회원 정보 업데이트
public void update(RuserVO bag) throws Exception{
    my.update("ruser.update", bag);
}
```

DAO

```
<!-- 회원정보 수정 -->
<update id="update" parameterType="ruserVO">
    update RUSER set userpw = #{userpw}, usertel = #{usertel},
    useremail = #{useremail}, usertype = #{usertype},
    userroom = #{userroom}, userinterest = #{userinterest}
    where userid = #{userid}
</update>
```

mapper

○ 회원정보 수정 프로세스

Controller - update 메서드를 사용, 브라우저에서 입력 받은 id에 해당하는 회원 정보를(vo) views page에 리턴
DAO - 컨트롤러로 부터 전달받은 값에 해당하는 값을 DB에서 가져오기 위해 mapper로 전달
mapper - id와 pw값이 동일할 경우 Parameter를 view 페이지 데이터베이스의 회원 정보 ruserVO값으로 리턴

회원정보수정

.....

.....

01099993333

views

apple@apple.com

전세

단독/다가구

스세권

수정 전

ABC USERID	ABC USERPW	ABC USERNAME	ABC USERTEL	ABC USEREMAIL
root	Asd1234!!	루트	01011112222	root@root.com

ABC USERBIRTH	ABC USERGENDER	ABC USERTYPE	ABC USERROOM	ABC USER
19890528	남자	매매	아파트	역세권

수정 후

ABC USERID	ABC USERPW	ABC USERNAME	ABC USERTEL	ABC USEREMAIL
root	Asd1234!!	루트	01099993333	apple@apple.com

ABC USERBIRTH	ABC USERGENDER	ABC USERTYPE	ABC USERROOM	ABC USER
19890528	남자	전세	오피스텔	편세권

○ 회원정보 수정 완료 및 DB 저장
회원 정보 수정 완료 후, 변경 값 데이터베이스에 저장 확인

3. 회원가입 - 기술설명(회원탈퇴)

```
// 회원 정보 삭제
@RequestMapping("ruser_delete")
public String delete(RuserVO bag, HttpSession session) throws Exception {
// 아이디 삭제시 결과값이 1일경우 아이디 삭제, 0일 경우 삭제 x
int result = dao.delete(bag);
if(result == 1) {
// 아이디 탈퇴가 성공할 경우 세션끊음
session.invalidate();
// 탈퇴될 경우 메인페이지로 재요청
return "redirect:main.jsp";
} else {
// 탈퇴가 실패할 경우 다시 삭제 브라우저 페이지로 이동
return "redirect:ruser_delete.jsp";
}
}
```

```
// 회원 정보 삭제
public int delete(RuserVO bag) throws Exception{
return my.delete("ruser.delete", bag);
}
```

```
<!-- 회원탈퇴 -->
<delete id="delete" parameterType="ruserVO">
delete from RUSER where userid = #{userid} and userpw = #{userpw}
</delete>
```

- 회원탈퇴
Controller - ID 로그인 되었을 경우 탈퇴 가능, int 타입으로 설정, id = pw 일치할 경우 삭제 완료. 삭제 후, 로그인 세션이 끊어지고 메인 페이지로 이동, 실패할 경우 다시 회원탈퇴 브라우저로 이동
- DAO - VO 값을 파라미터, 반환 값을 int 타입(1 or 0 결과값으로 받기 위해)으로 받아 mapper로 리턴
- mapper - ID, PW 값이 일치할 경우 데이터베이스에서 해당 아이디 삭제

수정 전

ABC USERID	ABC USERPW	ABC USERNAME	ABC USERTEL	ABC USEREMAIL
apple	Asd1234!!	김애플	01044445555	apple@apple.com

ABC USERBIRTH	ABC USERGENDER	ABC USERTYPE	ABC USERROOM	ABC USER
19890528	여자	전세	오피스텔	편세권

수정 후

ABC USERID	ABC USERPW	ABC USERNAME	ABC USERTEL	ABC USEREMAIL

ABC USERBIRTH	ABC USERGENDER	ABC USERTYPE	ABC USERROOM	ABC USER

- 회원 탈퇴 완료 및 DB 저장
회원 탈퇴 완료 후, 회원정보 삭제 되었는지 데이터베이스 확인

3. 회원가입 - 기술설명(로그아웃/로그인)

○ 로그아웃

로그아웃 클릭할 경우 wepapp 페이지에서 자바 코드 사용하여 유지된 세션을 invalidate(비활성화)시킨 후, redirect 하여 main 페이지로 이동

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="UTF-8">
7 <title>Insert title here</title>
8 </head>
9 <body>
10 <!-- 로그아웃 할 경우 세션을 끊고 main 페이지로 redirect 해줌 -->
11 <% session.invalidate();
12     response.sendRedirect("main.jsp");
13 %>
14 </body>
15 </html>
```

LOGOUT

○ 로그인

Controller - 브라우저로부터 받은 ID 값을 회원정보 DAO에 vo로 리턴, null이 아닐 경우 로그인 허용
userid 값으로 세션 연결되며 main 페이지로 이동, 실패할 경우 회원가입 페이지로 이동

DAO - 유저 vo 값을 파라미터 리턴 값을 vo(회원정보) 설정하여 DB가져오기 위해 쿼리문 mapper로 리턴

mapper - ID, PW에 해당하는 값을 DB에서 가져온 후 vo 값을 리턴 하여 로그인 허용

@Autowired
ProductDAO dao2; //productDAO 싱글톤 호출

LOGIN Controller

// 넘어갈 페이지가 다르다는 점을 Spring에 알려줘야 함 : 반환값 String 타입
// RuserVO bag을 파라미터로 받고, session으로 로그인 연결, Model로 view page로 이동
// model = request.setAttribute와 같은 역할

```
@RequestMapping("main")
public String login(RuserVO bag, HttpSession session, Model model) {
    RuserVO vo = dao.login(bag); //dao, login메서드 사용 vo 변수로 정의
```

```
    if (vo != null) {
        // 회원가입 정보(거래형태, 방개수, 희망세권)와 매칭된 매물 추천을 위한 알고리즘
        // 로그인 후 views 페이지에서 추천 매물 표출
```

```
        // 매칭된 결과는 상품정보가 담긴 디비에서 매칭된 정보를 가져오고
        // dao2 recommend 메서드(ProductDAO)에 vo를 담아 mapper로 전송
        // productMapper에서 추천 매물(회원가입 정보 = 상품 정보) sql문 dao로 보냄
        // ProductDAO recommend 함수에서 매칭되는 상품 정보를 가져옴
```

```
        session.setAttribute("userid", vo.getUserid());
        //로그인 성공할 경우 userid 값으로 세션 연결
        List<ProductVO> list = dao2.recommend(vo);
        // view 페이지에 매칭된 상품 정보 결과를 전송
        model.addAttribute("list", list);
        return "main"; // 로그인 성공할 경우 main 페이지로 이동
    } else {
        return "signin"; // 로그인 실패할 경우 회원가입 페이지로 이동
```

```
    public RuserVO login(RuserVO vo) {
        // 컨트롤러로부터 받은 id, pw값이 vo와 동일할 경우 로그인 허용
        return my.selectOne("ruser.login", vo);
```

DAO

```
<select id="login" parameterType="ruserVO" resultType="ruserVO">
    select * from RUSER where userid = #{userid}
    and userpw = #{userpw}
```

mapper

3. 회원가입 - 기술설명(거래 매물 추천)

○ 콘텐츠 기반 필터링 (Contents-Based Filtering)

1) 정의

- 아이템이 유사한 것들을 연관성을 찾아 사용자에게 추천하는 것

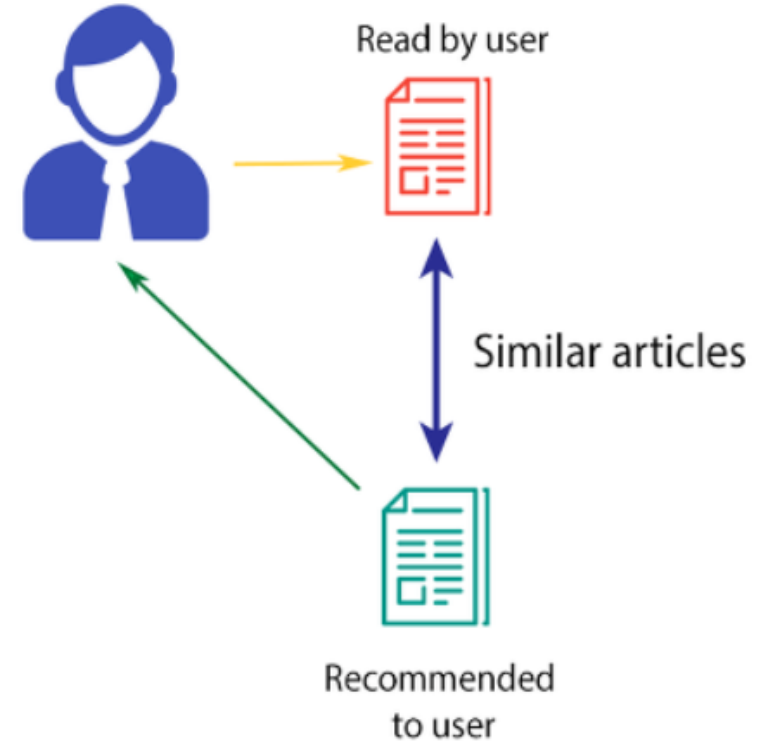
2) 기법

- TF-IDF(Tern-Frequency Inverse Document Frequency),
Term-Based(문서 유사성 찾기), Similarity 알고리즘, 문서 다중 분류 등

3) 단점

- 제품 스펙과 사용자 선호도 기반이기 때문에 덜 알려진 콘텐츠에 대한 정보 습득이 어려움
- 수작업으로 아이템 또는 콘텐츠가 입력이 되기 때문에 많은 노동력이 투입
- DB 효율화를 위한 많은 비용 발생

CONTENT-BASED FILTERING




3. 회원가입 - 기술설명(거래 매물 추천)

신촌역 추천매물 최근 검색조건으로 등록된 신규 매물입니다.

창천동

원룸, 투 · 썬리룸, 오피스텔

월세, 전세



원룸

전세 1억2000

5층, 19.83㎡, 관리비 5만
이대역 5분 lh 중기청 가능 신축 첫입주

원룸

전세 1억5000

3층, 19.83㎡, 관리비 5만
lh 중기청 가능 이대역 5분 신축 첫입주



아파트 / 137000

전세 / 84.95 / 역세권

서대문구에 위치한 아파트입니다.

아파트 / 55000

전세 / 84.9 / 역세권

서대문구에 위치한 아파트입니다.

아파트 / 60000

전세 / 84.968 / 역세권

서대문구에 위치한 아파트입니다.

○ 거래 매물 추천

- 레퍼런스 사이트(다방)의 경우 로그인 후 최근 검색한 상품 및 최근 본 상품과 연관성 있는 매물을 추천해줌

○ 실제 구현

- 콘텐츠 기반의 추천 알고리즘 사용
- 회원 가입 시 사용자가 선택한 3개의 카테고리 와 거래매물의 카테고리가 전부 일치하는 상품을 추천

* 3개 카테고리(거래형태, 룸 타입, 세권)이며, 회원가입과 상품등록시 동일한 카테고리 보유

3. 회원가입 - 기술설명(거래 매물 추천)

```
@Autowired
ProductDAO dao2; //productDAO 싱글톤 호출

// 넘어갈 페이지가 다르다는 점을 Spring에 알려줘야 함 : 반환값 String 타입
// RuserVO bag을 파라미터로 받고, session으로 로그인 연결, Model로 view page로 이동
// model = request.setAttribute와 같은 역할

@RequestMapping("main")
public String login(RuserVO bag, HttpSession session, Model model) {
    RuserVO vo = dao.login(bag); //dao, login메서드 사용 vo 변수로 정의

    if (vo != null) {
        // 회원가입 정보(거래형태, 방개수, 희망세권)와 매칭된 매물 추천을 위한 알고리즘
        // 로그인 후 views 페이지에서 추천 매물 표출

        // 매칭된 결과는 상품정보가 담긴 디비에서 매칭된 정보를 가져오고
        // dao2 recommend 메서드(ProductDAO)에 vo를 담아 mapper로 전송
        // productMapper에서 추천 매물(회원가입 정보 = 상품 정보) sql문 dao로 보냄
        // ProductDAO recommend 함수에서 매칭되는 상품 정보를 가져옴

        session.setAttribute("userid", vo.getUserid());
        //로그인 성공할 경우 userid 값으로 세션 연결
        List<ProductVO> list = dao2.recommend(vo);
        // view 페이지에 매칭된 상품 정보 결과들 전송
        model.addAttribute("list", list);
        return "main"; // 로그인 성공할 경우 main 페이지로 이동
    } else {
        return "signup"; // 로그인 실패할 경우 회원가입 페이지로 이동
    }
}
```

RuserController

DAO -회원 vo, 반환 값은 list 타입의 상품정보 vo이며 리턴 값이 복수이므로 selectList 사용하여 mapper 전송
mapper - 회원정보, 상품정보 각 3개의 값이 일치하는 상품의 정보를 리턴

// 상품 추천

```
public List<ProductVO> recommend(RuserVO bag) {
    return my.selectList("product.recommend", bag);
}
```

ProductDAO

```
<select id="recommend" parameterType="ruserVO" resultType="productVO">
    select * from PRODUCT where ctype = #{usertype}
    and ptype = #{userroom} and placetype = #{userinterest}
</select>
```

productmapper

○ 거래 매물 추천

Controller - 로그인 후, 상품DAO의 recommend 메서드에 회원정보가 담긴 vo를 보내서 List 타입의 list변수로 저장 거래매물 추천이 담긴 리스트를(회원정보, 상품정보의 3개 값이 일치하는 상품) view 페이지에 list로 리턴

3. 회원가입 - 기술설명(거래 매물 추천)

```
<% if(listSize >= 3){ %>
<td style = "width: 300px;">
  <a href="productdetail.jsp?pnum=${list[0].pnum}">
  <h4 class="Room_Type and Price">${list[0].ptype} / ${list[0].price}</h4>
  <h5 class="Specific_info">${list[0].ctype} / ${list[0].rsize} / ${list[0].placety}</h5>
  <h5 class="Place_info">${list[0].pinfo}</h5>
</td>
<td style = "width: 300px;">
  <a href="productdetail.jsp?pnum=${list[1].pnum}">
  <h4 class="Room_Type and Price">${list[1].ptype} / ${list[1].price}</h4>
  <h5 class="Specific_info">${list[1].ctype} / ${list[1].rsize} / ${list[1].placety}</h5>
  <h5 class="Place_info">${list[1].pinfo}</h5>
</td>
<td style = "width: 300px;">
  <a href="productdetail.jsp?pnum=${list[2].pnum}">
  <h4 class="Room_Type and Price">${list[2].ptype} / ${list[2].price}</h4>
  <h5 class="Specific_info">${list[2].ctype} / ${list[2].rsize} / ${list[2].placety}</h5>
  <h5 class="Place_info">${list[2].pinfo}</h5>
</td>
```

```
// 거래매물 추천을 위해 자바 코드를 사용하여
// 상품 정보가 담긴 ProductVO 리스트를 가져옴
// 최대 3개까지 매물을 추천할 것이기 때문에
// list의 size를 int타입의 listSize 변수로 저장
List<ProductVO> list
= (List<ProductVO>)request.getAttribute("list");
int listSize = list.size();
```

로그인 후 메인화면



아파트 / 137000
전세 / 84.95 / 역세권
서대문구에 위치한 아파트입니다.



아파트 / 55000
전세 / 84.9 / 역세권
서대문구에 위치한 아파트입니다.



아파트 / 60000
전세 / 84.968 / 역세권
서대문구에 위치한 아파트입니다.

상세화면

DMC래미안e편한세상 전용면적

역세권

주)래미안북덕방부동산중개법인
김미희(1082739244)
거래현황: 계약완료

전세 137000

84.95㎡

상품답기

상품상세설명

해당층/건물층 5 / 20층

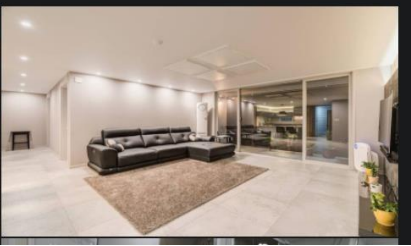
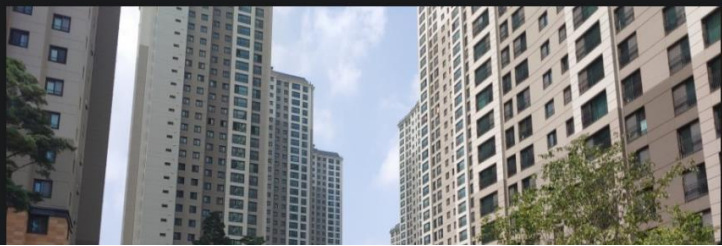
전용/공급면적 84.95㎡

방수/욕실수 3 / 2개

난방종류 개별난방

주차대수 총 1대

입주가가능 즉시입주



○ 거래매물 추천(views)

거래 매물을 페이지에 표출하기 위해, 자바 코드를 사용하여 상품 정보가 담긴 ProductVO 리스트를 호출, 최대 3개의 상품이 표출되도록 하기 위해 int타입으로 listSize 변수를 생성 회원, 상품정보 카테고리가 모두 매칭하는 상품의 개수가 최대 3개까지 표출

3. 회원가입 - 기술설명(메인, 거래 매물 추천)

```
// 메인 페이지 클릭시 매물 추천 유지                                RUserController
@RequestMapping("checkBasket")
@ResponseBody //메서드의 return값을 HTTP reponse의 body에 담는 역할

// RequestParam
// String id = request.getParameter("id"); out.print(id); 동일
// main 페이지 userid1 값을 가져옴
public List<ProductVO> checkBasket(@RequestParam(value="userid1") String userid) {

    // Ruser checkbasket 메서드의 userid에 해당하는
    // 방타입, 개수, 땡세권 값을 bag변수에 담음
    RuserVO bag = dao.checkBasket(userid);

    // ProductVO recommend 메서드 bag값을 list타입의 list 변수에 담아
    // RuserVO, ProductVO의 조건이 매칭되는
    // 상품에 대한 정보를 가져옴
    List<ProductVO> list = dao2.recommend(bag);

    // 방 이미지가 담긴 리스트를 반환값으로 넘겨줌
    return list;
}
```

```
// 메인화면 거래매물 추천                                RuserDAO
public RuserVO checkBasket(String id) {
    return my.selectOne("ruser.checkBasket" , id);
}
```

```
<!-- 메인으로 돌아갈 시 추천매물 유지해주는 매퍼 -->
<select id="checkbasket" parameterType="String" resultType="ruserVO">
    select usertype, userroom, userinterest
    from RUSER where userid = #{userid}
</select>

rusermapper
```

○ 거래매물 추천(메인페이지) *로그인 후 views 페이지에 추천 매물이 보여지나, webapp의 메인 연결의 경우 추천 매물 표출이 안됨

Controller - @ResponseBody 사용하여 ajax로 추천 매물 표출, 세션으로 연결된 아이디 값을 받기 위해
RequestParam 어노테이션으로 ID값 받아서 회원정보 vo 타입의 bag 변수에 담아 리턴

DAO - 로그인 한 ID 값을 파라미터 값으로 회원정보 vo를 mapper로 리턴

mapper - ID에 해당하는 거래타입, 방유형, 희망세권 정보를 리턴

3. 회원가입 - 기술설명(메인, 거래 매물 추천)

RuserController

```
// 메인 페이지 클릭시 매물 추천 유지
@RequestMapping("checkBasket")
@ResponseBody //메서드의 return값을 HTTP reponse의 body에 담는 역할

// RequestParam
// String id = request.getParameter("id"); out.print(id); 동일
// main 페이지 userid1 값을 가져옴
public List<ProductVO> checkBasket(@RequestParam(value="userid1") String userid) {

    // Ruser checkbasket 메서드의 userid에 해당하는
    // 방타입, 개수, 땡세권 값을 bag변수에 담음
    RuserVO bag = dao.checkBasket(userid);

    // ProductVO recommend 메서드 bag값을 list타입의 list 변수에 담아
    // RuserVO, ProductVO의 조건이 매칭되는
    // 상품에 대한 정보를 가져옴
    List<ProductVO> list = dao2.recommend(bag);

    // 방 이미지가 담긴 리스트를 반환값으로 넘겨줌
    return list;
}
```

webapp 메인 페이지

내 집은 신촌에 있나방

방찾기

마이페이지



아파트 / 137000
전세 / 84.95 / 역세권
서대문구에 위치한 아파트입니다.



아파트 / 55000
전세 / 84.9 / 역세권
서대문구에 위치한 아파트입니다.



아파트 / 60000
전세 / 84.968 / 역세권
서대문구에 위치한 아파트입니다.

// 상품 추천

```
public List<ProductVO> recommend(RuserVO bag) {
    return my.selectList("product.recommend", bag);
}
```

ProductDAO

```
<select id="recommend" parameterType="ruserVO" resultType="productVO">
    select * from PRODUCT where ctype = #{usertype}
    and ptype = #{userroom} and placetype = #{userinterest}
</select>
```

productmapper

○ 거래매물 추천(메인페이지) *로그인 후 views 페이지에 추천 매물이 보여지나, webapp의 메인 연결의 경우 추천 매물 표출이 안됨

Controller - ID에 해당하는 정보들을 담은 bag을 ProductDAO의 recommend 메서드로 보내어 로그인 후 매물추천 방식과 동일하게 파라미터 값으로 회원정보 vo를 받고 리턴 값으로 List타입 Product vo 리턴

DAO - 회원 vo, 반환 값은 list 타입의 상품정보 vo이며 리턴 값이 복수이므로 selectList 사용하여 mapper 전송

mapper - 회원정보, 상품정보 각 3개의 값이 일치하는 상품의 정보를 리턴