

MVC2 기법을 적용한 부동산 홈페이지 구현

2조 - 강지수, 신화원, 이동형, 윤정온, 최선종

목차

1. 프로젝트 개요

2. 시스템 구성도

3. 코드 소개

4. 프로젝트 소감

5. AWS(Ubuntu, window)

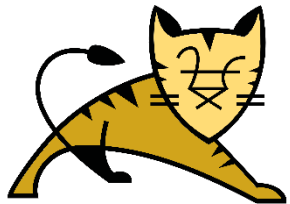
1. 프로젝트 개요

- **목 적** : 크롤링, MVC2 등 적용하여 부동산 거래 웹사이트 생성
(회원가입, 부동산 매물 정보, 매물 등록 및 관리, 마이페이지, 게시판 생성)
- **기 간** : 2021.03.15. ~ 04.15.(4주간)
- **기대효과**
 - ▲ 프로그래밍 언어 숙달
 - ▲ MVC2 구조 습득
- **결과** : 부동산 거래 홈페이지 구축 완료

2. 시스템 구성도 - 사용 프로그램



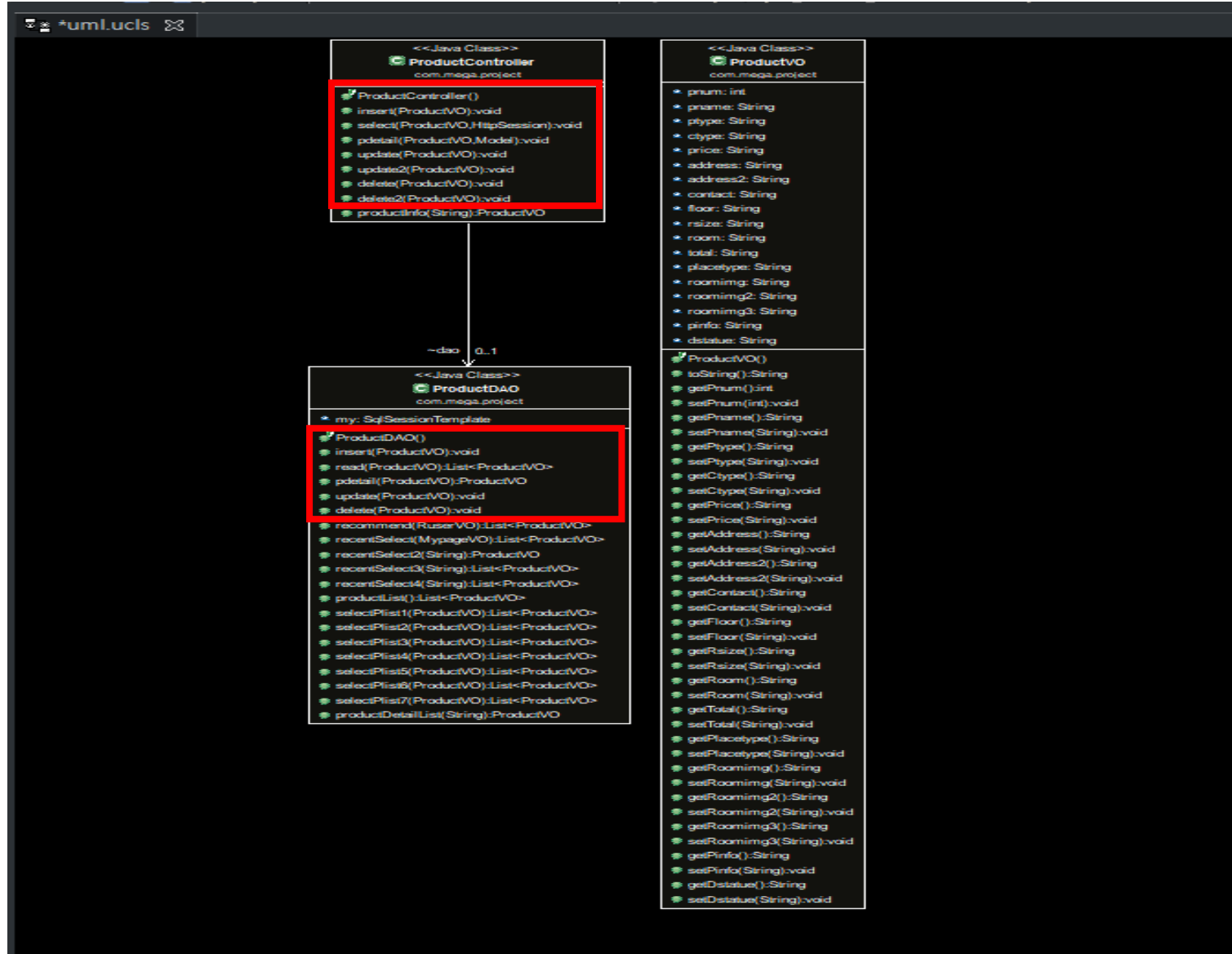
Kakao Developers Open API



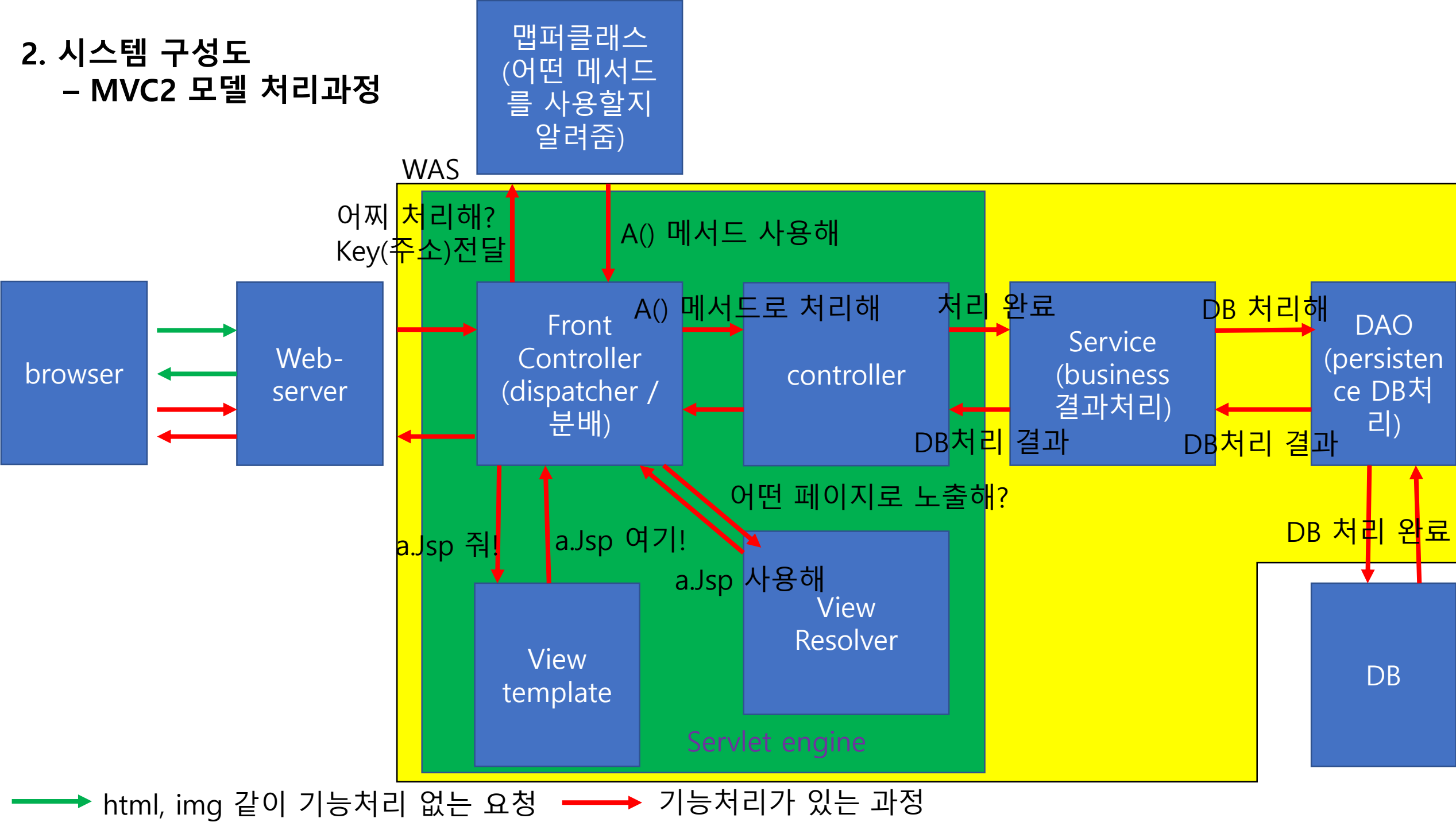
Apache Tomcat



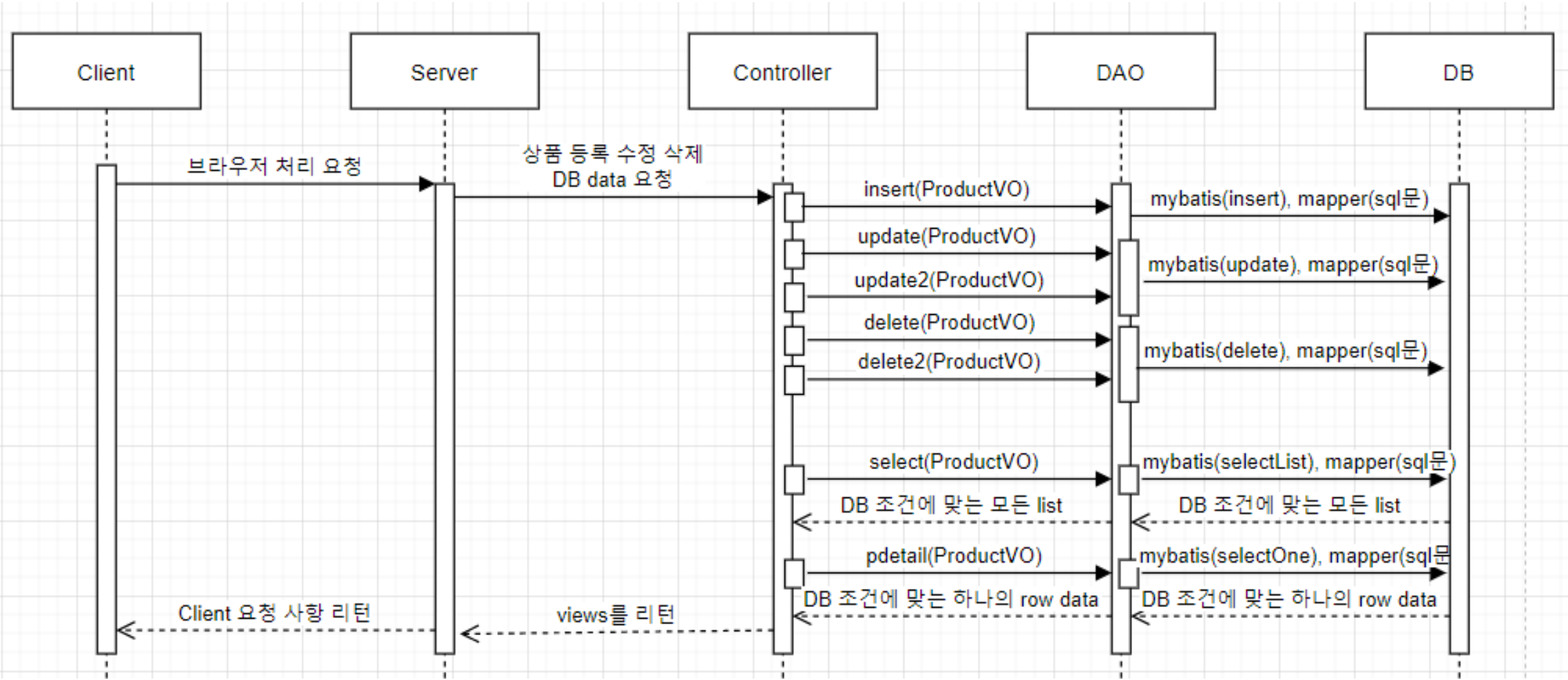
2. 시스템 구성도 - UML



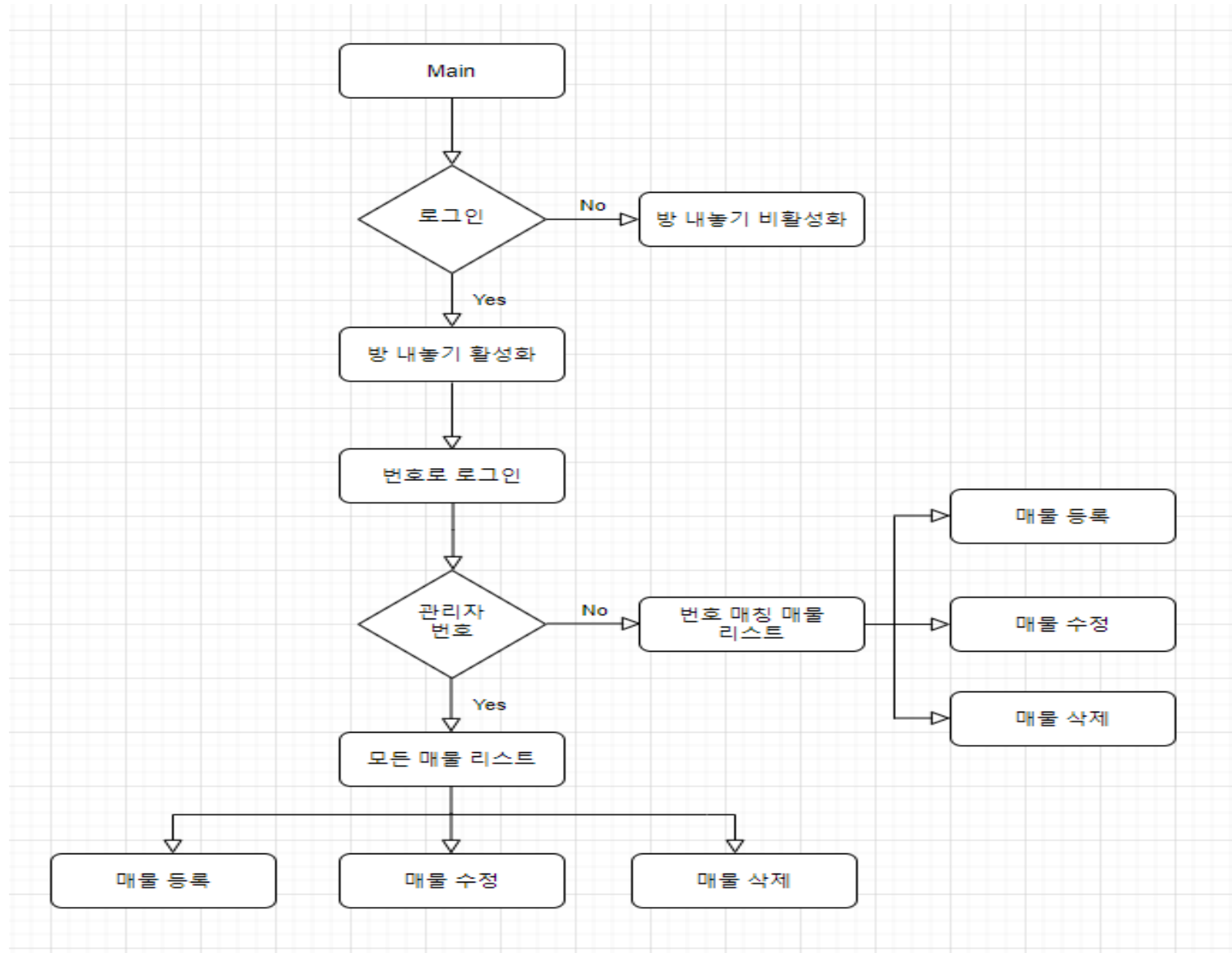
2. 시스템 구성도
- MVC2 모델 처리과정



2. 시스템 구성도 – Sequence diagram



2. 시스템 구성도 - 개발과정 순서도



2. 시스템 구성도 - Database

Table_LDH_01						
테이블 ID		PRODUCT		작성자	이동형	
테이블 설명		상품등록, 수정, 삭제 database				
번호	칼럼 ID	형식	NULL	KEY	내용	비고
1	PNUM	NUMBER	NOT NULL	PK	상품등록번호	자동 채번
2	PNAME	VARCHAR2	NOT NULL		건물이름	
3	PTYPE	VARCHAR2	NOT NULL		아파트, 오피스텔, 원룸 등	
4	CTYPE	VARCHAR2	NOT NULL		전세, 월세, 매매 등	
5	PRICE	NUMBER	NOT NULL		가격	
6	ADDRESS	VARCHAR2	NOT NULL		도로명 주소	
7	CONTACT	VARCHAR2	NOT NULL		판매자 연락처	
8	FLOOR	VARCHAR2	NOT NULL		층수(ex 10/15)	
9	RSIZE	VARCHAR2	NOT NULL		크기	
10	ROOM	VARCHAR2	NOT NULL		방갯수/화장실갯수	
11	TOTAL	VARCHAR2	NOT NULL		총 세대수	
12	PALCETYPE	VARCHAR2	NOT NULL		월세권(ex 역세권)	
13	ROOMIMG	VARCHAR2	NOT NULL		매물 사진	
14	PINFO	VARCHAR2			매물 정보	
15	DSTATUE	VARCHAR2	NOT NULL		거래 진행상태	
16	ADDRESS2	VARCHAR2	NOT NULL		상세주소	

컬럼명	#	Type	Type Mod	Not Null
123 PNUM	1	NUMBER(38,0)		[X]
ABC PNAME	2	VARCHAR2(100)		[X]
ABC PTYPE	3	VARCHAR2(100)		[X]
ABC CTYPE	4	VARCHAR2(100)		[X]
ABC PRICE	5	VARCHAR2(100)		[X]
ABC ADDRESS	6	VARCHAR2(100)		[X]
ABC CONTACT	7	VARCHAR2(100)		[X]
ABC FLOOR	8	VARCHAR2(100)		[X]
ABC RSIZE	9	VARCHAR2(100)		[X]
ABC ROOM	10	VARCHAR2(100)		[X]
ABC TOTAL	11	VARCHAR2(100)		[X]
ABC PLACETYPE	12	VARCHAR2(100)		[X]
ABC ROOMIMG	13	VARCHAR2(100)		[X]
ABC PINFO	14	VARCHAR2(500)		[]
ABC DSTATUE	15	VARCHAR2(100)		[X]
ABC ADDRESS2	16	VARCHAR2(100)		[X]

Properties

xe

이름:

PRODUCT_PK

소유자:

PRODUCT

Type:

PRIMARY KEY

Condition:

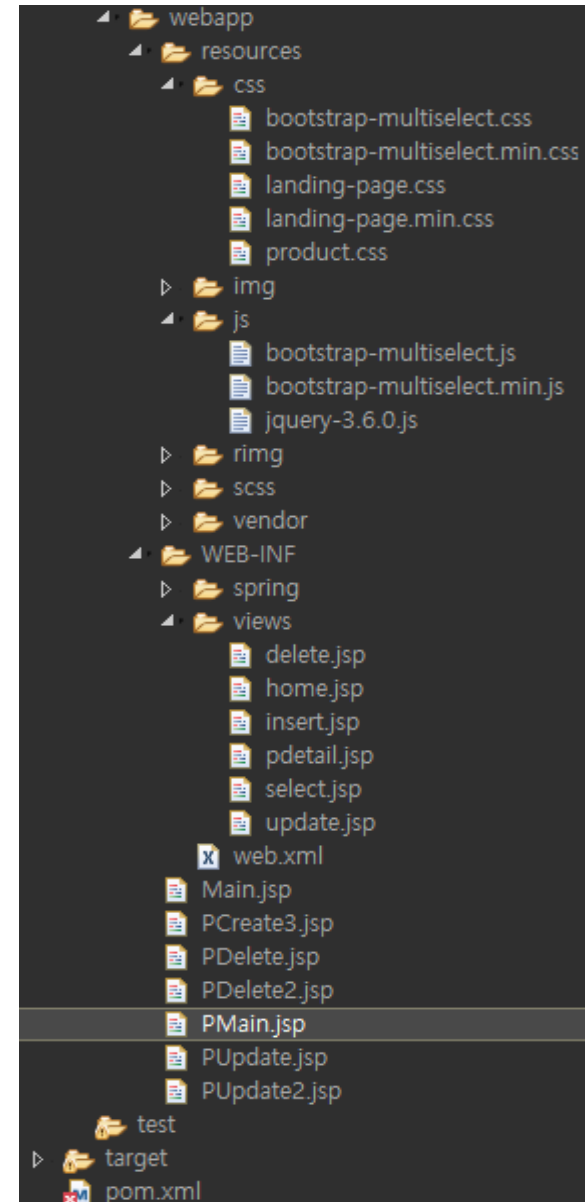
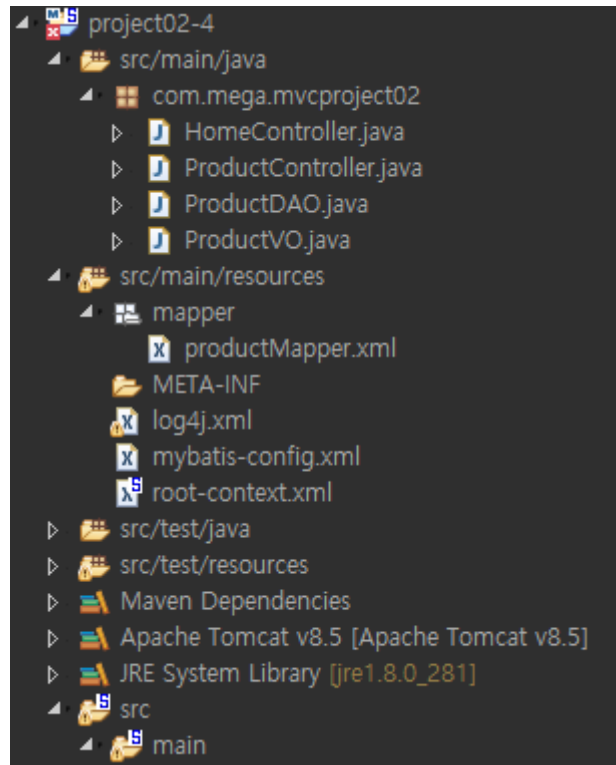
Status:

ENABLED

Constraint columns

Column 123 PNUM

2. 시스템 구성도 - 프로젝트 트리



3. 방 내놓기 - 기술설명(PMain.jsp)

localhost:8888/mvcproject02/PMain.jsp

내 집은 신촌에 있나방

방찾기

마이페이지

방내놓기

고객센터

매물거래시 연락 가능한 번호를 입력해주세요 (-생략)

010

방 내놓기

```
7 .center{ /* class가 center인 곳에 css 형식 적용 */
8   position: absolute;
9   top: 50%;
10  left: 50%;
11  /* 화면 배를 가로, 세로 상관없이 가운데 정렬을 위해서 %로 표기 */
12  /* body 기준 top과 left의 50%에 위치 */
13  transform: translate(-50%, -50%)
14  /* 시작점이 기준이기 때문에 정확하게 가운데를 맞추기 위해서 transform으로 위치 재설정 */
15 } /* PMain 페이지에 번호 입력 칸 가운데 위치 */
```

```
<div class="center">
```

```
<!-- div 설정을 통해서 css를 활용해서 div내에 위치한 태그들을 화면 가운데 위치시킨다. -->
```

```
<form action="select">
```

```
매물거래시 연락 가능한 번호를 입력해주세요 (-생략)<br>
```

```
<!-- 입력한 번호값을 controller의 가상주소 select로 보내준다. -->
```

```
<div style="margin: 0px 0px 0px 45px;">
```

```
<input name = "contact" style="text-align:center" maxlength="11">
```

```
<!-- 입력된 번호를 contact로 지정하여 받아줄때 사용한다. 전화번호는 최대 11자리까지-->
```

```
<button>방 내놓기</button>
```

```
<!-- form 안에 있는 버튼은 default값이 submit이다. -->
```

```
</div>
```

```
</form>
```

```
</div>
```

```
</body>
```

div를 class로 설정하고 css를 활용해 입력창을 가운데 위치하게 구현
번호 입력시 텍스트는 가운데 정렬을 하고 최대 전화번호 길이는 11자리까지만 입력가능
<form action>을 통해서 contact(번호)를 입력 후 버튼을 클릭
-> 컨트롤러의 가상주소 select를 통해 views의 select를 브라우저에 보여줌

3. 방 내놓기 - 기술설명(read 기능 - controller)

```
@Controller // 컨트롤러 선언
public class ProductController { // 입력값을 받아서 처리해주는 컨트롤러

    @Autowired
    ProductDAO dao;
    // 받아주는 기능 설정, 싱글톤으로 생성 후 하나의 주소값을 호출하여 계속 사용가능
    // ProductDAO의 객체를 dao로 생성
```

```
@RequestMapping("select")
// 브라우저에서 입력된 값을 가상주소 select로 보내주면 아래를 실행
public void select(ProductVO productVO, HttpSession session) {
    // 입력받은 값은 ProductVO 타입이고 productVO라는 변수명을 사용한다.
    // 세션을 사용하기 위해서 HttpSession을 import 시킨다.
    System.out.println("입력받은 번호는 : " + productVO.contact);
    // 입력받은 값 중 contact를 조건으로 사용하기 때문에 잘 받아왔는지 출력값 확인.
    List<ProductVO> list = dao.read(productVO);
    // 싱글톤인 ProductDAO의 read 메서드를 사용
    // db에 있는 여러개의 row를 가져와야해서 List 타입을 사용한다.
    // db에서 가져오는 data는 ProductVO 타입이다.
    // DAO에서 가져오는 결과값이 있으므로 List 타입의 list 변수에 넣어준다.
    System.out.println(list);
    // 가져온 dao의 리턴 값을 확인 하는 출력
    System.out.println("해당 전화번호 확인");
    session.setAttribute("userCon", list);
    // 넘겨줘서 사용할 VO 값을 따로 세션을 잡아준다.
    session.setAttribute("contact", productVO.contact);
    // 번호값을 세션으로 잡아주고 해당 번호값으로 리스트를 불러올때 사용한다.
}
```

컨트롤러로 선언해주고 ProductDAO를 싱글톤으로 만들어주고 하나의 주소값을 dao로 호출하여 사용.

PMain.jsp 에서 contact(번호)를 입력하고 버튼을 클릭시 컨트롤러의 가상주소 select로 연결이 되고 select 메서드를 시행.

입력받은 contact값을 확인해주는 출력 후 ProductDAO 의 read 메서드를 처리하고 리턴된 List타입의 값을 변수에 담아준다.

Data를 담은 후 에 list(dao.read를 통해 리턴한 data)와 contact 값을 세션처리 세션으로 다른 페이지에서 data를 활용.

3. 방 내놓기 - 기술설명(read기능 - DAO)

```
1 package com.mega.mvcproject02;
2
3 import java.util.List;
4
5 @Component
6 // 어노테이션의 기능 해당 클래스가 특정한 역할을 하게 등록한다, 싱글톤으로 만들어준다.
7 // 한개만 객체생성, 한개만 만드는데 싱글톤
8 // 같은 주소값을 계속 가져다가 사용한다.
9
10 public class ProductDAO {
11
12     @Autowired
13     SqlSessionTemplate my; // mybatis 사용하는 싱글톤
14
15 }
```

Component 선언해주고 mybatis를 싱글톤으로 만들어주고 하나의 주소값을 my로 호출하여 사용.

```
// 해당 번호로 등록된 상품 조회(read)
public List<ProductVO> read(ProductVO productVO) {
    // 컨트롤러에서 dao.read 기능을 사용하면 아래를 실행
    // mapper의 sql문을 통해 리턴되는 값이 List 형태이기 때문에 리턴 타입을 List 받아줘야한다.
    if (productVO.getContact().equals("01011111111")) { // 관리자 번호 변경 가능
        // 입력값의 ProductVO중 contact가 01011111111일때는 아래를 실행
        List<ProductVO> list = my.selectList("product.all", productVO);
        // list 변수에 mybatis에 selectList 메서드 사용
        // mapper의 namespace가 product이고 id가 all인 sql문 사용
        System.out.println("mapper 보내기 성공");
        // 보냈는지 확인하는 출력
        return list;
        // selectList로 db에서 가져온 data를 리턴해준다.
    } else { // 입력된 contact값이 01011111111이 아닐때는 아래를 실행
        List<ProductVO> list = my.selectList("product.select", productVO);
        // list 변수에 mybatis에 selectList 메서드 사용
        // mapper의 namespace가 product이고 id가 select인 sql문 사용
        System.out.println("mapper 보내기 성공");
        // 보냈는지 확인하는 출력
        return list;
        // selectList로 db에서 가져온 data를 리턴해준다.
    }
}
```

Controller에서 DAO의 read 메서드를 사용
입력받은 contac값을 조건으로 mybatis의 매퍼의 각 기능을 수행(all, select)

Mapper의 각 sql문을 통해서 리턴된 data를 List로 리턴시켜 Controller로 보내준다.
contact 조건에 따라 다른 data가 리턴된다.

3. 방 내놓기 - 기술설명(read기능 - mapper(관리자 번호))

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
3   "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
4
5 <mapper namespace="product">
6   <!-- CRUD 4가지 만들기 -->
7   <!-- product.메서드명 으로 mybatis를 사용할 수 있다. -->
8   <!-- root-context.xml의 지정된 경로로 처리해준다. -->
```

mapper namespace를 product로 지정하여 mybatis의 싱글톤 my로 해당 namespace의 id를 사용할 수 있다.

```
<select id="all" parameterType="productVO" resultType="productVO">
<!-- ProductDAO에서 id가 all인 기능을 찾을때 아래 SQL문을 실행하여 DB에서 data 리턴 -->
  select * from PRODUCT order by pnum
  <!-- ProductDAO에서 all 매퍼로 연결할때 조건 없이 모든 data를 리턴해준다. (오름차순 정렬) -->
  <!-- 리턴하는 data의 형태는 productVO라서 resultType을 꼭 지정해줘야함. -->
  <!-- 모든 상품을 볼 수 있는 관리자 번호일 때만 요청하는 mapper 기능이다. -->
</select>
```

ProductDAO에서 mybatis를 사용해 adll id를 사용하면 해당 sql문을 실행 (관리자 번호로 로그인시 실행)

sql 문이 select이므로 조건에 맞는 모든 data를 반환해준다. 그러므로 resultType을 지정 반환값은 mapper -> DAO -> Controller까지 반환되고 반환값을 가지고 views에 찍어준다.

3. 방 내놓기 - 기술설명(read기능 - mapper(관리자 번호 이외 번호))

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
3   "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
4
5 <mapper namespace="product">
6   <!-- CRUD 4가지 만들기 -->
7   <!-- product.메서드명 으로 mybatis를 사용할 수 있다. -->
8   <!-- root-context.xml의 지정된 경로로 처리해준다. -->
```

mapper namespace를 product로 지정하여 mybatis의 싱글톤 my로 해당 namespace의 id를 사용할 수 있다.

```
<select id="select" parameterType="productVO" resultType="productVO">
<!-- ProductDAO에서 id가 select인 기능을 찾을때 아래 SQL문을 실행하여 DB에서 data 리턴 -->
select * from PRODUCT where contact = #{contact} order by pnum
<!-- select문으로 contact 조건에 맞는 모든 컬럼값을 가져온다. (pnum기준으로 오름차순 정렬) -->
<!-- order by는 오름차순이 default 값으로 생략하면 오름차순 정렬 -->
<!-- contact가 동일한 data가 많기 때문에 * 사용시 모든 data를 리턴해온다.. -->
<!-- 리턴하는 row data의 형태는 productVO라서 resultType을 꼭 지정해줘야함. -->
<!-- select sql문은 반듯이 resultType가 필요하다. -->
</select>
```

ProductDAO에서 mybatis를 사용해 select id를 사용하면 해당 sql문을 실행

sql 문이 select이므로 조건에 맞는 모든 data를 반환해준다. 그러므로 resultType를 지정 반환값은 mapper -> DAO -> Controller까지 반환되고 반환값을 가지고 views에 찍어준다.

3. 방 내놓기 - 기술설명(views select.jsp)

```
<div style="margin: 0px 0px 0px 15px">
<h3>등록된 매물 현황</h3>
<!-- 등록된 매물 전체를 보여주는 페이지로 간략 정보만 노출 -->
<hr color="green">
<table>
  <tr>
    <th>등록번호</th>
    <th>매물 종류</th>
    <th>계약 종류</th>
    <th>매물 이름</th>
    <th>가격(만원)</th>
    <th>매물 크기(단위 m²)</th>
    <th>수정/삭제</th>
  </tr>
  <!-- 대표 목록은 굵은 글씨 포인트 주기 위해 th 사용 -->
</table>
```

Controller에서 views/select로 이동된 페이지
Client 에게 보여줄 브라우저를 만들어주는 페이지
Jstl로 java 구문을 사용할 수 있게 정의함
Java의 foreach 반복문을 사용해서 조건에 맞는
data를 전부 가져올 수 있다.(세션으로 잡아놓은
userCon을 활용)

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
5 <!-- jstl 사용하기 위해서 c를 정의해준다. -->
6 <c:forEach var="vo" items="${userCon}">
7   <!-- jstl의 foreach문 사용 가능! 변수를 vo로 사용하고 가져올 data정보는 userCon에서 가져온다. -->
8   <tr class="mouseon" onclick="location.href='pdetail?pnum=${vo.pnum}'">
9     <!-- tr에 class를 지정해줘서 css를 사용할 수 있다. tr영역에 마우스 올리면 색상변경 hover 사용 -->
10    <td>${vo.pnum}</td>
11    <td>${vo.pname}</td>
12    <td>${vo.ptype}</td>
13    <td>${vo.cname}</td>
14    <td>${vo.pprice}</td>
15    <td>${vo.psize}</td>
16    <td onclick="event.cancelBubble=true">
17      <!-- 수정/삭제 td에는 hover이벤트를 예외처리해서 수정과 삭제 페이지로 넘어가게 진행 -->
18      <button onclick="location.href='PUpdate.jsp?pnum=${vo.pnum}&pname=${vo.pname}&ptype=${vo.ptype}&contact=${vo.contact}'">수정</button>
19      <!-- 수정버튼 클릭시 userCon에 있는 data를 전부 PUpdate 페이지로 넘겨준다. -->
20      <button onclick="location.href='PDelete.jsp?pnum=${vo.pnum}&contact=${vo.contact}'">삭제</button>
21      <!-- 수정버튼 클릭시 userCon에 있는 pnum과 contact값을 PDelete 페이지로 넘겨준다. -->
22    </td>
23  </tr>
24 </c:forEach>
25
26 <button onclick="location.href='PCreate3.jsp?contact=${productVO.contact}'">방 내놓기</button>
27 <!-- 방 내놓기 버튼 클릭시 a태그를 사용해 PCreate 페이지 contact값을 같이 넘기고 이동한다. -->
28 <button onclick="location.href='PMain.jsp'">다른 번호 매물확인하기</button>
29 <!-- 다른 번호 매물확인하기 버튼 클릭시 a태그를 사용해 PMain 페이지로 이동 -->
30 </div>
```


3. 방 내놓기 - 기술설명(select.jsp)

등록된 매물 현황

등록번호	매물 종류	계약 종류	매물 이름	가격(만원)	매물 크기(단위 m ²)	수정/삭제
1	아파트	매매	DMC래미안클라시스	100800	84.9	수정 삭제
2	아파트	매매	DMC래미안클라시스	91500	84.9	수정 삭제
3	아파트	매매	DMC센트레빌	109500	84.98	수정 삭제
4	아파트	매매	DMC에코자이	145000	84.978	수정 삭제
20	아파트	전세	DMC래미안클라시스	55000	84.9	수정 삭제
21	아파트	전세	DMC센트럴아이파크	50400	84.95	수정 삭제
28	아파트	전세	DMC파크뷰자이1단지	60000	84.968	수정 삭제
29	아파트	전세	DMC파크뷰자이1단지	90000	84.971	수정 삭제
30	아파트	월세	DMC래미안클라시스	10000/90	84.9	수정 삭제
34	아파트	월세	DMC파크뷰자이2단지	30000/150	84.909	수정 삭제
41	오피스텔	매매	(90-58)	14000	18.98	수정 삭제
45	오피스텔	매매	신촌푸르지오시티	21200	23.38	수정 삭제
49	오피스텔	매매	신영스카이텔	17400	28.5	수정 삭제
53	오피스텔	전세	아리움	18000	15.88	수정 삭제
57	오피스텔	전세	브라운스톤연희	18900	28.79	수정 삭제
61	오피스텔	월세	SONG'S 8	10000/60	20.3	수정 삭제
65	오피스텔	월세	아리움 I	10000/65	16.89	수정 삭제
69	오피스텔	월세	썬미앙 오피스텔	9800/19	19.4	수정 삭제
73	단독/다가구	매매	단독	147000	173.6	수정 삭제
77	단독/다가구	매매	다가구	170000	145	수정 삭제
81	단독/다가구	전세	다가구	9000	15	수정 삭제
85	단독/다가구	전세	단독	20000	66.12	수정 삭제
89	단독/다가구	전세	다가구	5000	12	수정 삭제
93	단독/다가구	월세	다가구	10000/25	30	수정 삭제
97	단독/다가구	월세	다가구	12000/10	42	수정 삭제

[방 내놓기](#) [다른 번호 매물확인하기](#)

```

<c:forEach var="vo" items="${userCon}">
  <!-- list로 foreach문 사용 가능! 변수를 vo로 사용하고 가정을 data정보는 userCon에서 가져온다. -->
  <tr class="mouseon" onclick="location.href='pdetail?pnum=${vo.pnum}'">
    <!-- tr에 class를 지정해줘서 css를 사용할 수 있다. tr영역에 마우스 올리면 색상변경 hover 사용 -->
    <td>${vo.pnum}</td>
    <td>${vo.ptype}</td>
    <td>${vo.ctype}</td>
    <td>${vo.pname}</td>
    <td>${vo.price}</td>
    <td>${vo.rsize}</td>
    <td onclick="event.cancelBubble=true">
      <!-- 수정/삭제 버튼에는 hover이벤트를 메외처리해서 수정과 삭제 페이지로 넘어가게 진행 -->
      <button onclick="location.href='PUpdate.jsp?pnum=${vo.pnum}&pname=${vo.pname}&ptype=${vo.ptype}&contact=${vo.contact}'">수정</button>
      <button onclick="location.href='PDelete.jsp?pnum=${vo.pnum}&contact=${vo.contact}'">삭제</button>
      <!-- 수정버튼 클릭시 userCon에 있는 data를 전부 PUpdate 페이지로 넘겨준다. -->
      <!-- 수정버튼 클릭시 userCon에 있는 pnum과 contact값을 PDelete 페이지로 넘겨준다. -->
    </td>
  </tr>
</c:forEach>

<button onclick="location.href='PCreate.jsp?contact=${productVO.contact}'">방 내놓기</button>
<!-- 방 내놓기 버튼 클릭시 a태그를 사용해 PCreate 페이지 contact값을 같이 넘기고 이동한다. -->
<button onclick="location.href='PMain.jsp'">다른 번호 매물확인하기</button>
<!-- 다른 번호 매물확인하기 버튼 클릭시 a태그를 사용해 PMain 페이지로 이동 -->
</div>

th, td { /* views페이지의 select 페이지에서 테이블 형식으로 노출된 텍스트를 가운데 정렬 */
  text-align: center;
}

tr.mouseon:hover { background-color: gold; }
/* views의 select페이지에서 마우스를 올렸을때 배경색이 gold로 변하게 설정 */

```

Tr 영역 onclick시 location으로 페이지 이동, 수정/삭제 버튼 클릭시 이벤트 캔슬을 시켜 이동 페이지를 다르게 지정함.
하단 버튼 2개를 클릭했을때 페이지 이동은 onclick을 사용.

3. 방 내놓기 - 기술설명(create기능)

등록된 매물 현황

등록번호	매물 종류	계약 종류	매물 이름	가격(만원)	매물 크기(단위 m²)	수정/삭제
1	아파트	매매	DMC래미안클라시스	100800	84.9	수정 삭제
2	아파트	매매	DMC래미안클라시스	91500	84.9	수정 삭제
3	아파트	매매	DMC센트레빌	109500	84.98	수정 삭제
4	아파트	매매	DMC에코자이	145000	84.978	수정 삭제
20	아파트	전세	DMC래미안클라시스	55000	84.9	수정 삭제
21	아파트	전세	DMC센트럴아이파크	50400	84.95	수정 삭제
28	아파트	전세	DMC파크뷰자이1단지	60000	84.968	수정 삭제
29	아파트	전세	DMC파크뷰자이1단지	90000	84.971	수정 삭제
30	아파트	월세	DMC래미안클라시스	10000/90	84.9	수정 삭제
34	아파트	월세	DMC파크뷰자이2단지	30000/150	84.909	수정 삭제
41	오피스텔	매매	(90-58)	14000	18.98	수정 삭제
45	오피스텔	매매	신촌푸르지오시티	21200	23.38	수정 삭제
49	오피스텔	매매	신영스카이텔	17400	28.5	수정 삭제
53	오피스텔	전세	아리움	18000	15.88	수정 삭제
57	오피스텔	전세	브라운스톤연희	18900	28.79	수정 삭제
61	오피스텔	월세	SONG'S 8	1000/60	20.3	수정 삭제
65	오피스텔	월세	아리움 I	1000/65	16.89	수정 삭제
69	오피스텔	월세	썸미앙 오피스텔	9800/18	19.4	수정 삭제
73	단독/다가구	매매	단독	147000	173.6	수정 삭제
77	단독/다가구	매매	다가구	170000	145	수정 삭제
81	단독/다가구	전세	다가구	9000	15	수정 삭제
85	단독/다가구	전세	단독	20000	66.12	수정 삭제
89	단독/다가구	전세	다가구	5000	12	수정 삭제
93	단독/다가구	월세	다가구	1000/25	30	수정 삭제
97	단독/다가구	월세	다가구	12000/10	42	수정 삭제

[방 내놓기](#) [다른 번호 매물확인하기](#)

매물 등록하는 페이지 입니다.

매물 이름:

매물 종류:

계약 종류:

매물 가격(만원):

매물 주소:

연락처:

(다른 번호 등록시 등록할 번호로 로그인해 주세요)

층수(매물층/건물층):

크기(단위 m²):

방 갯수(방/화장실):

전체 세대수:

당세권(택1):

거래 상태:

매물 사진:

매물 상세정보

[등록하기](#) [돌아가기](#)

Views의 select 페이지에서 하단 방 내놓기 버튼을 클릭하면 매물 등록 페이지(PCreate3.jsp)로 이동이 되고 입력 받은 contact값을 미리 넣어준다. (GET방식)
(다른 번호 매물확인하기 클릭시 PMain.jsp으로 페이지 연결)

3. 방 내놓기 - 기술설명(create기능)

```
<div style="margin: 0px 0px 0px 15px">
<h3>매물 등록하는 페이지 입니다.</h3>
<hr color='blue'>
<form id="form"> <!-- id를 form으로 잡아놓은 이유는 js에서 serialize를 사용하기 위해서 -->
<!-- serialize를 통해서 아래 form에 잡혀 있는 모든 값을 한번에 받아서 넘길 수 있다. -->
    매물 이름 : <input name="pname"><br>
    매물 종류 : <select name="ptype">
        <option value="아파트">아파트</option>
        <option value="오피스텔">오피스텔</option>
        <option value="단독/다가구">단독/다가구</option>
    </select><br>
    계약 종류 : <select name="ctype">
        <option value="매매">매매</option>
        <option value="전세">전세</option>
        <option value="월세">월세</option>
    </select><br>
    매물 가격(만원) : <input name="price"><br>
    매물 주소 : <button type="button" onclick="execPostCode();">주소검색하기</button>
    <!-- 주소검색하기 클릭시 위의 js의 다음 주소 api를 실행한다. -->
    <input name="address" placeholder="도로명 주소" readonly><br>
    <!-- address를 readonly한 이유는 위의 주소 api를 통해서 address부분에 찍어주기 때문
    수정 할 수 없게 막아 놓는 처리 -->
    <input name="address2" placeholder="상세주소"><br>
    <!-- 도로명주소로 전체 주소를 찾고 상세주소는 직접 입력하는 address2 -->
    연락처 : <input name="contact" value="{contact}" readonly>
    <!-- 입력된 contact 값을 받아서 다른 번호로 수정 할 수 없게 막아주는 처리 -->
    <p style="font-size: 11px; color: red;">(다른 번호 등록시 등록할 번호로 로그인해 주세요)</p>
    <!-- p 태그를 사용해서 다른 번호로 등록할 경우에 대한 안내문 -->
```

매물 등록하는 페이지 입니다.

매물 이름:

매물 종류:

계약 종류:

매물 가격(만원):

매물 주소:

도로명 주소:

상세주소:

연락처:

(다른 번호 등록시 등록할 번호로 로그인해 주세요)

층수(매물층/건물층):

크기(단위㎡):

방 갯수(방/화장실):

전체 세대수:

명세권(택1):

거래 상태:

매물 사진:

매물 상세정보

```
층수(매물층/건물층) : <input name="floor"><br>
크기(단위㎡) : <input name="rsize"><br>
방 갯수(방/화장실) : <input name="room"><br>
전체 세대수 : <input name="total"><br>
명세권(택1) : <select name="placetype">
    <option value="역세권">역세권</option>
    <option value="편세권">편세권</option>
    <option value="학세권">학세권</option>
    <option value="숲세권">숲세권</option>
    <option value="스세권">스세권</option>
</select><br>
거래 상태 : <select name="dstatue">
    <option value="거래가능">거래가능</option>
    <option value="협의중">협의중</option>
    <option value="계약완료">계약완료</option>
</select><br>
매물 사진 : <input name="roomimg"><br>
매물 상세정보<br>
    <textarea name="pinfo" rows="20" cols="100"></textarea><br>
    <button type="button" id="b1">등록하기</button>
    <!-- 버튼의 id를 b1으로 하여 버튼 클릭시 js의 b1클릭 후 처리를 진행! -->
    <a href="select?contact=${contact}">
    <!-- a태그를 이용해서 돌아가기 버튼을 클릭시 contact 값을 가지고 select(가상주소)로 이동 -->
        <button type="button">돌아가기</button>
    <!-- form안에 버튼을 뒤서 브라우저에 버튼 2개가 나란하게 위치함. -->
    </a>
</form>
</div>
```

PCreate.jsp에서 상품을 등록하기 위해 각각의 입력값들을 넣는 페이지
입력된 값은 등록하기 버튼 클릭시 jquery를 통해 controller로 전송
주소검색하기 버튼 클릭시 '다음 주소 api'로 연결

3. 방 내놓기 - 기술설명(다음 주소 api)

매물 주소: 주소검색하기

도로명 주소

상세 주소

매물 주소: <button type="button" onclick="execPostCode();">주소검색하기</button>

<!-- 주소검색하기 클릭시 위의 JS의 다음 주소 api를 실행한다. -->

<input name="address" placeholder="도로명 주소" readonly>

<!-- address를 readonly한 이유는 위의 주소 api를 통해서 address부분에 찍어주기 때문
수정 할 수 없게 막아 놓는 저리 -->

<input name="address2" placeholder="상세주소">

<!-- 도로명주소로 전체 주소를 찾고 상세주소는 직접 입력하는 address2 -->

주소검색하기 버튼 클릭시 '다음 주소 api' 기능을 실행한다. '다음 주소 api'를 사용하기 위해서는 경로 지정 필요.

사용할 주소를 검색 하고 해당 결과값을 Name이 address인 input 란에 찍어준다.
(사용하지 않을 data는 주석으로 막아준다.)

```
<!-- 다음의 주소 api를 사용하기 위한 경로설정 -->
<script src="//t1.daumcdn.net/mapjsapi/bundle/postcode/prod/postcode.v2.js"></script>

execPostCode() {
    daum.Postcode({
        oncomplete: function(data) {
            // 팝업에서 검색결과 항목을 클릭했을때 실행할 코드를 작성하는 부분.

            // 도로명 주소의 노출 규칙에 따라 주소를 조합한다.
            // 내려오는 변수가 값이 없는 경우엔 공백('')값을 가지므로, 이를 참고하여 분기 한다.
            var fullRoadAddr = data.roadAddress; // 도로명 주소 변수
            var extraRoadAddr = ''; // 도로명 조합형 주소 변수

            // 법정동명이 있을 경우 추가한다. (법정리는 제외)
            // 법정동의 경우 마지막 문자가 "동/로/가"로 끝난다.
            if(data.bname !== '' && /[동|로|가]$/g.test(data.bname)){
                extraRoadAddr += data.bname;
            }
            // 건물명이 있고, 공동주택일 경우 추가한다.
            if(data.buildingName !== '' && data.apartment === 'Y'){
                extraRoadAddr += (extraRoadAddr !== '' ? ', ' + data.buildingName : data.buildingName);
            }
            // 도로명, 지번 조합형 주소가 있을 경우, 괄호까지 추가한 최종 문자열을 만든다.
            if(extraRoadAddr !== ''){
                extraRoadAddr = ' (' + extraRoadAddr + ')';
            }
            // 도로명, 지번 주소의 유무에 따라 해당 조합형 주소를 추가한다.
            if(fullRoadAddr !== ''){
                fullRoadAddr += extraRoadAddr;
            }
            // 우편번호와 주소 정보를 해당 필드에 넣는다.
            // console.log(data.zonecode);
            console.log(fullRoadAddr);
            // 내가 사용할 주소값만 놓고 나머지는 막아준다.

            // $("[name=address]").val(data.zonecode);
            $("[name=address]").val(fullRoadAddr);
            // 내가 사용할 주소값을 address라는 입력창에 넣어준다. (찍어준다.)

            /* document.getElementById('signUpUserPostNo').value = data.zonecode; //5자리 새우편번호 사용
            document.getElementById('signUpUserCompanyAddress').value = fullRoadAddr;
            document.getElementById('signUpUserCompanyAddressDetail').value = data.jibunAddress; */
        }
    });
    open();
}
```


3. 방 내놓기 - 기술설명(create기능)

매물 주소: 주소검색하기

도로명 주소

상세주소

주소 검색하기 버튼 클릭하면 해당 검색창 open

연희로 82



도로명 전체

03726

영문보기 | 지도

도로명 서울 서대문구 연희로 82 (브라운스톤연희)

지번 서울 서대문구 연희동 194-30

내외 지번주소 6건 더보기

매물 주소: 주소검색하기

서울 서대문구 연희로 82

상세주소

주소 검색 후 클릭시 자동으로
도로명 주소에 해당 결과값이
input 된다.



3. 방 내놓기 - 기술설명(create기능)

등록하기

id가 b1인 버튼
클릭 처리

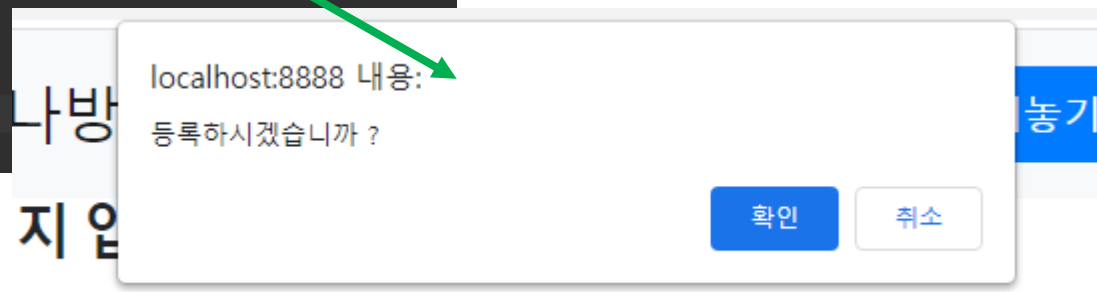
```
$(function() {  
    $("#b1").click(function() { // id가 b1인 버튼을 클릭했을때 아래를 수행한다.  
        if (confirm("등록하시겠습니까?") == true) { // 컨펌 팝업을 띄워서 등록 리체크 기능  
            var form = $("#form").serialize();  
            // serialize를 사용해서 id가 form인 부분에 있는 모든 값을 form 변수에 넣어준다.  
            $.ajax({  
                // ajax를 통해서 입력받은 form은 가상주소 insert로 보내고 alert를 통해 등록 완료 체크  
                url : "insert",  
                data : form,  
                success : function() {  
                    alert("등록이 완료되었습니다.")  
                }  
            })  
        }  
    })  
})
```

Id가 form 인 영역에 있는
모든 data를 한번에 변수에
넣어준다.(serialize사용)

```
@RequestMapping("insert")  
// 브라우저에서 입력된 값을 가상주소 insert로 보내주면 아래를 실행  
public void insert(ProductVO productVO) throws Exception {  
    // 입력받은 값은 ProductVO타입이고 productVO이라는 변수를 사용해서 찍어 줄 수 있다.  
    System.out.println(productVO);  
    // 각각의 입력받은 값 확인 productVO를 toString으로 오버라이딩(재정의)해서 입력값을 찍을 수 있다.  
    dao.insert(productVO); // ProductDAO class의 insert 메서드 실행  
    // 입력받은 productVO 변수를 dao의 insert로 보내준다.  
}
```

DAO에서 mapper 중 insert로
값을 넘겨준다.

```
// 상품 등록(create)  
public void insert(ProductVO productVO) { // 컨트롤러에서 dao.insert 기능을 사용하면 아래를 실행  
    my.insert("product.insert", productVO); // mybatis의 기능중 insert를 실행  
    // mybatis를 활용해서 mapper에 있는 namespace가 product인것에서 insert id를 가진 기능을 수행  
    System.out.println("맵퍼 보내기 성공");  
    // my.insert가 실행되는지 확인 하는 출력값  
}
```



3. 방 내놓기 - 기술설명(create기능)

```
<insert id="insert" parameterType="productVO">
<!-- ProductDAO에서 id가 insert인 기능을 찾을때 아래 SQL문을 실행하여 DB에 data 저장 -->
    insert into PRODUCT values ((select nvl(max(pnum),1)+1 from product),
    #{pname}, #{ptype}, #{ctype}, #{price}, #{address}, #{contact}, #{floor},
    #{rsize}, #{room}, #{total}, #{placetype}, #{roomimg}, #{pinfo}, #{dstatue},
    #{address2})
    <!-- 첫번째 컬럼인 pnum은 max값에 1씩 더해서 입력이 되게 자동채번 설정 다른 컬럼들은 입력값을 통해 입력 -->
</insert>
```

Mapper에서 insert sql문을 통해 입력받은 값 DB에 저장

컬럼명	#	Type	Type Mod	Not Null
123 PNUM	1	NUMBER(38,0)		[X]
ABC PNAME	2	VARCHAR2(100)		[X]
ABC PTYPE	3	VARCHAR2(100)		[X]
ABC CTYPE	4	VARCHAR2(100)		[X]
ABC PRICE	5	VARCHAR2(100)		[X]
ABC ADDRESS	6	VARCHAR2(100)		[X]
ABC CONTACT	7	VARCHAR2(100)		[X]
ABC FLOOR	8	VARCHAR2(100)		[X]
ABC RSIZE	9	VARCHAR2(100)		[X]
ABC ROOM	10	VARCHAR2(100)		[X]
ABC TOTAL	11	VARCHAR2(100)		[X]
ABC PLACETYPE	12	VARCHAR2(100)		[X]
ABC ROOMIMG	13	VARCHAR2(100)		[X]
ABC PINFO	14	VARCHAR2(500)		[]
ABC DSTATUE	15	VARCHAR2(100)		[X]
ABC ADDRESS2	16	VARCHAR2(100)		[X]

PNUM 컬럼의 값은 자동채번 기능을 사용해서 최대값에 1을 더해서 db에 insert 해주는 sql 기능을 사용

123 PNUM	ABC PNAME	ABC PTYPE	ABC CTYPE	ABC PRICE	ABC ADDRESS
1	DMC래미안클라시스	아파트	매매	100800	증가로 191
2	DMC래미안클라시스	아파트	매매	91500	증가로 191
3	DMC센트레빌	아파트	매매	109500	거북골로 120
4	DMC에코자이	아파트	매매	145000	거북골로 84
5	DMC파크뷰자이1단지	아파트	매매	147800	가재울미래로 2

Database에 저장된거 확인

컬럼명 일치 및 순서 중요

3. 방 내놓기 - 기술설명(create기능)

```
$(function() {
    $("#b1").click(function() { // id가 b1인 버튼을 클릭했을때 아래를 수행한다.
        if (confirm("등록하시겠습니까?") == true) { // 컨펌 팝업을 띄워서 등록 리제크 기능
            var form = $("#form").serialize();
            // serialize를 사용해서 id가 form인 부분에 있는 모든 값을 form 변수에 넣어준다.
            $.ajax({
                // ajax를 통해서 입력받은 form은 가상주소 insert로 보내고 alert를 통해 등록 완료 체크
                url : "insert",
                data : form,
                success : function() {
                    alert("등록이 완료되었습니다.")
                }
            })
            // ajax 실행 후 다시 select 게시판으로 돌아가는 location
            location.href = "select?contact=" + $("input[name=contact]").val();
            // 등록된 번호(contact)를 컨트롤러의 가상주소 select로 보내고 해당 contact에
            // db값을 리턴해서 views의 select가 브라우저에 보여진다.
        }
    })
})
```

ajax 처리 (매물등록) 후 location기능을 사용하여
다시 리스트가 보일 수 있도록 페이지 연결

등록된 매물 현황

등록번호	매물 종류	계약 종류	매물 이름	가격(만원)	매물 크기(단위 m ²)	수정/삭제
1	아파트	매매	DMC래미안클라시스	100800	84.9	수정 삭제
2	아파트	매매	DMC래미안클라시스	91500	84.9	수정 삭제
3	아파트	매매	DMC센트레빌	109500	84.98	수정 삭제
4	아파트	매매	DMC에코자이	145000	84.978	수정 삭제
20	아파트	전세	DMC래미안클라시스	55000	84.9	수정 삭제
21	아파트	전세	DMC센트럴아이파크	50400	84.95	수정 삭제
28	아파트	전세	DMC파크뷰자이1단지	60000	84.968	수정 삭제
29	아파트	전세	DMC파크뷰자이1단지	90000	84.971	수정 삭제
30	아파트	월세	DMC래미안클라시스	10000/90	84.9	수정 삭제
34	아파트	월세	DMC파크뷰자이2단지	30000/150	84.909	수정 삭제
41	오피스텔	매매	(90-58)	14000	18.98	수정 삭제
45	오피스텔	매매	신촌푸르지오시티	21200	23.38	수정 삭제
49	오피스텔	매매	신영스카이텔	17400	28.5	수정 삭제
53	오피스텔	전세	아리움	18000	15.88	수정 삭제
57	오피스텔	전세	브라운스톤연희	18900	28.79	수정 삭제
61	오피스텔	월세	SONG'S 8	1000/60	20.3	수정 삭제
65	오피스텔	월세	아리움 I	1000/65	16.89	수정 삭제
69	오피스텔	월세	썬미앙 오피스텔	9800/19	19.4	수정 삭제
73	단독/다가구	매매	단독	147000	173.6	수정 삭제
77	단독/다가구	매매	다가구	170000	145	수정 삭제
81	단독/다가구	전세	다가구	9000	15	수정 삭제
85	단독/다가구	전세	단독	20000	66.12	수정 삭제
89	단독/다가구	전세	다가구	5000	12	수정 삭제
93	단독/다가구	월세	다가구	1000/25	30	수정 삭제
97	단독/다가구	월세	다가구	12000/10	42	수정 삭제

[방 내놓기](#)

[다른 번호 매물확인하기](#)

3. 방 내놓기 - 기술설명(update기능)

등록된 매물 현황

등록번호	매물 종류	계약 종류	매물 이름	가격(만원)	매물 크기(단위 m ²)	수정/삭제
1	아파트	매매	DMC래미안클라시스	100800	84.9	수정 삭제
2	아파트	매매	DMC래미안클라시스	91500	84.9	수정 삭제
3	아파트	매매	DMC센트레빌	109500	84.98	수정 삭제
4	아파트	매매	DMC에코자이	145000	84.978	수정 삭제
20	아파트	전세	DMC래미안클라시스	55000	84.9	수정 삭제
21	아파트	전세	DMC센트럴아이파크	50400	84.95	수정 삭제
28	아파트	전세	DMC파크뷰자이1단지	60000	84.968	수정 삭제
29	아파트	전세	DMC파크뷰자이1단지	90000	84.971	수정 삭제
30	아파트	월세	DMC래미안클라시스	10000/90	84.9	수정 삭제
34	아파트	월세	DMC파크뷰자이2단지	30000/150	84.909	수정 삭제
41	오피스텔	매매	(90-58)	14000	18.98	수정 삭제
45	오피스텔	매매	신촌푸르지오시티	21200	23.38	수정 삭제
49	오피스텔	매매	신영스카이텔	17400	28.5	수정 삭제
53	오피스텔	전세	아리움	18000	15.88	수정 삭제
57	오피스텔	전세	브라운스톤연희	18900	28.79	수정 삭제
61	오피스텔	월세	SONG'S 8	1000/60	20.3	수정 삭제
65	오피스텔	월세	아리움 I	1000/65	16.89	수정 삭제
69	오피스텔	월세	썬미앙 오피스텔	9800/19	19.4	수정 삭제
73	단독/다가구	매매	단독	147000	173.6	수정 삭제
77	단독/다가구	매매	다가구	170000	145	수정 삭제
81	단독/다가구	전세	다가구	9000	15	수정 삭제
85	단독/다가구	전세	단독	20000	66.12	수정 삭제
89	단독/다가구	전세	다가구	5000	12	수정 삭제
93	단독/다가구	월세	다가구	1000/25	30	수정 삭제
97	단독/다가구	월세	다가구	12000/10	42	수정 삭제

[방 내놓기](#) [다른 번호 매물확인하기](#)

```
<button onclick="location.href='PUdate.jsp?pnun=${vo.pnum}&pname=${vo.pname}'">
<!-- 수정버튼 클릭시 userCon에 있는 data를 전부 PUdate 페이지로 넘겨준다. -->
```

매물 정보 수정페이지 입니다.

매물 번호:

매물 이름:

매물 종류:

계약 종류:

매물 가격(만원):

매물 주소:

연희로 82

1동 507호

연락처:

층수(매물층/건물층):

크기(단위 m²):

방 갯수(방/화장실):

전체 세대수:

당세권(택1):

거래 상태:

매물 사진:

매물 상세정보

서대문구에 위치한 오피스텔입니다.

[수정하기](#) [돌아가기](#)

PUdate.jsp로 페이지 이동시 모든 data값을 같이 넘겨줘서 수정페이지에서 받은 값을 미리 찍어줄 수 있다. 수정하고자 하는 data만 수정을 할 수 있다.

3. 방 내놓기 - 기술설명(update기능)

매물 정보 수정페이지 입니다.

매물 번호:

매물 이름:

매물 종류:

계약 종류:

매물 가격(만원):

매물 주소:

연락처:

층수(매물층/건물층):

크기(단위 m²):

방 갯수(방/화장실):

전체 세대수:

땡세권(택1):

거래 상태:

매물 사진:

매물 상세정보

서대문구에 위치한 오피스텔입니다.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%
String pnun = request.getParameter("pnun");
String pname = request.getParameter("pname");
String ptype = request.getParameter("ptype");
pageContext.setAttribute("ptype", ptype);
String ctype = request.getParameter("ctype");
pageContext.setAttribute("ctype", ctype);
String price = request.getParameter("price");
String address = request.getParameter("address");
String address2 = request.getParameter("address2");
String contact = request.getParameter("contact");
String floor = request.getParameter("floor");
String rsize = request.getParameter("rsize");
String room = request.getParameter("room");
String total = request.getParameter("total");
String placetype = request.getParameter("placetype");
pageContext.setAttribute("placetype", placetype);
String roomimg = request.getParameter("roomimg");
String pinfo = request.getParameter("pinfo");
String dstatue = request.getParameter("dstatue");
pageContext.setAttribute("dstatue", dstatue);
%>
<!-- db에 등록된 모든 값을 전부 가져오기 위해서 모든 컬럼 값을 받아준다. -->
```

Jstl을 사용하기 위해서 미리 선언을 해주고 수정버튼을 누르면서 넘겨준 모든 data(GET방식)를 변수로 받고 변수를 페이지 상에 다 찍어준다.

3. 방 내놓기 - 기술설명(update기능)

```
<h3>매물 정보 수정페이지 입니다.</h3>
<hr color="gold">
<form id="form"> <!-- id가 form인 영안에 있는 data를 serialize로 한번에 담아줄 수 있다. -->
<!-- 위에서 받아준 값을 미리 모두 넣어준다. 수정할 수 없게 막아 놓은 data들은 readonly 처리 -->
매물 번호 : <input name="pnum" value="%=pnum%" readonly><br>
매물 이름 : <input name="pname" value="%=pname%"><br>
매물 종류 : <select name="ptype">
  <option value="아파트" <c:if test="${ptype eq '아파트'}">selected="selected"</c:if> 아파트</option>
  <option value="오피스텔" <c:if test="${ptype eq '오피스텔'}">selected="selected"</c:if> 오피스텔</option>
  <option value="단독/다가구" <c:if test="${ptype eq '단독/다가구'}">selected="selected"</c:if> 단독/다가구</option>
  <!-- select 타입은 db에 입력된 값을 가져오기 위해 jstl 사용 db값을 가져와서 미리 selected 시켜놓음-->
</select><br>
계약 종류 : <select name="ctype">
  <option value="매매" <c:if test="${ctype eq '매매'}">selected="selected"</c:if> 매매</option>
  <option value="전세" <c:if test="${ctype eq '전세'}">selected="selected"</c:if> 전세</option>
  <option value="월세" <c:if test="${ctype eq '월세'}">selected="selected"</c:if> 월세</option>
</select><br>
매물 가격(만원) : <input name="price" value="%=price%"><br>
매물 주소 : <button type="button" onclick="execPostCode();" 주소검색하기</button><br>
<input name="address" value="%=address%" readonly><br>
<!-- 주소값은 다음 주소 api에서 넣어주기 때문에 readonly 처리 -->
<input name="address2" value="%=address2%"><br>
연락처 : <input name="contact" value="%=contact%" readonly><br>
층수(매물층/건물층) : <input name="floor" value="%=floor%"><br>
크기(단위㎡) : <input name="rsize" value="%=rsize%"><br>
방 갯수(방/화장실) : <input name="room" value="%=room%"><br>
전체 세대수 : <input name="total" value="%=total%"><br>
```

주소 변경시 등록하는 페이지와
동일하게 '다음 주소 api'를 활용한다.

받아준 data를 각 항목에 찍어주면 각 항목들이 들어가 있는 상태로 페이지를 열 수 있다.
select 타입의 항목들은 jstl을 사용해서 조건문 처리를 하고 해당 조건과 일치할 때
selected 처리를 해서 해당 항목으로 미리 선택되게 처리할 수 있다.
수정할 수 없는 부분은 readonly 처리해 주어 막아준다.

3. 방 내놓기 - 기술설명(update기능)

```
평세권(택1) : <select name="placetype">
    <option value="역세권" <c:if test="${placetype eq '역세권'}">selected="selected"</c:if>>역세권</option>
    <option value="편세권" <c:if test="${placetype eq '편세권'}">selected="selected"</c:if>>편세권</option>
    <option value="학세권" <c:if test="${placetype eq '학세권'}">selected="selected"</c:if>>학세권</option>
    <option value="숲세권" <c:if test="${placetype eq '숲세권'}">selected="selected"</c:if>>숲세권</option>
    <option value="스세권" <c:if test="${placetype eq '스세권'}">selected="selected"</c:if>>스세권</option>
</select>

<br>
거래 상태 : <select name="dstatue">
    <option value="거래가능" <c:if test="${dstatue eq '거래가능'}">selected="selected"</c:if>>거래가능</option>
    <option value="협의중" <c:if test="${dstatue eq '협의중'}">selected="selected"</c:if>>협의중</option>
    <option value="계약완료" <c:if test="${dstatue eq '계약완료'}">selected="selected"</c:if>>계약완료</option>
</select><br>
매물 사진 : <input name="roomimg" value="<%=roomimg%>"><br>
매물 상세정보<br>
<textarea name="pinfo" rows="20" cols="100"><%=pinfo%></textarea>
<br>
<button type="button" id="b1">수정하기</button>
<a href="select?contact=${contact}">
    <button type="button">돌아가기</button>
<!-- 돌아가기 실행시 컨트롤러의 가상주소 select로 contact값을 넘겨서 넘어가기. -->
</a>
</form>
</div>
```

id가 b1인 버튼 수정하기를 클릭하면 jquery와 ajax를 통해 해당 값들을 모두 controller로 넘겨준다. 돌아가기 버튼은 a 태그를 사용해서 controller의 select가상주소로 contact값을 보내서 views의 select 페이지로 이동한다.

3. 방 내놓기 - 기술설명(update기능)

```
$(function() {  
    $("#b1").click(function() { // id가 b1인 버튼을 클릭했을때 아래를 실행.  
        if (confirm("수정하시겠습니까 ?") == true) { // 수정하시겠습니까 체크하는 팝업  
            var form = $("#form").serialize();  
            // form이라는 변수에 id가 form인 영역의 값을 전부 저장해주는 serialize 사용  
            $.ajax({  
                // ajax를 통해서 form 변수에 들어있는 모든 data를 컨트롤러의 update(가상주소)로 전송  
                url : "update",  
                data : form,  
                success : function() {  
                    alert("수정이 완료되었습니다.")  
                    // data 전송 완료시 alert 실행  
                }  
            })  
            // ajax실행 후 다시 select페이지로 contact 값을 가지고 넘어감.  
            location.href = "select?contact=" + $("input[name=contact]").val()  
        }  
    })  
})
```

localhost:8888 내용:

수정하시겠습니까 ?

확인

취소

수정요청을 리체크 하는 confirm

ajax를 사용해서 serialize한 모든 data값을
controller의 가상주소 update로 전송해 준다.

ajax까지 처리 후 다시 상품 리스트가 보여지는 views의 select 페이지로 이동한다.

3. 방 내놓기 - 기술설명(update기능)

```
@RequestMapping("update")
// 브라우저에서 입력된 값을 가상주소 update로 보내주면 아래를 실행
public void update(ProductVO productVO) {
    System.out.println("수정할 data는 : " + productVO);
    // 입력받은 값을 bag에 넣고 toString(재정의)를 통해서 찍어주면 입력받은값 전체 확인 가능
    dao.update(productVO);
    // DAO 싱글톤의 update 메서드 사용
}
```

Ajax를 통해서 가상주소 update로 연결이 된다.
입력받아온 productVO를 DAO의 update로 보내준다.

Controller에서 보내준 값을 다시
mybatis를 통해 mapper로 보내준다.

```
// 상품 수정(update)
public void update(ProductVO productVO) { // 컨트롤러에서 dao.update 기능을 사용하면 아래를 실행
    my.update("product.update", productVO);
    // 수정할 data는 ProductVO 타입으로 입력을 받는다.
    // mybatis의 update 메서드를 통해서 mapper의 sql문 사용해서 db의 data를 수정한다.
    System.out.println("맵퍼 보내기 성공");
}
```

```
<update id="update" parameterType="productVO">
<!-- ProductDAO에서 id가 update인 기능을 찾을때 아래 SQL문을 실행하여 DB의 data를 수정한다. -->
    update PRODUCT set pname = #{pname}, ptype = #{ptype}, ctype = #{ctype},
    price = #{price}, address = #{address}, contact = #{contact}, floor = #{floor},
    rsize = #{rsize}, room = #{room}, total = #{total}, placetype = #{placetype},
    roomimg = #{roomimg}, pinfo = #{pinfo}, dstatue = #{dstatue}, address2 = #{address2}
    where pnum = #{pnum}
<!-- pnum을 조건으로 해당 row에 있는 모든 컬럼을 수정할때 사용하는 sql문 -->
<!-- select sql문이 아니라서 리턴타입이 없다. -->
</update>
```

mapper에서 sql문을 통해 db에
저장된 data를 update 처리 해준다.
pnum을 조건으로 주고 해당 row만
수정할 수 있다.

3. 방 내놓기 - 기술설명(delete기능)

등록된 매물 현황

등록번호	매물 종류	계약 종류	매물 이름	가격(만원)	매물 크기(단위㎡)	수정/삭제
1	아파트	매매	DMC래미안클라시스	100800	84.9	수정 삭제
2	아파트	매매	DMC래미안클라시스	91500	84.9	수정 삭제
3	아파트	매매	DMC센트레빌	109500	84.98	수정 삭제
4	아파트	매매	DMC에코자이	145000	84.978	수정 삭제
20	아파트	전세	DMC래미안클라시스	55000	84.9	수정 삭제
21	아파트	전세	DMC센트럴아이파크	50400	84.95	수정 삭제
28	아파트	전세	DMC파크뷰자이1단지	60000	84.968	수정 삭제
29	아파트	전세	DMC파크뷰자이1단지	90000	84.971	수정 삭제
30	아파트	월세	DMC래미안클라시스	10000/90	84.9	수정 삭제
34	아파트	월세	DMC파크뷰자이2단지	30000/150	84.909	수정 삭제
41	오피스텔	매매	(90-58)	14000	18.98	수정 삭제
45	오피스텔	매매	신촌푸르지오시티	21200	23.38	수정 삭제
49	오피스텔	매매	신영스카이텔	17400	28.5	수정 삭제
53	오피스텔	전세	아리움	18000	15.88	수정 삭제
57	오피스텔	전세	브라운스톤연희	18900	28.79	수정 삭제
61	오피스텔	월세	SONG'S 8	1000/60	20.3	수정 삭제
65	오피스텔	월세	아리움 I	1000/65	16.89	수정 삭제
69	오피스텔	월세	썸미앙 오피스텔	9800/19	19.4	수정 삭제
73	단독/다가구	매매	단독	147000	173.6	수정 삭제
77	단독/다가구	매매	다가구	170000	145	수정 삭제
81	단독/다가구	전세	다가구	9000	15	수정 삭제
85	단독/다가구	전세	단독	20000	66.12	수정 삭제
89	단독/다가구	전세	다가구	5000	12	수정 삭제
93	단독/다가구	월세	다가구	1000/25	30	수정 삭제
97	단독/다가구	월세	다가구	12000/10	42	수정 삭제

방 내놓기

다른 번호 매물확인하기

```
<button onclick="location.href='PDelete.jsp?pnum=${vo.pnum}&contact=${vo.contact}'">삭제</button>
<!-- 수정버튼 클릭시 userCon에 있는 pnum과 contact값을 PDelete 페이지로 넘겨준다. -->
```

PDelete.jsp로 페이지 이동시
pnum과 contact값을 같이 넘겨줘서
삭제페이지에서 받은 값을 찍어준다.

등록 매물 삭제 페이지 입니다.

삭제할 매물 번호를 확인해주세요 :

삭제할 매물과 연결된 번호 확인 :

삭제하기

돌아가기

삭제페이지는 data를 확인만 하므로 readonly처리

3. 방 내놓기 - 기술설명(delete기능) 등록 매물 삭제 페이지 입니다.

삭제할 매물 번호를 확인해주세요:

삭제할 매물과 연결된 번호 확인:

삭제하기 버튼은 id를 줘서 jquery로 처리
돌아가기 버튼은 a태그를 사용해서
views의 select 페이지가 보이도록 연결

```
<%@ page language="java" contentType="text/html; charset=UTF-8"%>
<%
String pnum = request.getParameter("pnum");
String contact = request.getParameter("contact");
%>
<!-- 넘겨준 pnum값과 contact값을 받아주는 처리 -->
```

```
<div style="margin: 0px 0px 0px 15px">
<h3>등록 매물 삭제 페이지 입니다.</h3>
<hr color="red">
삭제할 매물 번호를 확인해주세요 : <input id="pnum" value="<%=pnum%>" readonly><br>
삭제할 매물과 연결된 번호 확인 : <input name="contact" value="<%=contact%>" readonly><br>
<!-- 삭제할 pnum과 contact를 받아온 값으로 자동 입력시킨다. -->
<!-- 삭제할 게시를 번호와 연락처를 리체크하는 용도기 때문에 수정불가 readonly사용 -->
<button type="button" id="b1">삭제하기</button>
<a href="select?contact=${contact}">
<!-- 돌아가기 클릭시 컨트롤러의 가상주소 select로 contact 값을 가지고 이동 -->
<button>돌아가기</button>
</a>
</div>
```

넘어온 값을 받아서 해당 항목에 찍어주고 수정할 수 없게
readonly 처리
(pnum수정이 가능할 시 다른 매물이 삭제되는것 방지)

3. 방 내놓기 - 기술설명(delete기능)

```
<!-- jquery를 사용하기 위한 주소 설정 -->
<script type="text/javascript" src="resources/js/jquery-3.6.0.js"></script>
<script type="text/javascript">
    $(function() {
        $("#b1").click(function() {
            // id가 b1 이 버튼을 클릭했을때 아래를 실행
            if (confirm("<%=pnum%>번 매물을 삭제하시겠습니까 ?") == true) {
                // 매물번호를 확인 하는 팝업창 노출.
                $.ajax({ // ajax를 통해서 연결된 url로 data 보내기
                    url : "delete", // 컨트롤러의 가상주소 delete로 연결
                    data : {
                        pnum : $("#pnum").val()
                        // delete 가상주소로 넘겨줄 값은 pnum의 val(값)을 넘겨줌
                    },
                    success : function() {
                        alert("삭제가 완료되었습니다.")
                        // data를 delete url로 넘겨주면 삭제완료 alert 실행
                    }
                })
                location.href = "select?contact=" + $("input[name=contact]").val()
                // ajax 실행 후 다시 매물 리스트를 보여주는 컨트롤러의 가상주소 select로 연결! (contact값으로 등록매물 노출)
            }
        })
    })
</script>
```

localhost:8888 내용:

57번 매물을 삭제하시겠습니까 ?

확인

취소

삭제하기(id=b1) 버튼을 클릭시
리체크를 해주는 confirm을 띄움.

Ajax 실행 후 페이지를 views의 select로 연결하여 다시 등록 매물 리스트가 있는 페이지로 연결

3. 방 내놓기 - 기술설명(delete기능)

```
$.ajax({// ajax를 통해서 연결된 url로 data 보내기
  url : "delete", // 컨트롤러의 가상주소 delete로 연결
  data : {
    pnum : $("#pnum").val()
    // delete 가상주소로 넘겨줄 값은 pnum의 val(값)을 넘겨줌
  },
  success : function() {
    alert("삭제가 완료되었습니다.")
    // data를 delete url로 넘겨주면 삭제완료 alert 실행
  }
})
```

```
@RequestMapping("delete")
// 브라우저에서 입력된 값을 가상주소 delete로 보내주면 아래를 실행
public void delete(ProductVO productVO) {
  System.out.println("삭제하려는 상품 번호는 : " + productVO.pnum);
  // 입력받은 상품번호 확인하는 출력
  dao.delete(productVO); // DAO에 있는 delete 실행 DAO는 매퍼파일에서 delete 실행
}
```

ajax에서 넘어온 값을 받아서 다시 DAO로 전송

```
// 상품 삭제(delete)
public void delete(ProductVO productVO) {// 컨트롤러에서 dao.delete 기능을 사용하면 아래를 실행
  // 입력받은 번호값을 mybatis를 통해서 mapper로 전송
  my.delete("product.delete", productVO);
  // mapper의 sql문을 통해서 db의 data를 삭제한다.
  System.out.println("매퍼 보내기 성공");
}
```

DAO에서 mybatis를 사용해 mapper의 id가 delete인 곳으로 전송

```
<delete id="delete" parameterType="productVO">
<!-- ProductDAO에서 id가 delete 기능을 찾을때 아래 SQL문을 실행하여 DB에서 data 삭제 -->
  delete from PRODUCT where pnum = #{pnum}
  <!-- pnum을 조건으로 row data를 삭제하는 sql문 -->
</delete>
```

mapper에서 sql문을 통해 db에 들어있는 data를 삭제 조건을 pnum으로 주고 조건에 해당하는 row 값만 삭제

3. 방 내놓기 - 기술설명(read기능)

등록된 매물 현황

등록번호	매물 종류	계약 종류	매물 이름	가격(만원)	매물 크기(단위 m ²)	수정/삭제
1	아파트	매매	DMC래미안클라시스	100800	84.9	수정 삭제
2	아파트	매매	DMC래미안클라시스	91500	84.9	수정 삭제
3	아파트	매매	DMC센트레빌	109500	84.98	수정 삭제
4	아파트	매매	DMC에코자이	145000	84.978	수정 삭제
20	아파트	전세	DMC래미안클라시스	55000	84.9	수정 삭제
21	아파트	전세	DMC센트럴아이파크	50400	84.95	수정 삭제
28	아파트	전세	DMC파크뷰자이1단지	60000	84.968	수정 삭제
29	아파트	전세	DMC파크뷰자이1단지	90000	84.971	수정 삭제
30	아파트	월세	DMC래미안클라시스	10000/90	84.9	수정 삭제
34	아파트	월세	DMC파크뷰자이2단지	30000/150	84.909	수정 삭제
41	오피스텔	매매	(90-58)	14000	18.98	수정 삭제
45	오피스텔	매매	신촌푸르지오시티	21200	23.38	수정 삭제
49	오피스텔	매매	신영스카이텔	17400	28.5	수정 삭제
53	오피스텔	전세	아리움	18000	15.88	수정 삭제
57	오피스텔	전세	브라운스톤연희	18900	28.79	수정 삭제
61	오피스텔	월세	SONG'S 8	1000/60	20.3	수정 삭제
65	오피스텔	월세	아리움 I	1000/65	16.89	수정 삭제
69	오피스텔	월세	썬미앙 오피스텔	9800/19	19.4	수정 삭제
73	단독/다가구	매매	단독	147000	173.6	수정 삭제
77	단독/다가구	매매	다가구	170000	145	수정 삭제
81	단독/다가구	전세	다가구	9000	15	수정 삭제
85	단독/다가구	전세	단독	20000	66.12	수정 삭제
89	단독/다가구	전세	다가구	5000	12	수정 삭제
93	단독/다가구	월세	다가구	1000/25	30	수정 삭제
97	단독/다가구	월세	다가구	12000/10	42	수정 삭제

[방 내놓기](#) [다른 번호 매물확인하기](#)

```

<c:forEach var="vo" items="${userCon}">
  <!-- istl로 foreach문 사용 가능! 변수를 vo로 사용하고 가져올 data정보는 userCon에서 가져온다. -->
  <tr class="mouseon" onclick="location.href='pdetail?pnum=${vo.pnum}'">
    <!-- tr에 class를 지정해줘서 css를 사용할 수 있다. tr영역에 마우스 올리면 색상변경 hover 사용 -->
    <td>${vo.pnum}</td>
    <td>${vo.ptype}</td>
    <td>${vo.ctype}</td>
    <td>${vo.pname}</td>
    <td>${vo.price}</td>
    <td>${vo.rsize}</td>
    <td onclick="event.cancelBubble=true">
      <!-- 수정/삭제 td에는 hover이벤트를 예외처리해서 수정과 삭제 페이지로 넘어가게 진행 -->
      <button onclick="location.href='PUpdate.jsp?pnum=${vo.pnum}&pname=${vo.pname}'">수정</button>
      <!-- 수정버튼 클릭시 userCon에 있는 data를 전부 PUpdate 페이지로 넘겨준다. -->
      <button onclick="location.href='PDelete.jsp?pnum=${vo.pnum}&contact=${vo.contact}'">삭제</button>
      <!-- 수정버튼 클릭시 userCon에 있는 pnum과 contact값을 PDelete 페이지로 넘겨준다. -->
    </td>
  </tr>
</c:forEach>

```

tr 영역 어디를 클릭해도 페이지 이동 onclick 사용
수정/삭제 버튼도 tr 영역에 있기 때문에 onclick이 적용
예외처리 하기 위해 event.cancelBubble 사용

tr 영역 클릭시 controller의 pdetail로 연결되고 pnum
값을 넘겨준다.

3. 방 내놓기 - 기술설명(read기능)

```
@RequestMapping("pdetail")
// 브라우저에서 입력된 값을 가상주소 pdetail로 보내주면 아래를 실행
public void pdetail(ProductVO productVO, Model model) {
    System.out.println("입력받은 매물 번호는 " + productVO.pnum);
    // pnum으로 삭제 처리를 하기 때문에 넘어온 매물 번호 확인하는 출력
    ProductVO productV02 = dao.pdetail(productVO);
    // dao의 pdetail 메서드를 사용해서 productV02에 넣어주기.
    // 다시 넣어주는 이유 : 방 사진 1개만 등록하면 변경되서 2개가 나올 수 있도록 하기 위해.
    String pic = productV02.getRoomimg();
    // productV02의 roomimg 컬럼을 pic 변수에 넣어준다.
    System.out.println(pic);
    String pic2 = pic.replace("1", "2");
    // pic2변수에는 pic변수에 들어있는 값 중 1을 2로 교체해서 pic2로 넣어준다.
    productV02.setRoomimg2(pic2);
    // 1 -> 2로 변경한 pic2를 productV02에 set메서드 사용해서 넣어준다.
    System.out.println(pic2);
    String pic3 = pic2.replace("2", "3");
    productV02.setRoomimg3(pic3);
    model.addAttribute("productV02", productV02);
    // 최종적으로 모델을 통해서 views에는 productV02를 넘겨준다.
    // model은 views까지만 값을 넘겨주는 역할
}
```

```
// 상품 상세 조회(read)
public ProductVO pdetail(ProductVO productVO) {
    // 컨트롤러에서 dao.pdetail 기능을 사용하면 아래를 실행
    // 리턴되는 타입이 db있는 1개의 row이기 때문에 ProductVO 타입으로 리턴이 된다.
    System.out.println("매퍼 보내기 성공");
    return my.selectOne("product.detail", productVO);
    // 리턴은 항상 맨 마지막에 위치!
    // 리턴을 mybatis의 selectOne 메서드를 사용해서 sql문을 통해서 db의 1개의 row값을 가져온다.
}
```

Controller에서 넘어온 값을 mapper로 넘겨준다.

```
<select id="detail" parameterType="productVO" resultType="productVO">
<!-- ProductDAO에서 id가 detail인 기능을 찾을때 아래 SQL문을 실행하여 DB에서 data 리턴 -->
    select * from PRODUCT where pnum = #{pnum}
    <!-- ProductDAO에서 detail 매퍼로 연결할때 조건에 맞는 모든 row data를 리턴해준다. -->
    <!-- 리턴하는 row data의 형태는 productVO라서 resultType을 꼭 지정해줘야함. -->
</select>
```

Pdetail의 특이점은 사진을 넣어주는 방식인데
상품 등록 / 수정에서 사진파일은 1개만 넣고, replace를
사용해서 서버에 있는 사진을 추가해 줄 수 있다.
최종적으로 productV02를 model로 views 페이지로 보냄.

mapper에서는 sql문을 사용해서 pnum 조건에 맞는
모든 data를 가져온다. Select 문은 리턴이 있기 때문에
resultType을 꼭 지정해 줘야한다.

3. 방 내놓기 - 기술설명(read기능)

매물 상세 페이지

등록번호 57 연락처 010	매물 종류 오피스텔 층수(매물층/전체층) 5 / 20	계약 종류 전세 매물 크기(m²) 28.79	매물 이름 브라운스톤연희 방갯수(방/화장실) 3 / 2	가격(만원) 18900 전체 세대수 500	매물 주소 연희로 82 맹세권 스세권	상세 주소 1동 507호 진행상태 협의중
--------------------------	--	-----------------------------------	---	----------------------------------	-------------------------------	---------------------------------

매물사진



매물 정보

서대문구에 위치한 오피스텔입니다.

- 수정
- 삭제
- 돌아가기

최종 views의 pdetail 페이지를 보여준다.

3. 방 내놓기 - 기술설명(read기능)

매물 상세 페이지

등록번호	매물 종류	계약 종류	매물 이름	가격(만원)	매물 주소	상세 주소
57	오피스텔	전세	브라운스톤연희	18900	연희로 82	1동 507호
연락처	층수(매물층/전체층)	매물 크기(m ²)	방갯수(방/화장실)	전체 세대수	땡세권	진행상태
010	5 / 20	28.79	3 / 2	500	스세권	협의중

```
<h3>매물 상세 페이지</h3>
<hr color="green">
<div class="detail">
  <table class="detail">
    <tr>
      <th>등록번호</th>
      <th>매물 종류</th>
      <th>계약 종류</th>
      <th>매물 이름</th>
      <th>가격(만원)</th>
      <th>매물 주소</th>
      <th>상세 주소</th>
    </tr>
    <tr>
      <!-- 브라우저에서 컨트롤러를 통해 넘어온 값(db에서 가져온 값(sql문))을 보여준다. -->
      <!-- ProductController에서 pdetail은 productV02를 모델로 넘겨준다. -->
      <td>${productV02.pnum}</td>
      <td>${productV02.ptype}</td>
      <td>${productV02.ctype}</td>
      <td>${productV02.pname}</td>
      <td>${productV02.price}</td>
      <td>${productV02.address}</td>
      <td>${productV02.address2}</td>
    </tr>
  </table>
</div>
```

```
<tr>
  <th>연락처</th>
  <th>층수(매물층/전체층)</th>
  <th>매물 크기(m2)</th>
  <th>방갯수(방/화장실)</th>
  <th>전체 세대수</th>
  <th>땡세권</th>
  <th>진행상태</th>
</tr>
<tr>
  <!-- 브라우저에서 컨트롤러를 통해 넘어온 값(db에서 가져온 값(sql문))을 보여준다. -->
  <!-- ProductController에서 pdetail은 productV02를 모델로 넘겨준다. -->
  <td>${productV02.contact}</td>
  <td>${productV02.floor}</td>
  <td>${productV02.rsize}</td>
  <td>${productV02.room}</td>
  <td>${productV02.total}</td>
  <td>${productV02.placetype}</td>
  <td>${productV02.dstatue}</td>
</tr>
</table>
<!-- 브라우저에 찍히는 구성을 위해 테이블 구성-->
</div>
```

controller에서 model로 넘겨준 값을 사용해서 views 페이지에 찍어 준다.

3. 방 내놓기 - 기술설명(read기능)

매물 상세 페이지

등록번호	매물 종류	계약 종류	매물 이름	가격(만원)	매물 주소	상세 주소
57	오피스텔	전세	브라운스톤연희	18900	연희로 82	1동 507호
연락처	층수(매물층/전체층)	매물 크기(m²)	방갯수(방/화장실)	전체 세대수	땡세권	진행상태
010	5 / 20	28.79	3 / 2	500	스세권	협의중

```
<div class="detail">
  <table class="detail">
    <tr>
      <th>등록번호</th>
      <th>매물 종류</th>
      <th>계약 종류</th>
      <th>매물 이름</th>
      <th>가격(만원)</th>
      <th>매물 주소</th>
      <th>상세 주소</th>
    </tr>
```

```
.detail{
  width: 100%;
}
```

Div와 table에 같은 클래스를 지정하고 css를 활용.
Width를 %로 지정해서 브라우저의 크기에
상관없이 항상 브라우저에 딱 차게 정렬 된다.

매물 정보

서대문구에 위치한 오피스텔입니다.

수정

삭제

돌아가기

수정/삭제버튼은 위에서 나온 내용과 동일함. 연결되는 가상주소만 다름.
PUpdate2.jsp, PDelete2.jsp 로 연결
수정 / 삭제 페이지에서 돌아가기 버튼 클릭시 매물 상세 페이지로 연결하기 위함.

4. 프로젝트 소감 – 이동형

두번째 프로젝트에서는 상품관리 파트를 담당하게 되었다. 상품 등록 수정 삭제를 CRUD 방식으로 구현을 계획했다. 첫번째 프로젝트에서도 CRUD를 사용했기 때문에 구현하는 기간이 오래 걸리지 않을거라고 생각을 했다. 하지만 첫 프로젝트 방식과는 다르게 MVC2 모델로 두번째 프로젝트를 구축하게 되었고 생소한 부분들이 많아 천천히 이해하며 프로젝트를 진행했다.

프로젝트를 진행하는데 있어서 기획단계가 중요하다는 생각이 많이 들었다. 기획단계에서 철저하게 준비를 해서 개발을 진행했으면 수정사항이나 중간에 추가되는 기능들이 적었을거 같다는 생각이 들었다. 특히 DB 테이블을 만드는데 있어서 컬럼이 추가 되는 상황이 많았다. 기획단계에서 철저하게 계획을 했으면 불필요한 요소를 줄일 수 있다는 생각이 들었다.

코드를 만들면서 어려웠던 점은 내가 원하는 기능을 찾아보고 활용할 때 어떤 방법으로 찾아야 하는지가 어려웠다. 구현하고 싶은 기능은 있는데 그걸 어떻게 검색을 해야할지를 몰라서 검색만 1시간을 한적도 있었다. 결국은 같은 조분들에게 물어보고 기능을 알게되고 검색을 해서 해결할 수 있었다.

두번째 어려웠던 점은 조원들과 같은 공간에서 프로젝트를 진행하지 않고 각자 집에서 프로젝트를 진행하니 서로 의견을 나누는 부분에 있어서 내용 전달이 정확하게 되지 않는다는 점이였다. 그래서 일주일에 한번 모여서 얘기하면서 프로젝트를 진행할때 안되던 많은 것들이 해결되는 경우가 있었는데 이런 경험을 통해서 소통의 중요성을 느낄 수 있었다.

두개의 프로젝트를 진행하면서 CRUD 중심으로 진행했다. 다음 프로젝트에서는 새로운 기능들을 사용해 보고 싶다.

Ubuntu(putty, 파워셸, cmd Winscp) 연결 과정

5. AWS

AWS Management Console

AWS 서비스

▼ 최근 방문한 서비스



Billing

▼ 전체 서비스



컴퓨팅

EC2

Lightsail

Lambda

Batch

Elastic Beanstalk

Serverless Application

Repository

AWS Outposts

EC2 Image Builder



Machine Learning

Amazon SageMaker

Amazon Augmented AI

Amazon CodeGuru

Amazon DevOps Guru

Amazon Comprehend

Amazon Forecast

Amazon Fraud Detector

Amazon Kendra

Amazon Lex

성공적으로 시작됨 i-0e3f8a61a656faaa2,i-0f8bd9efdf6269292

인스턴스 (2/2) 정보

연결

인스턴스 상태

작업

인스턴스 시작

인스턴스 필터링

<input checked="" type="checkbox"/>	Name	인스턴스 ID	인스턴스 상태	인스턴스 유형	상태 검사
<input checked="" type="checkbox"/>	ubuntu18	i-0e3f8a61a656faaa2	실행 중	t2.micro	-
<input checked="" type="checkbox"/>	win2012server	i-0f8bd9efdf6269292	실행 중	t2.micro	초기화

인스턴스에 연결 정보

다음 옵션 중 하나를 사용하여 인스턴스 i-0e3f8a61a656faaa2 (ubuntu18)에 연결

EC2 인스턴스 연결

Session Manager

SSH 클라이언트

EC2 직렬 콘솔

인스턴스 ID

i-0e3f8a61a656faaa2 (ubuntu18)

1. SSH 클라이언트를 엽니다.
2. 프라이빗 키 파일을 찾습니다. 이 인스턴스를 시작하는 데 사용되는 키는 yad.pem입니다.
3. 필요한 경우 이 명령을 실행하여 키를 공개적으로 볼 수 없도록 합니다.

chmod 400 yad.pem

4. 퍼블릭 DNS을(를) 사용하여 인스턴스에 연결:

ec2-13-58-112-253.us-east-2.compute.amazonaws.com

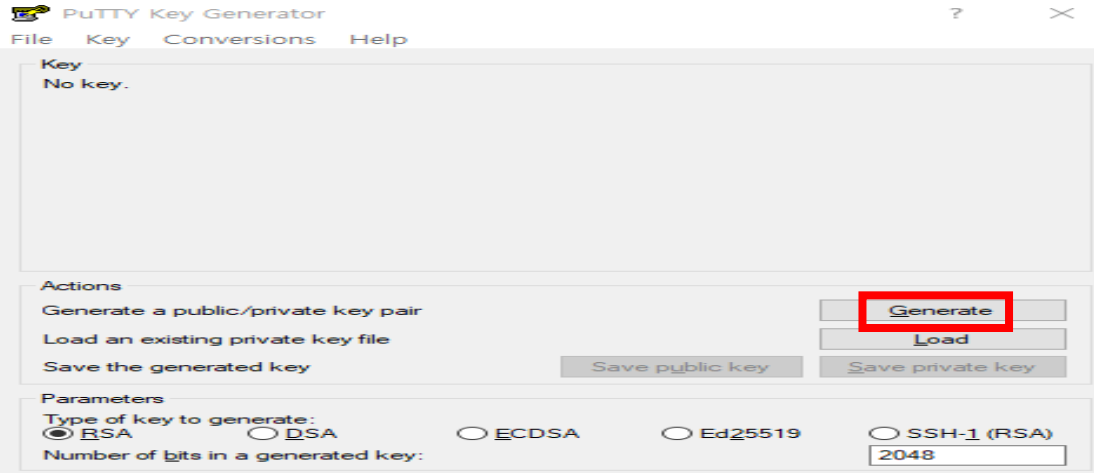
예:

ssh -i "yad.pem" ubuntu@ec2-13-58-112-253.us-east-2.compute.amazonaws.com

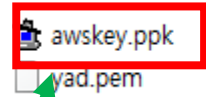
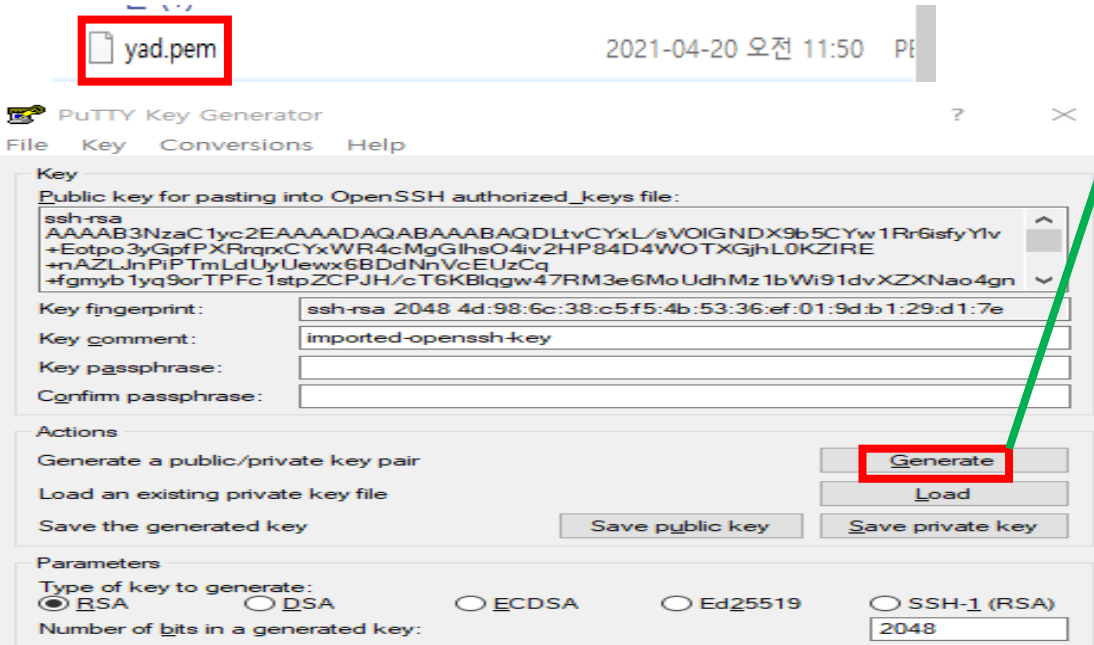
참고: 대부분의 경우 추정된 사용자 이름은 정확합니다. 하지만 AMI 사용 지침을 읽고 AMI 소유자가 기본 AMI 사용자 이름을 변경했는지 확인하십시오.

AWS의 Ubuntu 서버에 연결하기 위해서는 퍼블릭 DNS와 키가 있어야 한다.

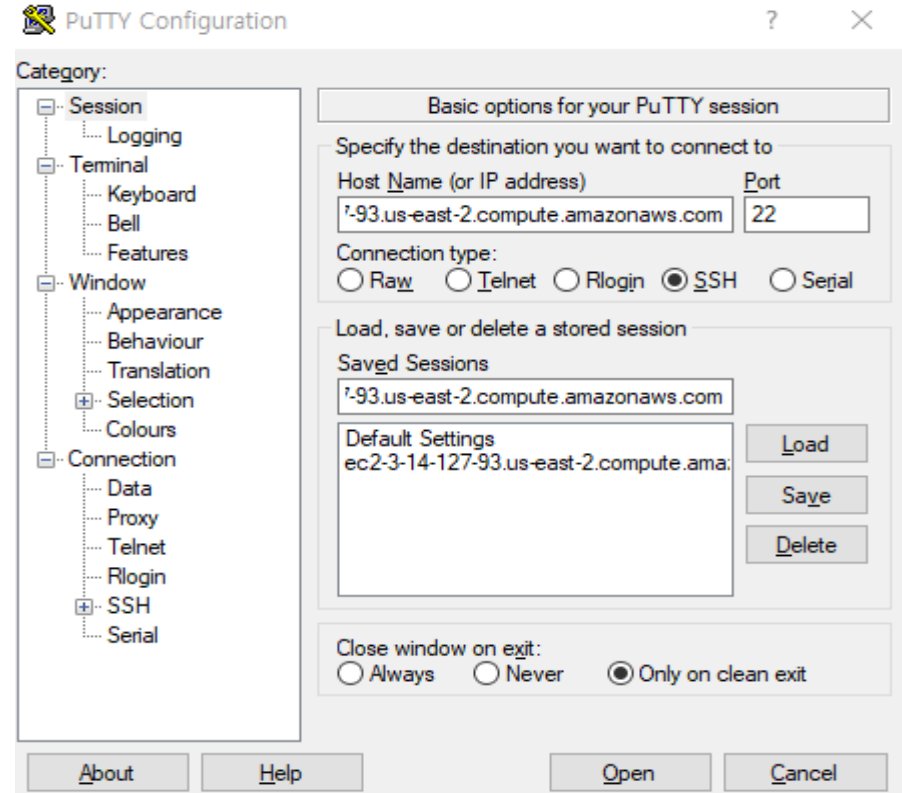
5. AWS – putty



Putty에 연결하기 위해서는 기존 key인 pem파일을 ppk파일로 변환해야한다.(puttygen 사용)

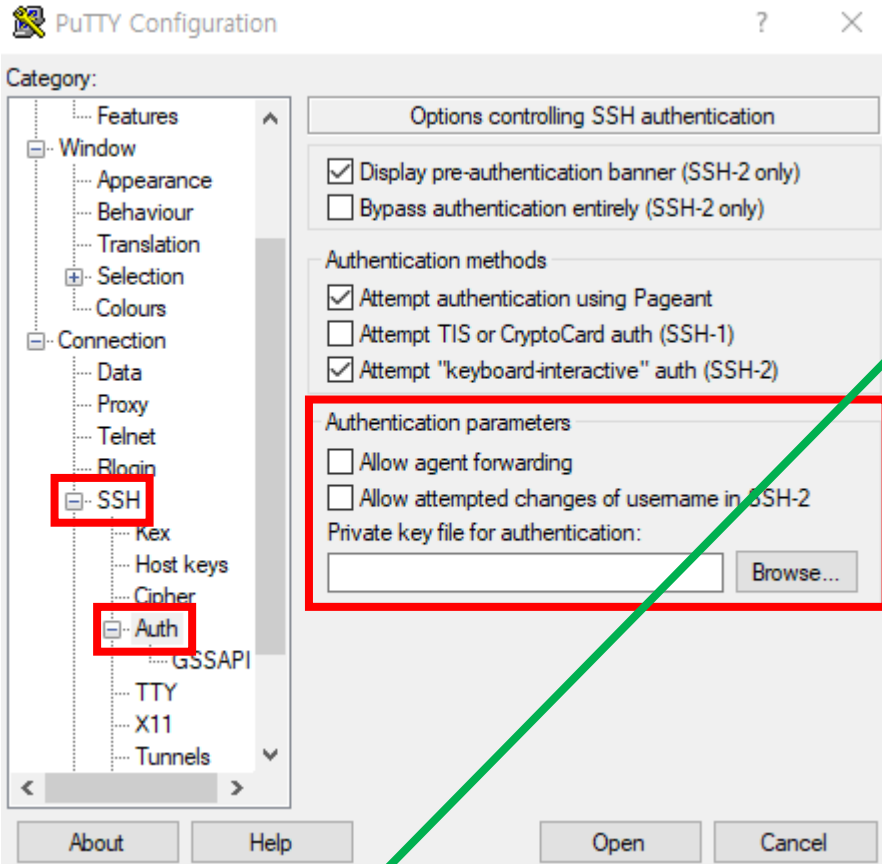


Pem을 활용하여 ppk 키를 생성한다.

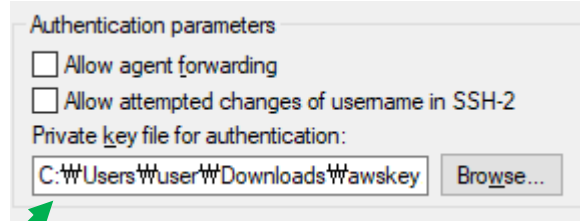


Putty 실행 후 Host와 sessions을 설정해준다.

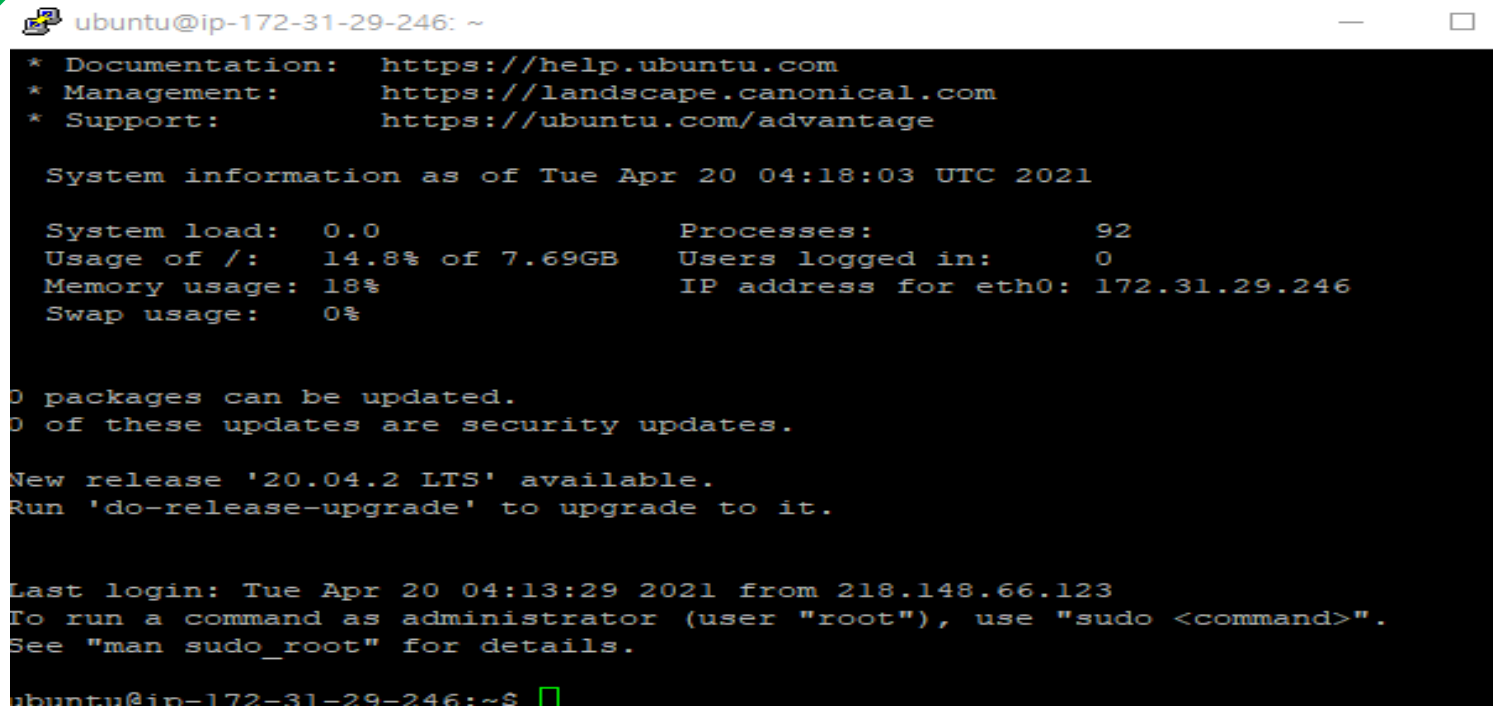
5. AWS – putty



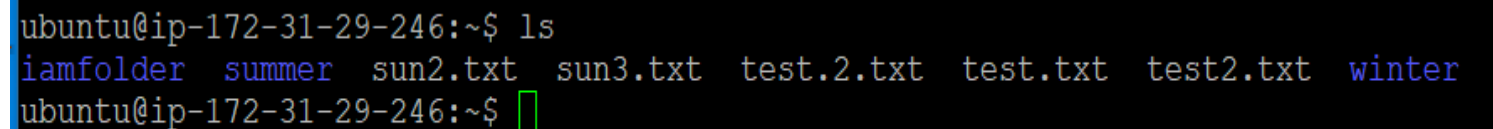
ppk키 설정하기



ppk키 연결 후 접속할 퍼블릭 DNS를
AWS에서 복사해서 사용한다.



Putty로 AWS ubuntu 연결 후 명령어를 활용해서 사용할 수 있다.



5. AWS – PowerShell

Windows PowerShell

```
PS C:\Users\User\Downloads>
```

ppk key가 있는 곡에서 PowerShell 실행

Windows PowerShell

```
PS C:\Users\User\Downloads> ssh -i "yad.pem" ubuntu@ec2-3-133-7-163.us-east-2.compute.amazonaws.com
```

사용할 AWS 퍼블릭 DNS 주소를 복사해서 사용

퍼블릭 DNS 연결 완료 후
AWS ubuntu 명령어로 사용가능

ubuntu@ip-172-31-29-246: ~

```
PS C:\Users\user\Downloads> ssh -i "yad.pem" ubuntu@ec2-3-133-7-163.us-east-2.compute.amazonaws.com
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1045-aws x86_64)
```

```
* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage
```

System information as of Tue Apr 20 07:36:25 UTC 2021

```
System load:  0.0      Processes:            97
Usage of /:   20.9% of 7.69GB   Users logged in:     1
Memory usage: 19%      IP address for eth0: 172.31.29.246
Swap usage:   0%
```

```
* Canonical Livepatch is available for installation.
- Reduce system reboots and improve kernel security. Activate at:
  https://ubuntu.com/livepatch
```

```
20 packages can be updated.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable
```

```
New release '20.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
```

```
Last login: Tue Apr 20 07:32:00 2021 from 218.148.66.123
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

```
ubuntu@ip-172-31-29-246:~$
```

```
ubuntu@ip-172-31-29-246:~$ ls
iamfolder  summer  sun2.txt  sun3.txt  test.2.txt  test.txt  test2.txt  winter
ubuntu@ip-172-31-29-246:~$
```

5. AWS – CMD

ubuntu@ip-172-31-29-246: ~

Microsoft Windows [Version 10.0.19042.928]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User>cd downloads

C:\Users\User\Downloads>ssh -i "yad.pem" ubuntu@ec2-3-133-7-163.us-east-2.compute.amazonaws.com

Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1045-aws x86_64)Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1045-aws x86_64)

* Documentation: <https://help.ubuntu.com>
* Management: <https://landscape.canonical.com>
* Support: <https://ubuntu.com/advantage>

System information as of Tue Apr 20 07:47:32 UTC 2021

System load:	0.0	Processes:	101
Usage of /:	20.9% of 7.69GB	Users logged in:	1
Memory usage:	19%	IP address for eth0:	172.31.29.246
Swap usage:	0%		

* Canonical Livepatch is available for installation.
- Reduce system reboots and improve kernel security. Activate at:
<https://ubuntu.com/livepatch>

20 packages can be updated.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

New release '20.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Apr 20 07:42:11 2021 from 218.148.66.123
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

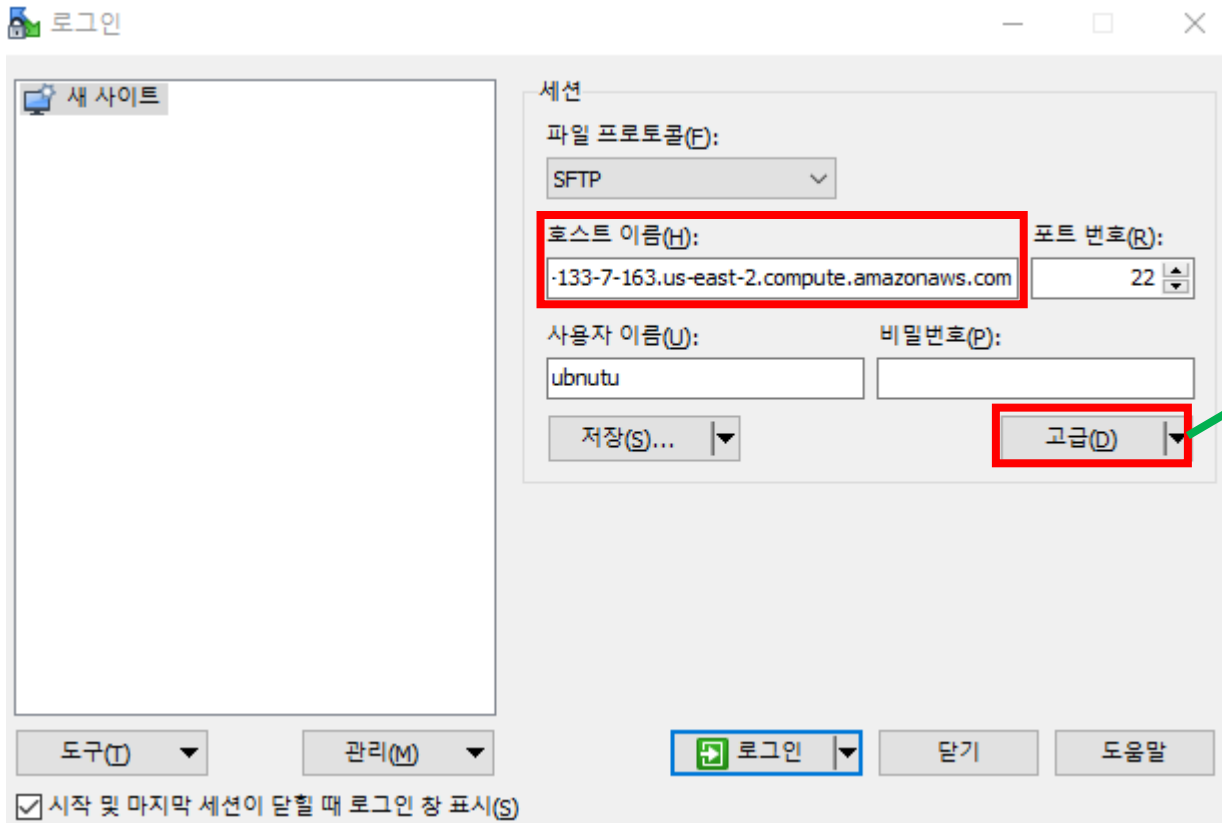
ubuntu@ip-172-31-29-246:~\$ ls
iamfolder summer sun2.txt sun3.txt test.2.txt test.txt test2.txt winter
ubuntu@ip-172-31-29-246:~\$

→ ppk key가 위치한 곳으로 이동

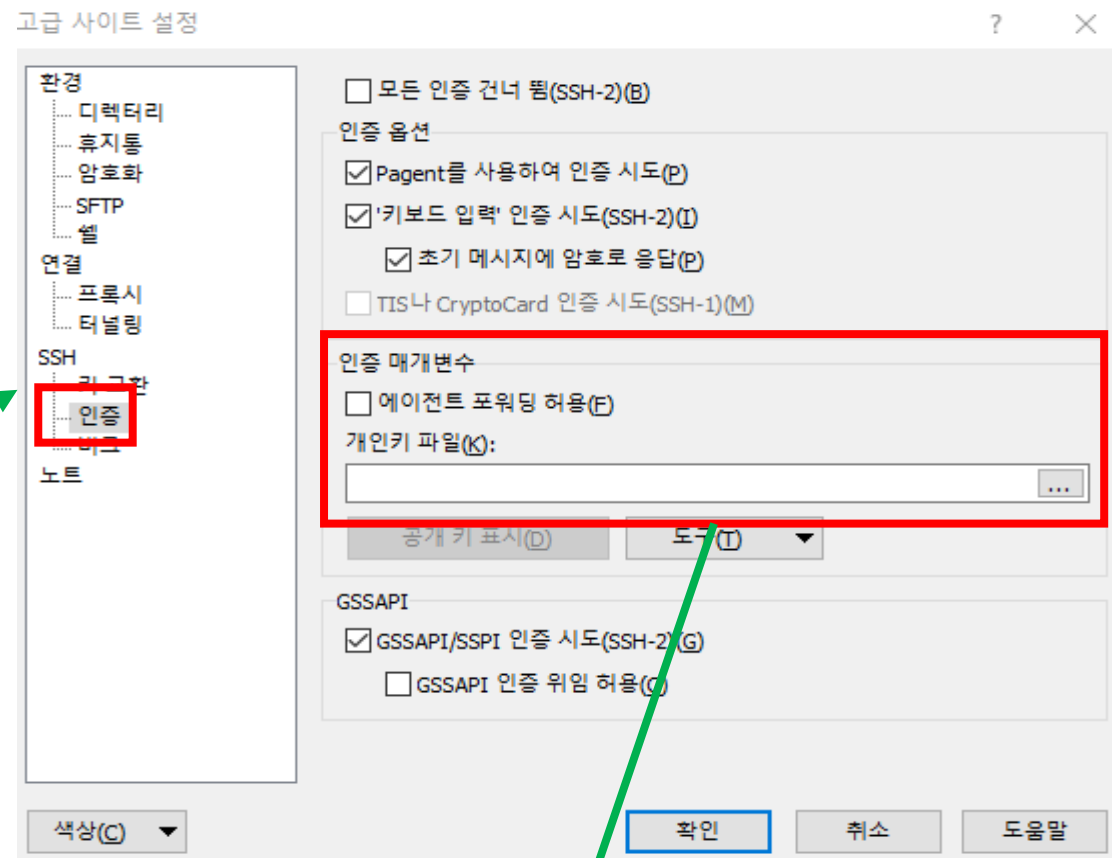
→ AWS 퍼블릭 DNS 주소 접속

→ 접속 완료 후 명령어 사용가능

5. AWS – Winscp

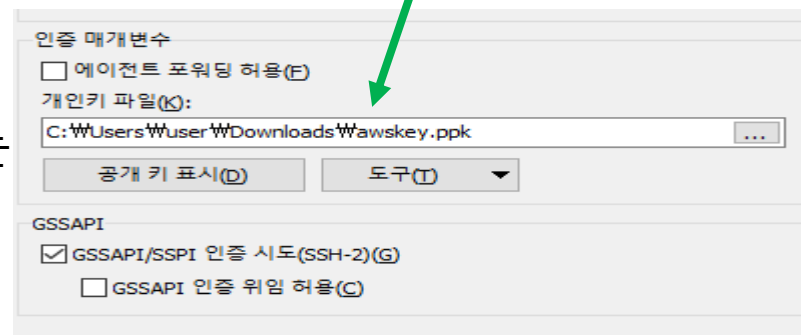


AWS 퍼블릭 DNS 사용



고급 옵션에서 SSH 인증에서 개인키 파일 설정하기

Winscp에서 개인키는
ppk파일을 사용한다.



5. AWS – Winscp

The image shows the WinSCP interface with a session configuration window open. The session name is '새 사이트' (New Site). The file protocol is set to 'SFTP'. The host is 'ubnutu@ec2-3-133-7-163.us-east-2.compute.amazonaws.com'. The port is '22'. The session is saved with the name 'ubnutu@ec2-3-133-7-163.us-east-2.compute.amazonaws.com'. The '확인' (Confirm) button is highlighted with a red box. A green arrow points from the '확인' button to the session list in the bottom left, where the session is now listed. Another green arrow points from the '확인' button to the main file transfer area, which shows the local file system 'C:\Users\User\Documents\#' on the left and the remote file system '/home/ubuntu/' on the right. The remote file system contains various files and folders, including 'iamfolder', 'summer', 'winter', 'sun2.txt', 'sun3.txt', 'test.2.txt', 'test.txt', 'test2.txt', and '데이터프레임연습.ipyn...'. The status bar at the bottom shows '4 숨김 0 B / 35.1 KB (0 / 9)' and '9 숨김'.

나의 컴퓨터와 원격으로 연결된 AWS ubuntu 간
파일 이동도 가능하게 연결 가능(Winscp는 GUI기반)

Window(RDP) 연결 과정

aws

서비스 ▼

Q

서비스, 기능, 마켓플레이스 제품, 설명서 검색

[Alt+S]

🔍

🔔

ya.d ▼

오하이오 ▼

지원

🔵

New EC2 Experience

Tell us what you think

✕

EC2 대시보드 New

이벤트

태그

제한

▼ 인스턴스

인스턴스 New

인스턴스 유형

시작 템플릿

스팟 요청

Savings Plans

예약 인스턴스 New

전용 호스트

용량 예약

▼ 이미지

AMI

▼ Elastic Block Store

볼륨

스냅샷

수명 주기 관리자

▼ 네트워크 및 보안

리소스

🔄

⚙️

미국 동부 (오하이오) 리전에서 다음 Amazon EC2 리소스를 사용하고 있음:

실행 중인 인스턴스	0	로드 밸런서	0
배치 그룹	0	보안 그룹	6
볼륨	2	스냅샷	0
인스턴스(모든 상태)	3	전용 호스트	0
키 페어	1	탄력적 IP	0

📘

AWS Launch Wizard for SQL Server를 사용하여 AWS에서 Microsoft SQL Server Always On 가용성 그룹을 손쉽게 크기 조정, 구성 및 배포할 수 있습니다. 자세히 알아보기

✕


인스턴스 시작

시작하려면 콘솔의 가상 서버인 Amazon EC2 인스턴스를 시작하십시오.

인스턴스 시작 ▼

참고: 인스턴스는 미국 동부 (오하이오) 리전에서 시작됩니다.

AWS의 인스턴스 생성


Windows

Microsoft Windows Server 2016 Base with Containers - ami-
0b01bfeec8bf7f1f6

선택

프리 티어 사용 가능

Microsoft Windows 2016 Datacenter edition with Containers. [English]

64비트(x86)

루트 디바이스 유형: ebs 가상화 유형: hvm ENA 활성화: 예

사용할 프로그램 선택

시작 상태

✓ **지금 인스턴스를 시작 중입니다.**
다음 인스턴스 시작이 개시됨: [i-03b3242901cf36dec](#) [시작 로그 보기](#)

ⓘ **예상 요금 알림 받기**
[결제 알림 생성](#) AWS 결제 예상 요금이 사용자가 정의한 금액을 초과하는 경우(예를 들면 프리 티어를 초과하는 경우) 이메일 알림을 받습니다.

인스턴스에 연결하는 방법

인스턴스를 시작 중이며, 사용할 준비가 되어 **실행 중** 상태가 될 때까지 몇 분이 걸릴 수도 있습니다. 새 인스턴스에서는 사용 시간이 즉시 시작되어 인스턴스를 중지 또는 종료할 때까지 계속 누적됩니다.

인스턴스 보기를 클릭하여 인스턴스의 상태를 모니터링합니다. 인스턴스가 **실행 중** 상태가 되고 나면 [인스턴스] 화면에서 인스턴스에 연결할 수 있습니다. 인스턴스에 연결하는 방법 [알아보기](#).

▼ 다음은 시작에 도움이 되는 유용한 리소스입니다.

- Windows 인스턴스에 연결하는 방법
- AWS 프리 티어에 대해 알아보기
- Amazon EC2: 사용 설명서
- Amazon EC2: Microsoft Windows 설명서
- Amazon EC2: 토론 포럼

인스턴스가 시작되는 동안 다음을 수행할 수도 있습니다.

- [상태 검사 경보 생성](#) 해당 인스턴스가 상태 검사를 통과하지 못하는 경우 알림을 받습니다. (추가 요금이 적용될 수 있음)
- [추가 EBS 볼륨 생성 및 연결](#) (추가 요금이 적용될 수 있음)
- [보안 그룹 관리](#)

[인스턴스 보기](#)

시작하면 인스턴스 시작

인스턴스 (1/1) 정보

↺

연결

인스턴스 상태 ▼

작업 ▼

인스턴스 시작 ▼

인스턴스 필터링

< 1 > ⚙

search: i-03b3242901cf36dec ✕

필터 지우기

<input checked="" type="checkbox"/>	Name ▼	인스턴스 ID ▲	인스턴스 상태 ▼	인스턴스 유형 ▼	상태 검사
<input checked="" type="checkbox"/>	win2016se...	i-03b3242901cf36dec	<div>✔ 실행 중</div>	t2.micro	-

인스턴스 상태 확인

인스턴스에 연결 정보

다음 옵션 중 하나를 사용하여 인스턴스 i-03b3242901cf36dec에 연결

Session Manager

RDP 클라이언트

EC2 직렬 콘솔

선택한 원격 데스크톱 클라이언트를 사용하고 아래의 RDP 바로 가기 파일을 다운로드하여 실행하면 Windows 인스턴스에 연결할 수 있습니다.

원격 데스크톱 파일 다운로드

메시지가 표시되면 다음 세부 정보를 사용하여 인스턴스에 연결합니다.

Public DNS

ec2-18-188-196-108.us-east-2.compute.amazonaws.com

사용자 이름

Administrator

암호

암호 가져오기

인스턴스 연결을 위해
RDP 클라이언트 설정
Public DNS, 이름, 암호
3가지 필요

암호 가져오기 실행

인스턴스를 디렉터리에 조인한 경우 디렉터리 자격 증명을 사용하여 인스턴스에 연결할 수 있습니다.

✓ 어제 (1)

yad.pem

2021-04-20 오전 11:50

PEM 파일

2KB

✓ 올해 초 (1)

Windows 암호 가져오기 [정보](#)

이 인스턴스에 대한 초기 Windows 관리자 암호를 검색하고 해독합니다.

암호를 해독하려면 이 인스턴스에 대한 키 페어가 필요합니다.

i

이 인스턴스와 연결된 키 페어

yad

키 페어로 이동:

🔗

Browse

✓ yad.pem

1.704KB

또는 아래 키 페어 내용을 복사하여 붙여 넣습니다.

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAy7bwmMS/7FTpRjQ1/W+QmMNUa+orH8mJb/hKLaaN8hqXz10a
6q8QmMVkeHDIBilbDulr9hz/OA+Fjk1xo4S9CmSERPpwGSyZz4j05i3VMIHsMegQ
3TZ1XBFMwqvn4Jsm9cqvaK0zxXNbLaWQjyR/3E+igZaoMOO0TN3ujKFHYTM9W1ov
dXb12VzWqOIJ8iGCREqBixb+exgTQISNKtp8reBnq/KMcCXCL9/G9xu9tocbC2dd
lkBzC4/hFz9H4KQKzc6/gvSM2GUXReG8aG33zn9dNtsKEZIRU1taMjezM4bcecvf
GUoHz8/yFzxz+/ymta0NVweFFKfCmNz5MT+jFQIDAQABAolBAQCJs6N24KgLDGqS
9f/4zEOltgN+3s8/sW+hrGgX65nPqYlsvXbQZH4HhoafkzziSHOZ3/BvcLJkJrnU

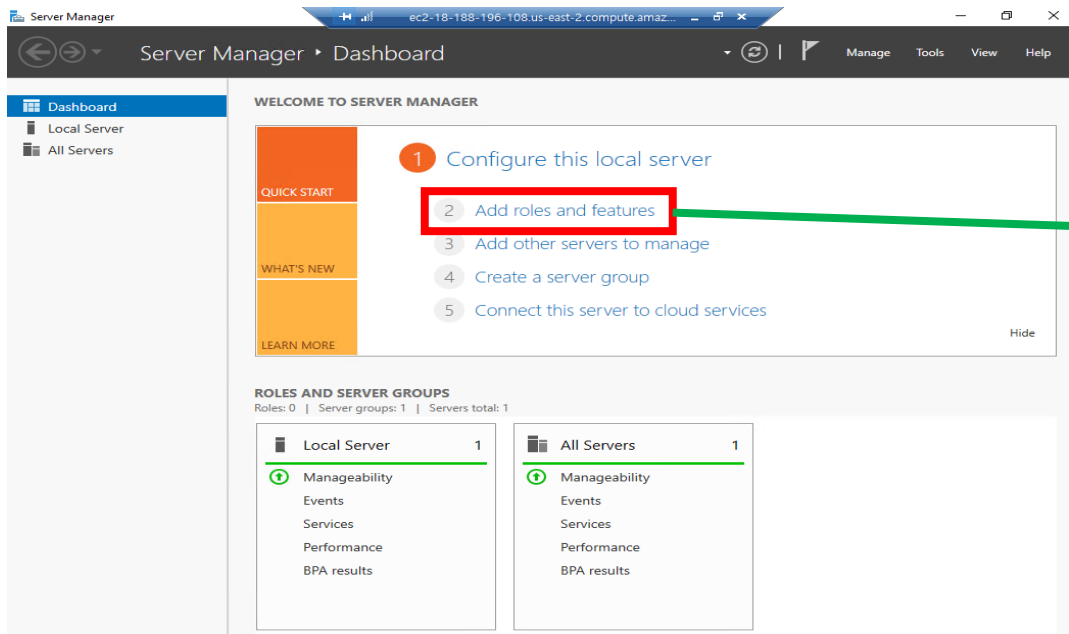
개인 pem 파일로 암호 해독

암호 해독 처리를 통해 개인 암호 확인

개인 암호 확인 및 상단 데스크톱 파일 다운로드

암호 복사 후 입력하기.

A screenshot of a Windows Security window titled "Windows 보안". The main heading is "사용자 자격 증명 입력" (Enter user credentials). Below it, a message states: "이 자격 증명은 ec2-18-188-196-108.us-east-2.compute.amazonaws.com에 연결할 때 사용됩니다." (This credential is used when connecting to ec2-18-188-196-108.us-east-2.compute.amazonaws.com). There is a label "Administrator" above a password input field containing black dots and an eye icon. Below the input field, the text "DESKTOP-J7OJM29W\Administrator" is displayed. At the bottom left, there is an unchecked checkbox labeled "기억" (Remember) and a link "다른 옵션 선택" (Select other options). Two large grey buttons at the very bottom are labeled "확인" (OK) and "취소" (Cancel).



연결된 RDP에서 server 설정 2번에 add 실행

Select one or more roles to install on the selected server.

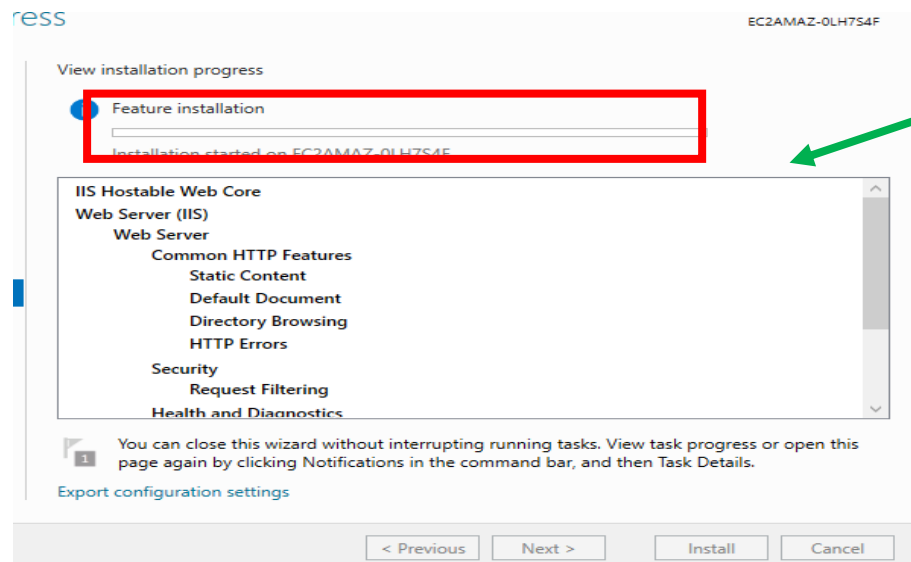
Roles

- ☐ Active Directory Certificate Services
- ☐ Active Directory Domain Services
- ☐ Active Directory Federation Services
- ☐ Active Directory Lightweight Directory Services
- ☐ Active Directory Rights Management Services
- ☐ Device Health Attestation
- ☐ DHCP Server
- ☐ DNS Server
- ☐ Fax Server
- ☒ File and Storage Services (1 of 12 installed)
- ☐ Host Guardian Service
- ☐ Hyper-V
- ☐ MultiPoint Services
- ☐ Network Controller
- ☐ Network Policy and Access Services
- ☐ Print and Document Services
- ☐ Remote Access
- ☐ Remote Desktop Services
- ☐ Volume Activation Services
- ☐ Web Server (IIS)

Select one or more features to install on the selected server.

Features

- ☐ .NET Framework 3.5 Features
- ☒ .NET Framework 4.6 Features (2 of 7 installed)
- ☐ Background Intelligent Transfer Service (BITS)
- ☐ BitLocker Drive Encryption
- ☐ BitLocker Network Unlock
- ☐ BranchCache
- ☐ Client for NFS
- ☒ Containers (Installed)
- ☐ Data Center Bridging
- ☐ Direct Play
- ☐ Enhanced Storage
- ☐ Failover Clustering
- ☐ Group Policy Management
- ☐ Host Guardian Hyper-V Support
- ☐ I/O Quality of Service
- ☒ IIS Hostable Web Core
- ☐ Internet Printing Client
- ☐ IP Address Management (IPAM) Server
- ☐ iSNS Server service



Install 진행

ROLES AND SERVER GROUPS

Roles: 2 | Server groups: 1 | Servers total: 1

File and Storage Services 1

- Manageability
 - Events
 - Performance
 - BPA results

IIS 1

- Manageability
 - Events
 - Services
 - Performance
 - BPA results

Local Server 1

- Manageability
 - Events
 - Services
 - Performance
 - BPA results

4/21/2021 3:58 AM

All Servers 1

- Manageability
 - Events
 - Services
 - Performance
 - BPA results

4/21/2021 3:58 AM

IIS 설치 확인

Name	보안 그룹 ID	보안 그룹 이름
<input checked="" type="checkbox"/> win2016server2	sg-0308579d75945c40b	launch-wizard-6

sg-0308579d75945c40b - launch-wizard-6

세부 정보 | **인바운드 규칙** | 아웃바운드 규칙 | 태그

인바운드 규칙 (1) 인바운드 규칙 편집

유형	프로토콜	포트 범위	소스	설명 - 선택 사항
RDP	TCP	3389	0.0.0.0/0	-

인바운드 규칙 2 삭제

유형 정보: 모든 TCP
소스 유형 정보: 위치 무관

프로토콜 정보: TCP
소스 정보: 0.0.0.0/0, ::/0

포트 범위 정보: 0 - 65535
설명 - 선택 사항 정보:

규칙 추가

참고: 기존 규칙을 편집하면 편집된 규칙이 삭제되고 새 세부 정보로 새 규칙이 생성됩니다. 이렇게 하면 새 규칙이 생성될 때까지 해당 규칙에 의존하는 트래픽이 잠시 중단될 수 있습니다.

취소 변경 사항 미리 보기 규칙 저장

sg-00dd2fa9c9d739a5e - launch-wizard-7

세부 정보 | **인바운드 규칙** | 아웃바운드 규칙 | 태그

인바운드 규칙 (3)

인바운드 규칙 편집

유형	프로토콜	포트 범위	소스	설명 - 선택 사항
모든 TCP	TCP	0 - 65535	0.0.0.0/0	-
모든 TCP	TCP	0 - 65535	::/0	-
RDP	TCP	3389	0.0.0.0/0	-

AWS 인바운드 규칙 설정

AWS 인바운드 규칙 확인

인스턴스: i-03fdbdf5b9812a62c(win2016server)

세부 정보 | 보안 | 네트워킹 | 스토리지 | 상태 검사 | 모니터링 | 태그

▼ 인스턴스 요약 정보

인스턴스 ID i-03fdbdf5b9812a62c (win2016server)	퍼블릭 IPv4 주소 3.14.126.1 개방 주소법	프라이빗 IPv4 주소 172.31.24.179
인스턴스 상태 실행 중	퍼블릭 IPv4 DNS ec2-3-14-126-1.us-east-2.compute.amazonaws.com 개방 주소법	프라이빗 IPv4 DNS ip-172-31-24-179.us-east-2.compute.internal
인스턴스 유형 t2.micro	탄력적 IP 주소 -	VPC ID vpc-a65dc8cd
AWS Compute Optimizer 찾기 권장 사항을 위해 AWS Compute Optimizer에 옵트인합니다. 자세히 알아보기	IAM 역할 -	서브넷 ID subnet-52749f2f

▼ 인스턴스 세부 정보 정보

접속 가능한 퍼블릭 DNS 확인

File Explorer: This PC > Local Disk (C:) > inetpub > wwwroot

Name	Date modified	Type	Size
hobby.htm	4/21/2021 4:38 AM	HTML Document	1 KB
iisstart.htm	4/21/2021 4:35 AM	HTML Document	1 KB
iisstart.png	4/21/2021 4:28 AM	PNG File	98 KB
me.htm	4/21/2021 4:36 AM	HTML Document	1 KB

RDP에 있는 페이지 확인

Browser: ec2-3-14-126-1.us-east-2.compute.amazonaws.com

동형's 홈페이지

환영합니다.
[동형's info](#)

Browser: ec2-3-14-126-1.us-east-2.compute.amazonaws.com/me.htm

동형's 정보

이름: 이동형
나이: 32
취미: 아래 링크로!! [동형's hobby](#)

Browser: ec2-3-14-126-1.us-east-2.compute.amazonaws.com/hobby.htm

동형's 취미

취미1: 여행
취미2: 쇼핑
취미3: 영화감상
취미4: 걸기
취미5: 누워있기
[메인으로](#)

각 페이지 연동 확인

GITHUB 연결과정(gitbash)

5. GITHUB-gitbash

```
user@DESKTOP-J7OJM29 MINGW64 ~  
$ pwd  
/c/Users/user  
  
user@DESKTOP-J7OJM29 MINGW64 ~  
$ cd Documents/  
  
user@DESKTOP-J7OJM29 MINGW64 ~/Documents  
$ cd data09  
  
user@DESKTOP-J7OJM29 MINGW64 ~/Documents/data09 (main)  
$ pwd  
/c/Users/user/Documents/data09  
  
user@DESKTOP-J7OJM29 MINGW64 ~/Documents/data09 (main)  
$ git inir  
git: 'inir' is not a git command. See 'git --help'.  
  
The most similar command is  
    init  
  
user@DESKTOP-J7OJM29 MINGW64 ~/Documents/data09 (main)  
$ git init  
Reinitialized existing Git repository in c:/Users/user/Documents/data09/.git/  
  
user@DESKTOP-J7OJM29 MINGW64 ~/Documents/data09 (main)  
$ git add .  
  
user@DESKTOP-J7OJM29 MINGW64 ~/Documents/data09 (main)  
$ git commit -m 'data commit'  
On branch main  
nothing to commit, working tree clean  
  
user@DESKTOP-J7OJM29 MINGW64 ~/Documents/data09 (main)  
$ cd ../  
  
user@DESKTOP-J7OJM29 MINGW64 ~/Documents  
$ pwd  
/c/Users/user/Documents  
  
user@DESKTOP-J7OJM29 MINGW64 ~/Documents  
$ git init  
Initialized empty Git repository in c:/Users/user/Documents/.git/
```

```
user@DESKTOP-J7OJM29 MINGW64 ~/Documents (master)  
$ git add .  
warning: could not open directory 'My Music/': Permission denied  
warning: could not open directory 'My Pictures/': Permission denied  
warning: could not open directory 'My Videos/': Permission denied  
warning: LF will be replaced by CRLF in data/seaborn그 래 프 그 리 기 .html.  
The file will have its original line endings in your working directory  
warning: LF will be replaced by CRLF in data/seaborn그 래 프 그 리 기 .ipynb.  
The file will have its original line endings in your working directory  
warning: LF will be replaced by CRLF in data/데 이 터 프 레 일 합 치 기 3 (1).ipynb.  
The file will have its original line endings in your working directory  
warning: LF will be replaced by CRLF in data/데 이 터 프 레 일 합 치 기 3 (2).ipynb.  
The file will have its original line endings in your working directory  
warning: LF will be replaced by CRLF in data/데 이 터 프 레 일 합 치 기 3.html.  
The file will have its original line endings in your working directory  
warning: LF will be replaced by CRLF in data/데 이 터 프 레 일 합 치 기 3.ipynb.  
The file will have its original line endings in your working directory  
warning: adding embedded git repository: data09  
hint: You've added another git repository inside your current repository.  
hint: Clones of the outer repository will not contain the contents of  
hint: the embedded repository and will not know how to obtain it.  
hint: If you meant to add a submodule, use:  
hint:  
hint:   git submodule add <url> data09  
hint:  
hint: If you added this path by mistake, you can remove it from the  
hint: index with:  
hint:  
hint:   git rm --cached data09  
hint:  
hint: See "git help submodule" for more information.  
warning: LF will be replaced by CRLF in practice/.ipynb_checkpoints/practice-checkpoint.ipynb.  
The file will have its original line endings in your working directory  
warning: LF will be replaced by CRLF in practice/practice.ipynb.  
The file will have its original line endings in your working directory
```

Gitbash를 활용하여 GITHUB에 파일 올리기

5. GITHUB-gitbash

```
user@DESKTOP-J7OJM29 MINGW64 ~/Documents (master)
$ git commit -m 'data up'
[master (root-commit) d0c132f] data up
50 files changed, 44501 insertions(+)
create mode 100644 data/2010.csv
create mode 100644 data/2010.xlsx
create mode 100644 data/2011.csv
create mode 100644 data/2012.csv
create mode 100644 data/2012.xlsx
create mode 100644 data/9937603E5DD4AF0419.png
create mode 100644 data/9972874E5DD4B7201B.png
create mode 100644 data/99C5484E5DD4B72018.png
create mode 100644 data/99F5E24E5DD4B7201F.png
create mode 100644 data/___MACOSX/._2010.csv
create mode 100644 data/___MACOSX/._2010.xlsx
create mode 100644 data/___MACOSX/._2011.csv
create mode 100644 data/___MACOSX/._2012.csv
create mode 100644 data/___MACOSX/._2012.xlsx
create mode 100644 data/___MACOSX/._9937603E5DD4AF0419.png
create mode 100644 data/___MACOSX/._9972874E5DD4B7201B.png
create mode 100644 data/___MACOSX/._99C5484E5DD4B72018.png
create mode 100644 data/___MACOSX/._99F5E24E5DD4B7201F.png
create mode 100644 data/___MACOSX/._concat_1.csv
create mode 100644 data/___MACOSX/._concat_2.csv
```

```
user@DESKTOP-J7OJM29 MINGW64 ~/Documents (master)
$ git branch -M main

user@DESKTOP-J7OJM29 MINGW64 ~/Documents (main)
$ git remote add origin https://github.com/leedh90/datacollect.git
```

Gitbash를 활용하여 GITHUB에 파일 올리기
주의사항 : 로그인 아이디 및 비밀번호 체크

```
user@DESKTOP-J7OJM29 MINGW64 ~/Documents (main)
$ git push -u origin main
Logon failed, use ctrl+c to cancel basic credential prompt.
error: unable to read askpass response from 'c:/Program Files/Git/mingw64/libexec/git-core/git-gui--askpass'
Username for 'https://github.com':
error: unable to read askpass response from 'c:/Program Files/Git/mingw64/libexec/git-core/git-gui--askpass'
Password for 'https://github.com':
remote: No anonymous write access.
fatal: Authentication failed for 'https://github.com/leedh90/datacollect.git/'

user@DESKTOP-J7OJM29 MINGW64 ~/Documents (main)
$ git push -u origin main
Logon failed, use ctrl+c to cancel basic credential prompt.
Enumerating objects: 54, done.
Counting objects: 100% (54/54), done.
Delta compression using up to 4 threads
Compressing objects: 100% (53/53), done.
Writing objects: 100% (54/54), 637.90 KiB | 9.24 MiB/s, done.
Total 54 (delta 20), reused 0 (delta 0)
remote: Resolving deltas: 100% (20/20), done.
To https://github.com/leedh90/datacollect.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

leedh90 / datacollect

<> Code ⓘ Issues ⓘ Pull requests ⏮ Actions 📁 Projects 📖 Wiki ⓘ Security

main 1 branch 0 tags

Go to file Add file Code

user data up		d0c132f 32 minutes ago	1 commit
data	data up	32 minutes ago	
data09	data up	32 minutes ago	
oCam	data up	32 minutes ago	
practice	data up	32 minutes ago	
desktop.ini	data up	32 minutes ago	

Help people interested in this repository understand your project by adding a README. Add a README