



Spring 기반 중고차 웹사이트

VC  
VARCHAR

코알라

김수연, 김종현, 이준선, 이향준, 임환국, 황지민

## 프로젝트 개요

1

- 1. 기획 및 개요
- 2. 역할 분담
- 3. 개발 환경
- 4. 개발 일정
- 5. 버전 관리



## 설계

2

- 1. ERD
- 2. User Flow
- 3. Logic Process



## 코드 설명

3

- 1. 주요기능
- 2. 코드 간략 설명
- 3. 문제 해결



## 시연

4

- 1. VARCHAR 사이트 시연

## 프로젝트 개요

1

1. 기획 및 개요

2. 역할 분담

3. 개발 환경

4. 개발 일정

5. 버전 관리

6. 웹 페이지 구성



### • 기획 배경

- 최근 반도체 대란과 원가 상승으로 온라인 중고차 시장이 활발

### • 기획 목적

- 2Layered-architecture 를 이용한 Spring 이관작업

### • 기대 효과

- Mybatis를 사용한 동적SQL 활용 능력
- WAS를 이용한 실시간 채팅기능 구현 능력
- 다국어처리(국제화)로 웹 접근성 능력

P04

# 팀원소개 및 역할분담



김수연

**Main Part : Model**

**Sub Part : Controller**

**담당 기능 : Spring 이관작업**

로그인 API

WebSocket 실시간 채팅



임한욱

**Main Part : Model**

**Sub Part : Controller**

**담당 기능 : 관리자 페이지**

결제 API

WebSocket 실시간 채팅



이준선

**Main Part : View**

**Sub Part : Model**

**담당 기능 : Spring 이관작업**

로그인 API

WebSocket 실시간 채팅



김종현

**Main Part : View**

**Sub Part : Model**

**담당 기능 : Spring 이관작업**

로그인 API

WebSocket 실시간 채팅



이향준

**Main Part : Controller**

**Sub Part : View**

**담당 기능 : 관리자 페이지**

결제 API

WebSocket 실시간 채팅



황지민

**Main Part : Controller**

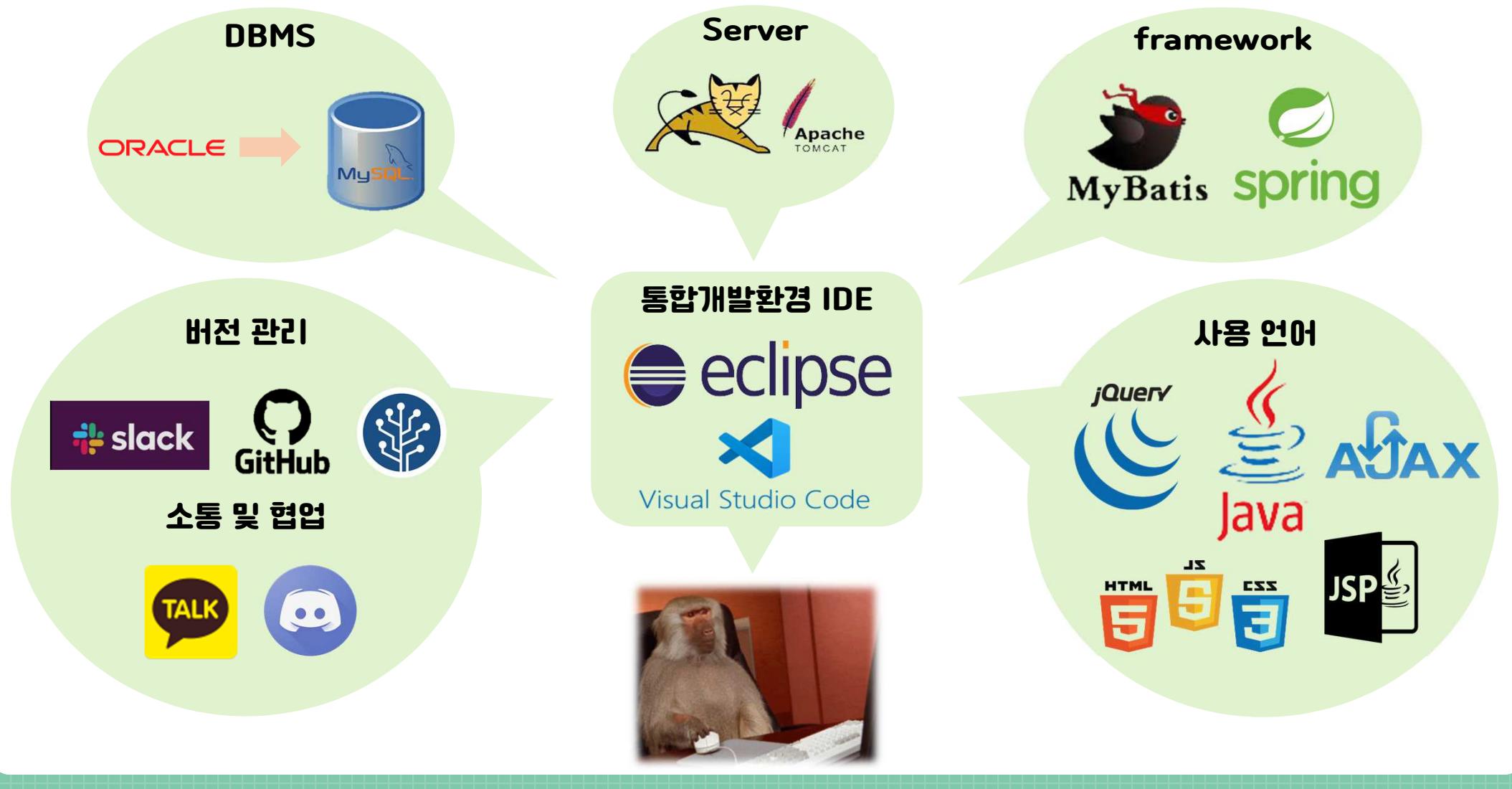
**Sub Part : View**

**담당 기능 : 관리자 페이지**

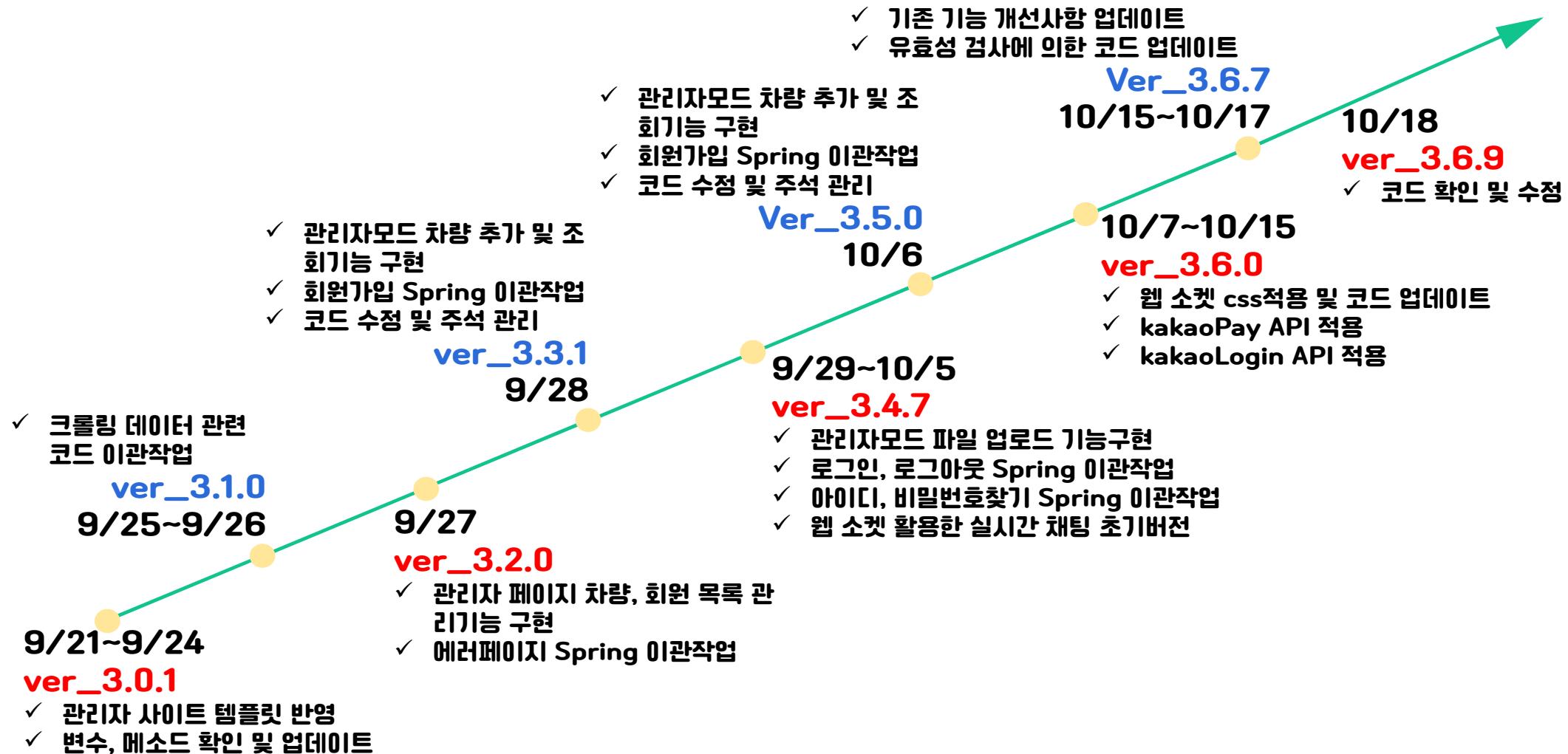
결제 API

WebSocket 실시간 채팅

# 개발 환경



기획일정	September										October																		
	21	22	23	24	25	26	27	28	29	30	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
주제 선정 및 템플릿 선정	■	■	■	■	■	■																							
DB구축 및 변수, 메서드 설정						■																							
JSP → Spring 이관작업						■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■								
로그인 API																■	■	■	■										
결제 API															■	■	■	■	■										
관리자 사이트						■	■	■	■					■	■	■	■	■	■										
실시간 채팅(채팅상담)														■	■	■	■	■	■	■	■	■							
유효성 검사 및 업데이트																							■	■					
PPT&발표준비																								■	■				



P08

V  
VARCHAR

설계

2

1. ERD

2. User Flow

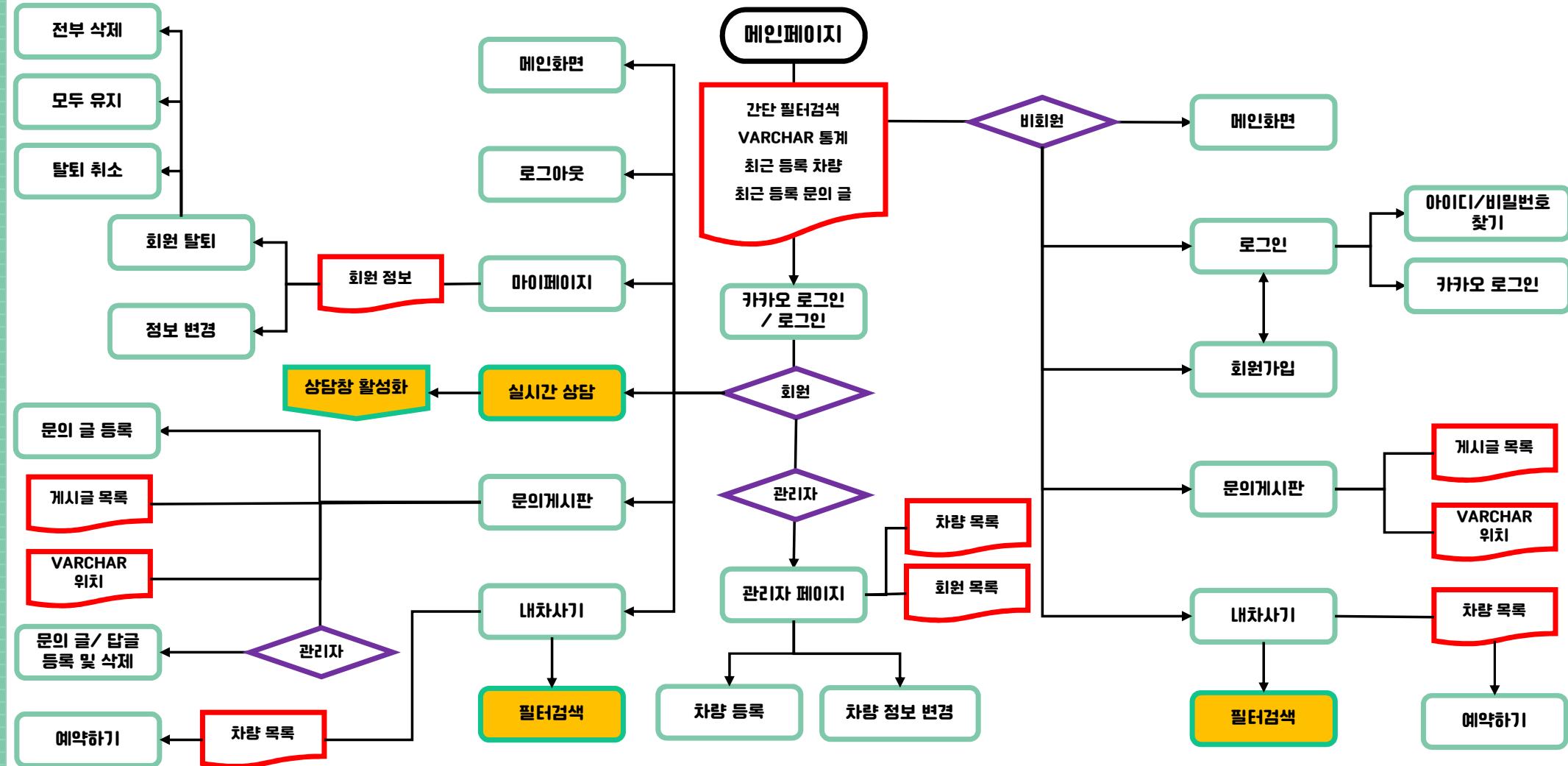
3. Logic Process

C MEMBER			
아이디	MID	VARCHAR(20)	PK
비밀번호	MPW	VARCHAR(20)	NOT NULL
사용자 이름	MNAME	VARCHAR(10)	NOT NULL
사용자 별명	MNICKNAME	VARCHAR(20)	NOT NULL
사용자 주소	MADD	VARCHAR(200)	NOT NULL
휴대폰 번호	MPHONE	VARCHAR(20)	NOT NULL
사용자 이메일	MEMAIL	VARCHAR(100)	NOT NULL
사용자 권한	MROLE	VARCHAR(20)	NOT NULL

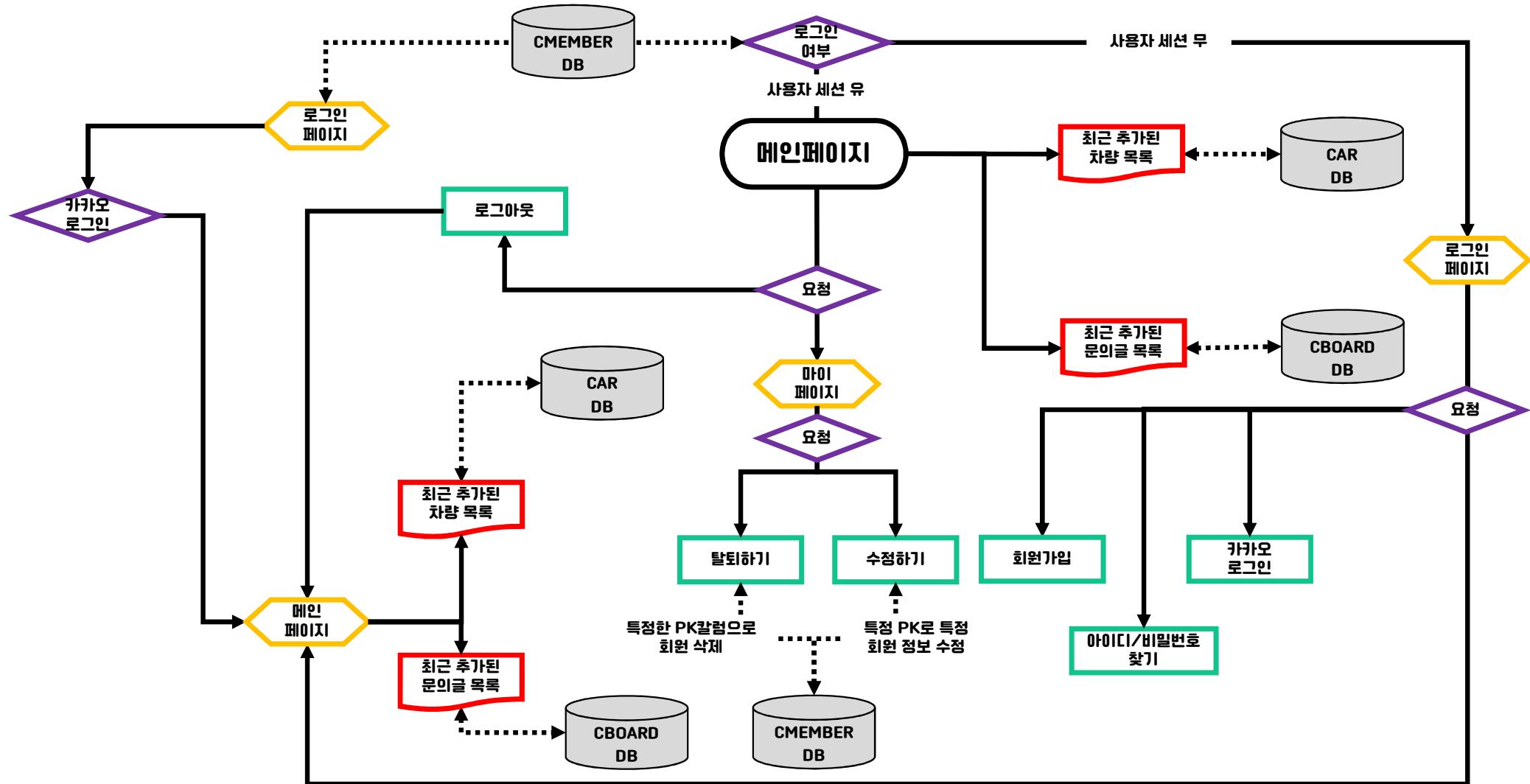
C BOARD			
글 고유번호	BNUM	INT	PK
사용자 아이디	MID	VARCHAR(20)	NOT NULL
사용자 별명	MNICKNAME	VARCHAR(20)	NOT NULL
글 제목	BTITLE	VARCHAR(50)	NOT NULL
글 내용	BCONTENT	VARCHAR(500)	NOT NULL
글 개수	BCNT	INT	DEFAULT 0
글 날짜	BDATE	VARCHAR(20)	NOT NULL

C REPLY			
답글 고유번호	RID	INT	PK
사용자 아이디	MID	VARCHAR(10)	NOT NULL
글 고유번호	BNUM	INT	NOT NULL
답글 내용	CMSG	VARCHAR(500)	NOT NULL
제약조건	CONSTRAINT CBOARD_CREPLY FOREIGN KEY(BNUM) REFERENCES CBOARD(BNUM) ON DELETE CASCADE		

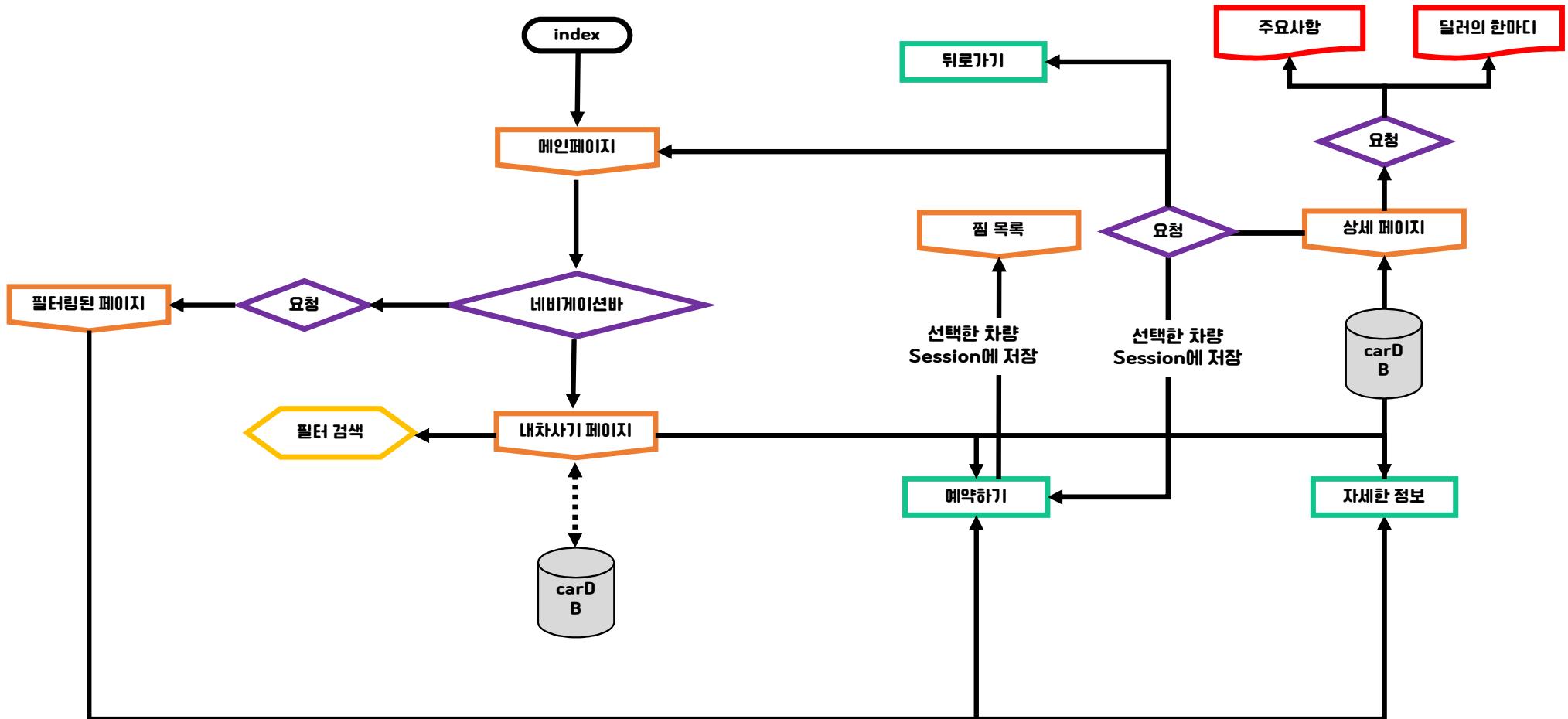
CAR			
차량고유번호	CNUM	INT	PK
차량 풀명	CTITLE	VARCHAR(300)	NOT NULL
차량 정보	CSUBTITLE	VARCHAR(300)	NOT NULL
차량 연식	CYEAR	INT	NOT NULL
차량 연료	CFUEL	CARCHAR(20)	NOT NULL
주행 거리	CKM	INT	NOT NULL
차량 가격	CPRICE	INT	NOT NULL
파는 지역	CCITY	VARCHAR(20)	NOT NULL
차량 사진	CIMG	VARCHAR(500)	NOT NULL



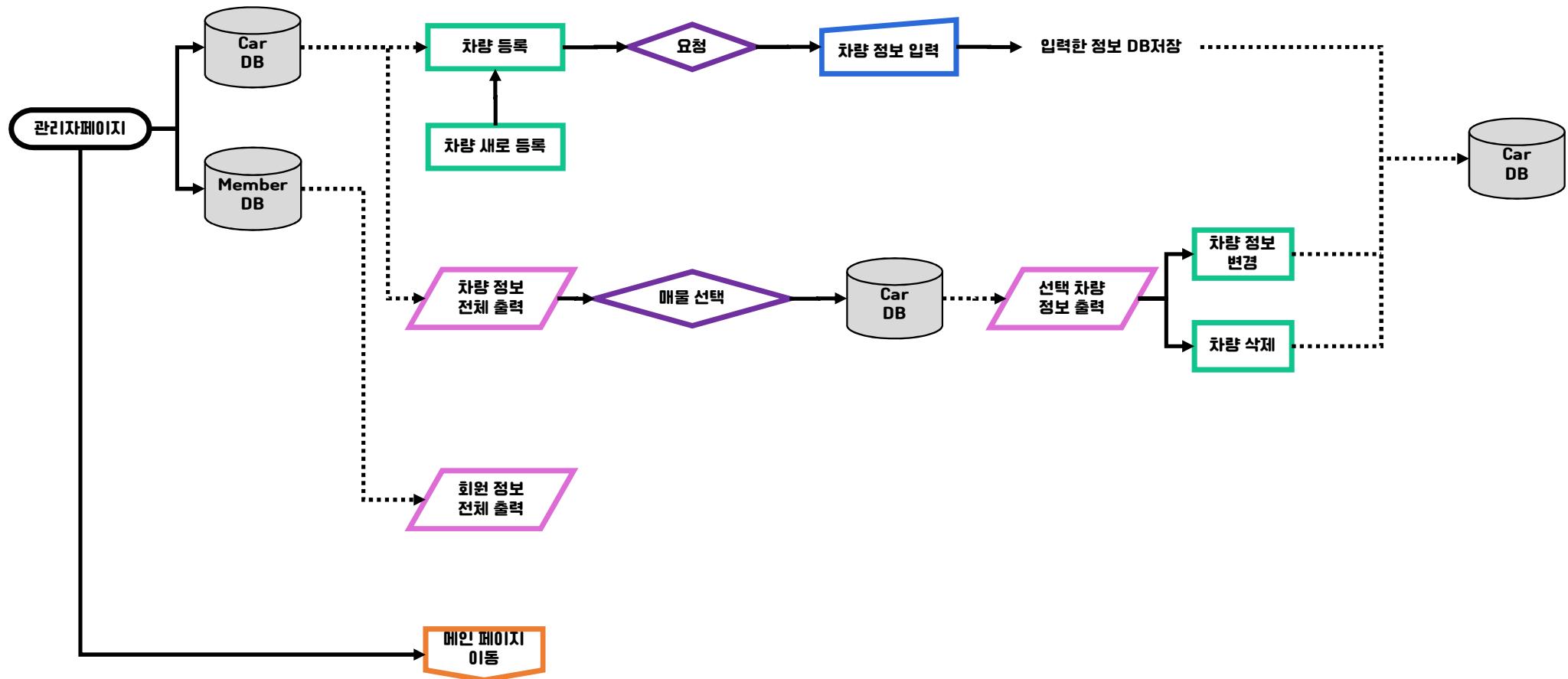
# Logic Process - 메인/회원



# Logic Process - 내 차 사기



# Logic Process - 관리자 페이지



P14

V  
CHAR

## 코드 설명

2

1. 주요 기능

2. 코드 간략 설명

3. 문제 해결

**Close Filter**

상품 정렬  
연식  
연료  
주행거리  
가격  
지역

상세조건 검색  
초기화

링컨 네비게이터 3.5 4WD L 블...  
87000 km 상담예약  
예약 하기 자세한 정보

포르쉐 911 터보 S 카브리올레  
20000 km 상담예약  
예약 하기 자세한 정보

벤츠 AMG GT S  
70000 km 상담예약  
예약 하기 자세한 정보

아우디 A4 2.0 TDI 콰트로 프레...  
180000 km 1200만원  
예약 하기 자세한 정보

롤스로이스 팬텀 6.7 V12  
90000 km 18500만원  
예약 하기 자세한 정보

랜드로버 뉴 레인지로버 5.0 V...  
90000 km 9900만원  
예약 하기 자세한 정보

**차량 상품 목록 더보기**

총 12/70 개의 상품을 보고 있습니다.  
**View more**

현재 검색 조건 반영 필요

더 보여줄 상품이 없으면 비활성화

Cart

## 차량 상품 목록 더보기 : Oracle ver.

```
sql_selectAll = "SELECT * FROM (SELECT B.*, ROWNUM AS R FROM (SELECT A.* FROM (SELECT * FROM CAR WHERE 1=1 "
+ CfuelSql + " " + CcitySql + " " + CyearSql + " " + CkmSql + " " + CpriceSql + ") A "+Check+) B) WHERE R BETWEEN "+svo.getRange1()+" AND "+svo.getRange2();
```



## 차량 상품 목록 더보기 : MySQL ver.

```
sql_selectAll = "SELECT * FROM (SELECT * FROM CAR WHERE 1=1 "
+ CfuelSql + " " + CcitySql + " " + CyearSql + " " + CkmSql + " " + CpriceSql + ")A "+Check+' LIMIT '+svo.getRange1()+ ","+svo.getRange2();
```

값을 12로 고정

## 현재 검색 조건 &amp; 상품 범위 비동기 요청 : filterSearch.jsp

```
$.ajax({
    type : 'GET',
    url : '${pageContext.request.contextPath}/showMore.do',
    data : {
        cityList : JSON.stringify(cityList),
        fuelList : JSON.stringify(fuelList),
        checksort : sort,
        price_min : pmin,
        price_max : pmax,
        year_min : ymin,
        year_max : ymax,
        km_min : kmin,
        km_max : kmax,
        range1 : range1,
        /   range2 : range2,
    },
});
```

## Oracle → MySQL 변경

- ROWNUM into LIMIT**
- ROWNUM BETWEEN 1 AND 12 → LIMIT 0, 12**
- 차게시판 페이지에 더보기를 위해 전달하는 범위 값 변경**

## CarSearchAction.java (JSP 코드)

```

// 가격
//검색할 때 입력한 값이 null이 아닌경우 = 범위를 위해 값을 입력하고 검색한 경우
// => 첫 차량검색페이지를 실행한 경우 파라미터 값이 null임
if(request.getParameter("pmin") != null && request.getParameter("pmax") != null) {
    int price_min = Integer.parseInt(request.getParameter("pmin")); //가격의 최소 값
    int price_max = Integer.parseInt(request.getParameter("pmax")); //가격의 최대 값
    svo.setPrice_min(price_min);
    svo.setPrice_max(price_max);
}
System.out.println("가격확인 : "+svo.getPrice_min()+"~"+svo.getPrice_max());

//연식
//검색할 때 입력한 값이 null이 아닌경우 = 범위를 위해 값을 입력하고 검색한 경우
if(request.getParameter("vmin") != null && request.getParameter("ymax") != null) {
    int year_min = Integer.parseInt(request.getParameter("vmin")); //연식의 최소 값
    int year_max = Integer.parseInt(request.getParameter("ymax")); //연식의 최대 값
    svo.setYear_min(year_min);
    svo.setYear_max(year_max);
}
System.out.println("연식확인 : "+svo.getYear_min()+"~"+svo.getYear_max());

//주행거리
if(request.getParameter("kmin") != null && request.getParameter("kmax") != null) {
    int km_min=Integer.parseInt(request.getParameter("kmin")); //주행거리 최소 값
    int km_max=Integer.parseInt(request.getParameter("kmax")); //주행거리 최대 값
    svo.setKm_min(km_min);
    svo.setKm_max(km_max);
}

String sort = request.getParameter("sort"); //select 정렬
request.setAttribute("sort", sort);
svo.setChecksort(sort);

```

## Command 객체의 자동 바인딩

- JSP에서는 Parameter값을 전달받아서 VO객체에 set
- Spring에서는 전달받은 Parameter가 VO Field에 존재하면 Command객체에 자동 바인딩

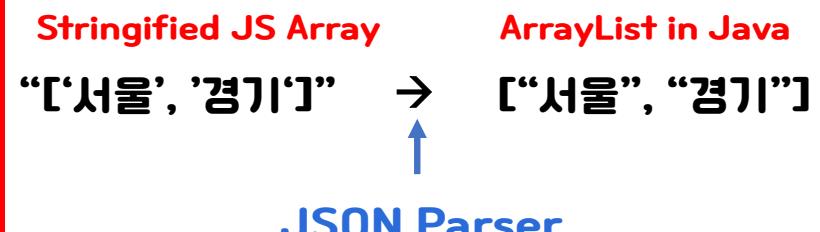
Spring에서는 불필요한 코드

```
CarSearchAction.java
$.ajax({
    type : 'GET',
    url : '${pageContext.request.contextPath}/showMore.do',
    data : {
        cityList : JSON.stringify(cityList),
        fuelList : JSON.stringify(fuelList),
        checksort : sort,
        price_min : pmin,
        price_max : pmax,
        year_min : ymin,
        year_max : ymax,
        km_min : kmin,
        km_max : kmax,
        range1 : range1,
        // range2 : range2,
    },
})
```

```
ShowMore.java
@RequestBody
@RequestMapping(value="/showMore.do")
public JSONObject showMore(@RequestParam(value="cityList") String cityOptions,
                           @RequestParam(value="fuelList") String fuelOptions,
                           SearchVO svo, Model model) {
    svo.setDataList(0);
    ArrayList<String> cityList = new ArrayList<String>();
    ArrayList<String> fuelList = new ArrayList<String>();
    try {
        JSONParser parser = new JSONParser();
        JSONArray cityTmp = (JSONArray)parser.parse(cityOptions);
        JSONArray fuelTmp = (JSONArray)parser.parse(fuelOptions);
        cityList = cityTmp;    fuelList = fuelTmp;
    } catch (ParseException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
    svo.setCityList(cityList);    svo.setFuelList(fuelList);
}
```

## JSONParser을 통한 배열 파싱

- 문자열화 JS배열을 비동기 방식으로 전달
- JSONParser는 전달받은 문자열 JSON데이터를 Java에서 사용가능한 형식으로 파싱
- 언어 차이에 따른 별도 데이터 가공 과정 생략으로 코드 간결화



## ShowMore.java

```

final int moreContent = 12; // 더보기를 클릭할 때마다 보여줄 상품 개수
svo.setRange2(moreContent);

List<CarVO> dataList = searchService.selectAll(svo);

// 다음에 보여줄 데이터 존재 여부 --> 더보기 버튼 활성화 / 비활성화
boolean showMore = true; // 더보기 버튼 활성화 여부
// 미리 다음에 보여줄 동일 항목 데이터 개수 계산

svo.setRange1(svo.getRange1() + moreContent);
svo.setRange2(moreContent);

//     svo.setRange2(range2);
List<CarVO> nextDataList = searchService.selectAll(svo);
// 더 보여줄 데이터가 없다면 --> 더보기 버튼 비활성화
if(nextDataList.size() == 0) {
    showMore = false;
}

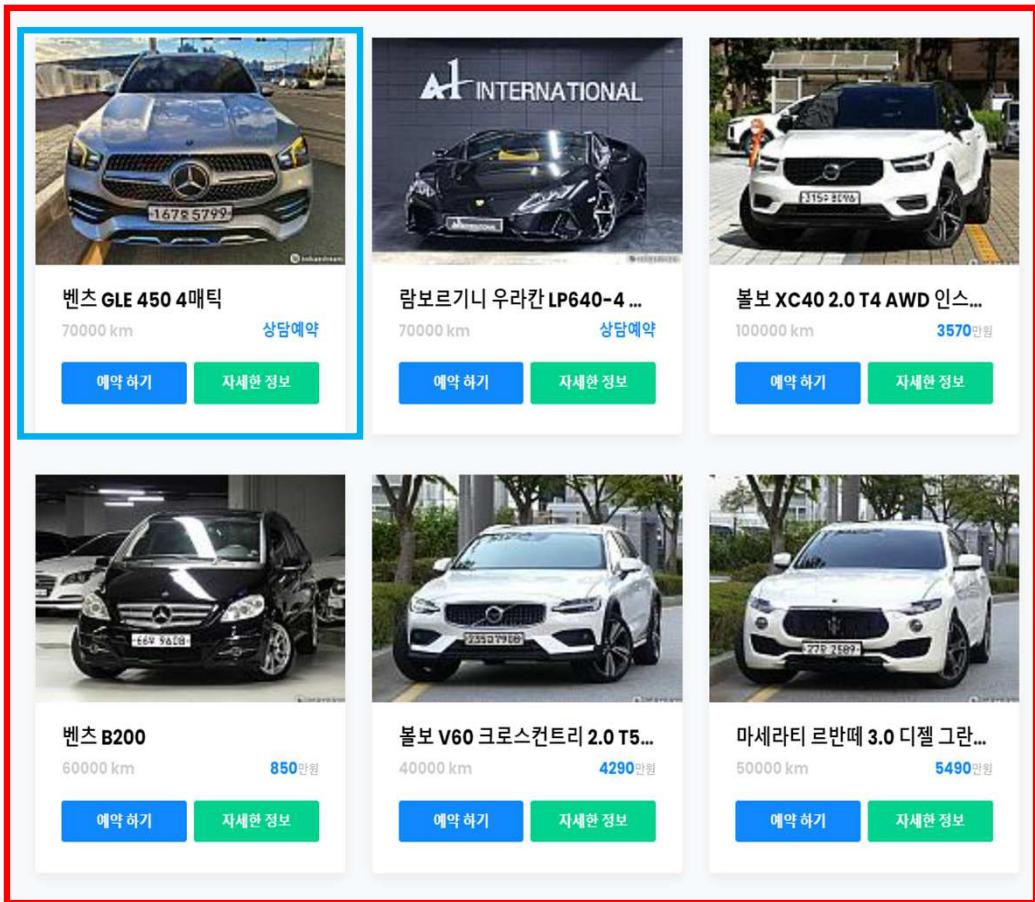
```

## 더 보여줄 차량 & 더보기 버튼 활성화 여부

- **유지보수를 위해 더 보여줄 차량 개수 상수화**
- **다음 범위 차량 개수 계산 → 더보기 버튼 활성화 여부 판단**

P20

# 주요기능 - 더 보기\_Spring 이관작업

VC  
VARCHAR

&lt;div id="viewItems"&gt;

## filterSearch.jsp

```

let NodeList = ""; // 추가할 HTML 전체 요소
$(result.dataList).each(function() { // 더 보여줄 차량 데이터에 대한 반복
    let newNode = "";
    newNode += "<div class='col-md-4'>";
    newNode += "<div class='car-wrap rounded'>";
    newNode += "<div class='img rounded d-flex align-items-end'>";
    newNode += "";
    newNode += "</div>";
    newNode += "<div class='text'>";
    newNode += "<h2 class='mb-0'>";
    newNode += "<a href='detail.do?cnum=" + this.cnum + "'>" + this.ctitle + "</a>";
    newNode += "</h2>";
    newNode += "<div class='d-flex mb-3'>";
    newNode += "<span class='cat'>" + this.ckm + " km</span>";

    // 상담예약 상품의 가격 저정은 400000만원 이상
    cprice = this.cprice >= 400000 ? "상담예약" : this.cprice + "<span>만원</span>";
    newNode += "<p class='price ml-auto'>" + cprice + "</p>";
    newNode += "</div>";
    newNode += "<p class='d-flex mb-0 d-block'>";
    newNode += "<a href='storeAdd.do?cnum=" + this.cnum + "' class='btn btn-primary py-2 mr-1'>예약 하기</a>";
    newNode += "<a href='detail.do?cnum=" + this.cnum + "' class='btn btn-secondary py-2 ml-1'>자세한 정보</a>";
    newNode += "</p>";
    newNode += "</div>";
    newNode += "</div>";
    newNode += "</div>";
    NodeList += newNode;
}

$(NodeList).appendTo($("#viewItems")); // 상품 내용 추가할 html 영역

if(!result.showMore) { // 상품 모두 출력 -> 더보기 버튼 XX
    document.querySelector(".moreContent").style.display = "none";
}
}

```

# 문제 해결 - 필터검색\_Spring 이관작업

## filterSearch.jsp

```

@Repository("searchDAO")
public class SearchDAO {
    @Autowired
    private JdbcTemplate jdbcTemplate;

    //메서드를 하나만 사용할 예정이므로 각 조건에 따라 들어갈 sql문의 초기값 설정
    String CfuelSql = "";
    String CcitySql = "";
    String CyearSql = "";
    String CkmSql = "";
    String CpriceSql = "";
    String sql_selectAll = "";
    String Check="";

    public List<CarVO> selectAll(SearchVO svo){ // 검색 결과 전체 조회 (전부 다 가져오기)
        //데이터는 CarVO로부터 받아올 예정이기 때문에 제네릭을 CarVO로 설정
        //필터에 사용될 데이터는 SearchVO에서 사용할 것이므로 인자로 두었음

        if(svo.getFuelList().size() > 0 && !svo.getFuelList().contains("전체")){
            //연료의 필터 값을 저장한 배열객체의 길이가 1 이상일 때
            StringBuilder cfuelSb = new StringBuilder(); // append 메서드를 사용하기 위해 StringBuilder을 통해 객체 생성
            ArrayList<String> fuelData = svo.getFuelList(); //fuelData 배열객체에 필터값을 저장한 값 삽입

            for(int i = 0; i < svo.getFuelList().size(); i++){
                cfuelSb.append("\'" + fuelData.get(i) + "\'"); //cfuelSB객체에 'index[i]번째의 필터배열값' 뒤에 붙여줌
                if(i+1 < svo.getFuelList().size()) //만약 필터 배열의 길이가 i+1보다 크다면
                    cfuelSb.append(","); // 중간에 ','를 붙여줌
            }
            //cfuelSql 객체 = AND CFUEL IN ('for문을 통해 나온 필터배열값')
            CfuelSql = "AND CFUEL IN ("+cfuelSb.toString()+"')";
        }
        else {
            CfuelSql = "";
        }
    }
}

```

**JSP에서는 DAO 필요시마다 객체 생성**

**Spring에서는 싱글톤 패턴 유지로 빈값을 별도로 전달**

## Spring Container의 Singleton Pattern

- 필터 검색 중 직전 검색내용을 초기화하고 재검색 해도 반영이 안되는 현상 발생
- JSP 프로젝트에서는 상단의 문자열 변수들을 공백으로 초기화
- Spring Container는 Singleton Pattern을 유지하므로 실행할 때마다 DAO를 객체화 하지 않음

# 주요기능 - 사용자Session\_Spring 이관작업

V  
CHAR

## MemberController.java

```

@Controller
@SessionAttributes({"userId","loginType","mrole"})
public class MemberController {

    @Autowired
    private MemberService memberService;
    @Autowired
    private SendMsgService sendmsgService;
    @Autowired
    private SendEmailService sendemailService;

    ////////////// Member ///////////
    //---로그인
    @RequestMapping(value="/login.do",method=RequestMethod.POST)
    public String selectOneMember(MemberVO mVO, HttpSession session, Model model) {
        mVO=memberService.selectOne(mVO);
        System.out.println("로그인 로그11 : " + mVO);

        if(mVO==null) {
            return "redirect:login.jsp";
        }
        else {
            model.addAttribute("userId", mVO.getMid()); //model에 "member"가 add!! -> session에 저장
            model.addAttribute("mrole", mVO.getMrole());

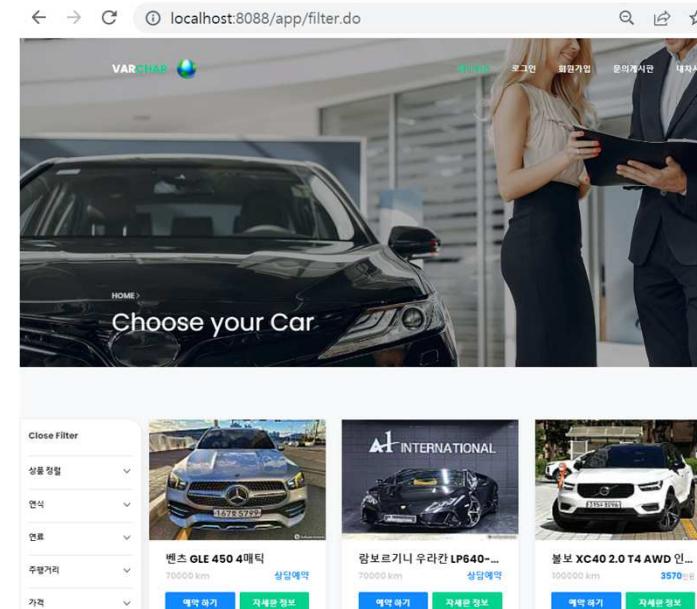
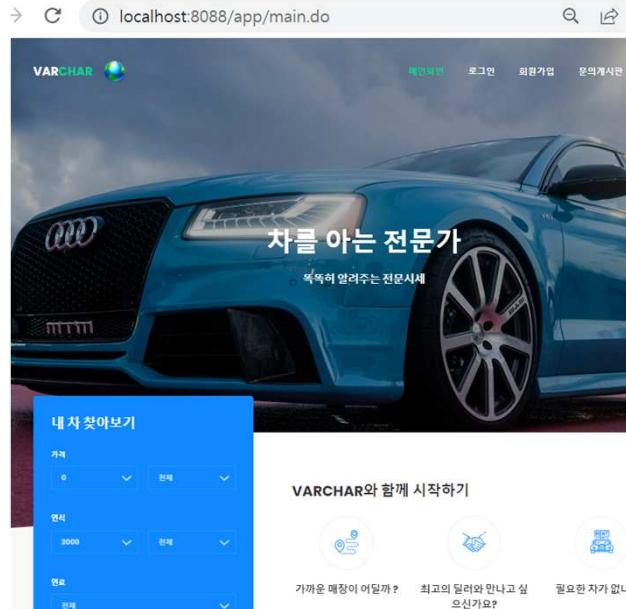
            return "redirect:main.do";
        }
    }
    //---로그아웃
    @RequestMapping("/logout.do")
    public String logout(SessionStatus sessionStatus) {
        sessionStatus.setComplete(); //저장된 model 객체를 없애줌
        return "redirect:login.do"; //VR 디플트 값이 forward
    }
}

```

## 로그인, 로그아웃 Session 이용

- 보안상의 이유로 Session에는 사용자의 모든 정보가 아닌 ID만 저장
- 관리자 페이지 및 댓글 관리를 위해 Session에 회원구분 정보(mrole) 저장
- Spring Container가 제공하는 Singleton Pattern
  - @SessionAttributes & Model 객체를 통한 Session 저장
  - Session을 모두 삭제하는 것이 아닌 저장된 Model 객체만을 정리하기 위해 SessionStatus 사용

# 주요기능 - 페이지 유지



```

import javax.servlet.http.HttpServletRequest;

public class PreserveURL {
    public static String preserveURL(HttpServletRequest request) {
        // Http Header의 referer 참조
        String prevURL = request.getHeader("referer");

        // 요청한 직전 웹 페이지로 복귀
        return "redirect:" + prevURL;
    }
}

```

## Http Headers의 refer 참조

- 직전 요청 정보 확인 가능

→ Refer 정보를 저장하여 적재적소 페이지 유지 가능

```

> document.referrer // Http header의 referrer
< 'http://localhost:8088/app/main.do'

```

# 주요기능 - 다국어 처리

- 상단 네비게이션 바에 다국어처리 기능 추가
- 페이지 유지 기능을 통해 모든 페이지에서 언어 선택 가능

다국어처리 이벤트 추가



중국어 ver.



일본어 ver.



## 오류문구

```
Stacktrace:] 을(를) 발생시켰습니다.
javax.servlet.jsp.JspTagException: No message found under code 'navigation.main' for locale 'ko_KR'.
    at org.springframework.web.servlet.tags.MessageTag.doEndTag(MessageTag.java:200)
    at org.apache.jsp.tag.web.nav_tag._jspx_meth_spring_005fmessage_005f0(nav_tag.java:202)
    at org.apache.jsp.tag.web.nav_tag.doTag(nav_tag.java:122)
    at org.apache.jsp.login_jsp._jspx_meth_koala_005fnav_005f0(login_jsp.java:360)
    at org.apache.jsp.login_jsp._jspService(login.jsp.java:163)
```

## Controller

```
//---- 언어 변경 시 기존 페이지 유지
@RequestMapping(value="/language.do", method=RequestMethod.GET)
public String ChangeLanguage(HttpServletRequest request, @RequestParam("lang") String lang) {
    System.out.println("언어 변경 : " + lang);
    System.out.println(PreserveURL.preserveURL(request));
    return PreserveURL.preserveURL(request);
```

직접 URL

## languageSelector.js

```
const countries = document.getElementsByClassName("language");
const languages = ['ko', 'jp', 'en', 'zh'];

for(let i = 0; i < countries.length; i++) {
    countries[i].onclick = () => {
        location.href = "language.do?lang=" + languages[i];
```

## 다국어처리 경로 오류

- 경로를 .jsp 파일로 지정 시 언어 감지 불가로 오류 발생
- 경로를 .do로 변경 후 controller를 거쳐서 이동
- 언어 변경 시 해당 페이지 유지를 위해서 직접 URL 경로를 저장하는 PreserveURL 클래스 모듈화

# 주요기능 - 필터검색\_mybatis 이관작업

V  
VARCHAR

## SearchDAO.java

```

if(svo.getFuelList().size() > 0 && !svo.getFuelList().contains("전체")) //연료
    System.out.println("DAO 로그 fuel 통과 중 fuel : " + svo.getFuelList());

StringBuilder cfuelSb = new StringBuilder(); // append 메서드를 사용하기 위해
ArrayList<String> fuelData = svo.getFuelList(); //fuelData 배열객체에 필터값

for(int i = 0; i < svo.getFuelList().size() ; i++){ //필터 배열의 길이만큼 반복
    cfuelSb.append("\'" + fuelData.get(i)+ "\'"); //cfuelSB객체에 'index[i]'번쨰 문자열을 cfuelSb에 추가
    if(i+1 < svo.getFuelList().size()) //만약 필터 배열의 길이가 i+1보다 크거나 같으면 ','
        cfuelSb.append(","); // 중간에 ','를 추가

}
System.out.println("DAO 로그 cfuelSb append : " + cfuelSb);
//cfuelSql 객체 = AND CFUEL IN ('for문을 통해 cfuelSb에 넣은 문자열')
CfuelSql = "AND CFUEL IN (" + cfuelSb.toString();
}

```

## filter-mapping.xml - Mybatis 이관

```

<if test= "fuelList.size != 0">
    AND CFUEL IN
        <foreach item= "cfuel" index= "index" collection= "fuelList"
            open= "(" separator= "," close= ")"
            #{"cfuel}
        </foreach>
</if>

```

# WHERE 1=1

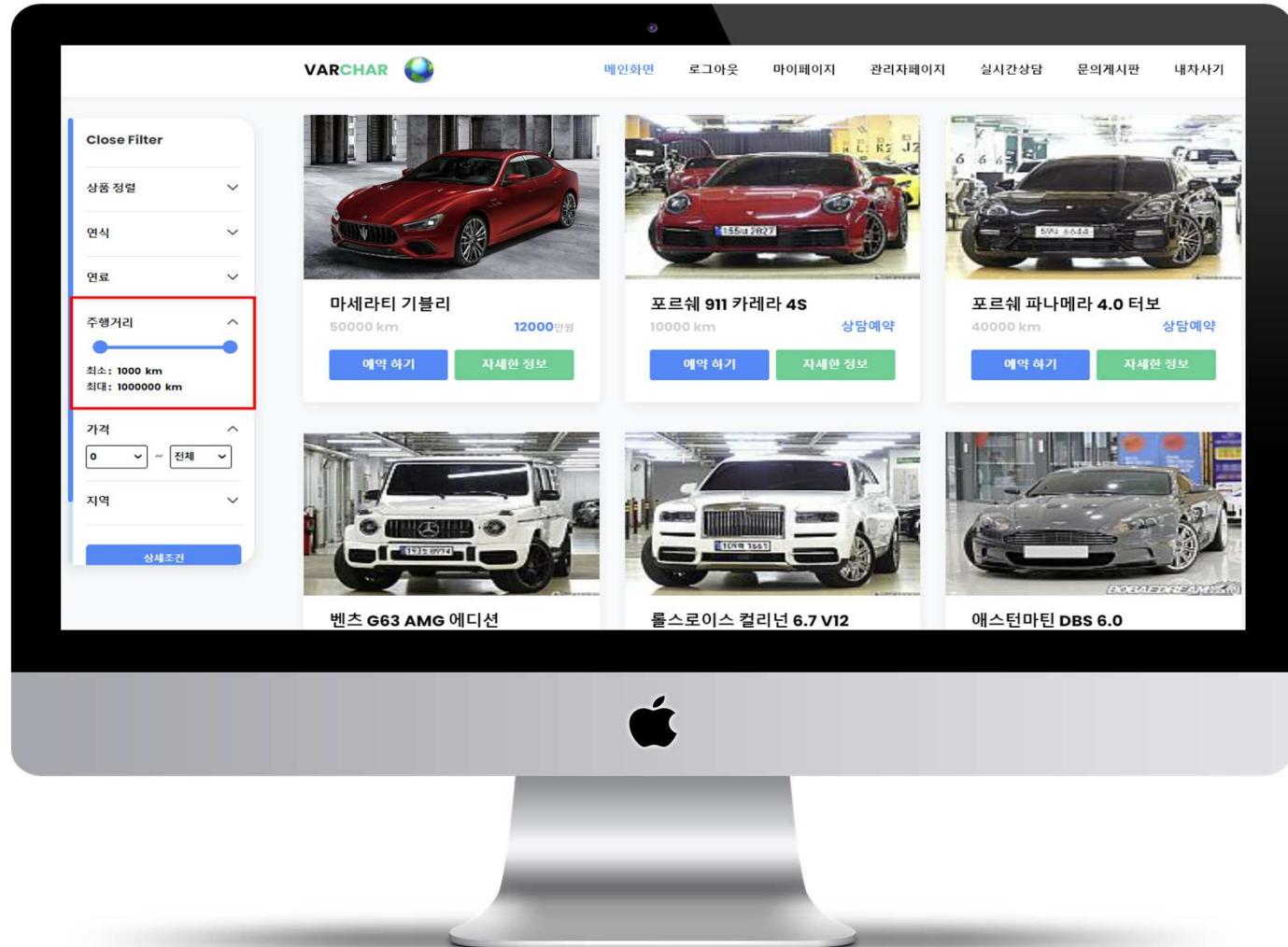
- 기존 자바로 구현했던 필터 기능을 mybatis로 이관
- Foreach의 속성을 활용해 배열 길이에 따라 SQL문 생성
- Separator 속성을 통해 구분자 생성

- Java 필터
- 다중 파라미터(배열)인 경우 append 함수를 사용하여 문자를 이어주며 index 만큼 반복
- 배열 길이를 비교하여 구분자 생성

# 주요기능 - 필터검색\_mybatis 이관작업

```
filter-mapping.xml
<if test="km_min >= 0">
    AND CKM BETWEEN #{km_min} AND #{km_max}
</if>
```

- 배열이 아닌 단일 파라미터 (주행거리, 가격, 연식)인 경우
  - 위 if 태그 내 조건식에 부합할 경우 쿼리문 추가 작성
  - 범위 데이터로 BETWEEN 쿼리문 사용



# 주요기능 - 관리자 페이지

**Manager.jsp**

```
<!-- 연료 option 처리 -->
<li>연료<select name="cfuel" class="dataTable-input">
    <option>가솔린</option>
    <option>디젤</option>
    <option>LPG</option>
    <option>전기</option>
</select></li>

<li>주행거리<input class="dataTable-input" type="number"
    min="0" step="1000" max="2147483600"
    name="ckm" required autocomplete="off">
</li>
```

- 정수 데이터인 경우 직접 입력
- 정수가 아닌 데이터를 기입했을 때의 **option** 태그 활용

**관리자페이지**



연료

- 가솔린
- 가솔린
- 디젤
- LPG
- 전기

지역

**관리자페이지**



주행거리

# 주요기능 - 관리자 페이지

```

ManagerController.java

@RequestMapping(value="insertCar.do", method=RequestMethod.POST)
public String insertCar(CarVO cvo,HttpServletRequest request,MultipartHttpServletRequest
String referer = (String)request.getHeader("Referer");
System.out.println("referer 값 : "+referer);
MultipartFile uploadFile = multipartRequest.getFile("file");
System.out.println("파일 인서트 : "+uploadFile);
try {

    if(!uploadFile.isEmpty()) {//업로드한 파일 존재여부 확인
        String fileName = uploadFile.getOriginalFilename(); //업로드한 파일명
        uploadFile.transferTo(new File(fileRoute+fileName)); //저장할 경로 결정
        cvo.setCimg("images/"+fileName);
        System.out.println("게시글 작성 파일이름 : "+fileName);
    }
    managerService.insertCar(cvo);
} catch (Exception e) {
    e.printStackTrace();
}
return "redirect:manager.do";
}

```

**관리자 페이지**

Varchar

새로 등록하기

차량명  
상세설명  
연식  
2000  
연도  
기술인  
주행거리  
가격  
지역  
서울  
이미지  
파일 선택 선택된 파일 없음

등록하기

**Car Information**

차량명	설명	연료	주행거리	가격	지역	연식
벤츠 S550L 4마티	리어 모니터/후진시 도어 헤드/8기통 디젤엔진 세단	가솔린	6000	2147483647	서울	2022
벤츠 S63 AMG 4마티+ 주제	정식/무사고/현대/발터리스/가능/부실 AS 경찰	가솔린	40000	2147483647	경기	2018
BMW X6 xDrive 50i	앞/뒤 후기형 라이트+M 버전 프론트 범퍼 개조	가솔린	240000	1580	경기	2011
람보르기니 아벤큐스도르 SSV LP770-4	노팅힐타운 무사고/무너리스/전체 PPF/가변배기	가솔린	9000	2147483647	경기	2019
람보르기니 우루스 4.0 V8	무사고/총0도/무사고/수퍼차/DNA 650마력 SUV	가솔린	4000	2147483647	경기	2021
기아 캐스티아 6.2 ESV 4WD	현금차량/기민소유/무사고/10만승/2열VIP석/시트加热	가솔린	80000	2147483647	경기	2015
벤츠 아리에고 GLS 600 4마티	무주행 신차/리스/개인리스/미디션전용 서비스도스텝	가솔린	31000	2147483647	서울	2022
포르쉐 911 GT3	정식/한정판무사고/포르쉐/아방타운/그란포르쉐	가솔린	170000	2147483647	서울	2022
벤츠 S550L 4마티	정식/무사고/Q 브스트/풀레인저/최신형 플래그십	가솔린	100000	2147483647	서울	2022
포르쉐 파나메라 4.0 GTS	정식/한정판무사고/차량/한국판수입/49마력 GTS	가솔린	8000	2147483647	서울	2022

Showing 1 to 10 of 71 entries

**User Information**

userId	userPw	userName	userNickname	useraddr	userphone	useremail
admin	qwer1234	김수연	sdflsf	서울시	010-111-112	rlatndus2005@naver.com
park1	aaaa1234	김종현	왕정면	서울시	010-111-112	rlatndus2005@naver.com
park10	aaaa1234	김종현	왕정면	서울시	010-111-112	rlatndus2005@naver.com
park11	aaaa1234	김종현	왕정면	서울시	010-111-112	rlatndus2005@naver.com
park1234	aaaa1234	김종현	왕정면	서울시	010-111-112	rlatndus2005@naver.com
park2	aaaa1234	김종현	왕정면	서울시	010-111-112	rlatndus2005@naver.com
park3	aaaa1234	김종현	왕정면	서울시	010-111-112	rlatndus2005@naver.com
park4	aaaa1234	김종현	왕정면	서울시	010-111-112	rlatndus2005@naver.com
park5	aaaa1234	김종현	왕정면	서울시	010-111-112	rlatndus2005@naver.com
park6	aaaa1234	김종현	왕정면	서울시	010-111-112	rlatndus2005@naver.com

Showing 1 to 10 of 13 entries

- 무한 insert 방지를 위한 redirect 방식 응답 설정
- 파일 업로드를 위해 MultipartFile 인터페이스 사용

P30

# 주요기능 - 관리자 페이지

V  
VARCHAR

## ManagerController.java

```
@RequestMapping(value="selectCar.do")
public String selectCar(CarVO cvo, Model model, HttpSession session) {
    cvo = managerService.selectOne(cvo);
    model.addAttribute("data", cvo);
    System.out.println("업데이트 : "+session.getAttribute("data"));
    return "manager.do";
}
```

## Manager.jsp

```
<!-- 연료 option 처리 -->
- 연료 <select name="cfuel" class="dataTable-input"
    id="ymin">
    <option
        <c:if test="${data.cfuel == '가솔린'}">selected='selected'</c:if>>가솔린</option>
    <option
        <c:if test="${data.cfuel == '디젤'}">selected='selected'</c:if>>디젤</option>
    <option
        <c:if test="${data.cfuel == 'LPG'}">selected='selected'</c:if>>LPG</option>
    <option
        <c:if test="${data.cfuel == '전기'}">selected='selected'</c:if>>전기</option>
    </select>
</li>
<li>주행거리<input class="dataTable-input" type="number" min="0" step="1000" max="2147483600"
    name="ckm" required value="${data.ckm}" autocomplete="off"></li>
<li>가격<input class="dataTable-input" type="number" min="0" step="100" max="2147483600"
    name="cprice" required value="${data.cprice}"
    autocomplete="off"></li>

```

## 관리자페이지 - 차량 조회



The screenshot shows the Manager.jsp page with a red box highlighting the dropdown menu for fuel type ('cfuel'). The menu contains five options: Gasoline, Diesel, LPG, Electric, and another Gasoline option. The first Gasoline option is selected.



The screenshot shows the Varchar application's car management interface. A red Maserati Quattroporte is displayed. To its right is a table titled 'Car Information' with the following data:

차량명	설명	연료	주행거리	가격	지역	연식
sca	ascas	가솔린	20000	2000	서울	2000
마세라티 기블리	보험이력0원/무사고/출고 당시 그대로	가솔린	50000	12000	서울	2017

A red box highlights the '마세라티 기블리' entry in the table. Below the table, a red arrow points from the highlighted entry to the car image above.

- 차량명 클릭시 해당 차량 정보 출력

# 주요기능 - 관리자 페이지

## ManagerController.jsp - update

```
@RequestMapping(value="managerUpdate.do", method=RequestMethod.POST)
public String updateCar(@ModelAttribute("data")CarVO cvo, MultipartHttpServletRequest request) {
    MultipartFile uploadFile = multipartRequest.getFile("file");
    if(!uploadFile.isEmpty()) { //업로드한 파일 존재여부 확인
        String fileName = uploadFile.getOriginalFilename(); //업로드한 파일명
        uploadFile.transferTo(new File(fileRoute+fileName)); //저장할 경로 결정
        cvo.setCimg("images/"+fileName);
        System.out.println("게시글 작성 파일이름 : "+fileName);
    }
    managerService.updateCar(cvo);
    return "redirect:manager.do";
}
```

- 차량 정보 수정 시 이미지 태그 내부에 required 속성 제거

## ManagerController.jsp - delete

```
@RequestMapping(value="deleteCar.do")
public String deleteCar(@ModelAttribute("data")CarVO cvo, SessionStatus sessionStatus) {
    System.out.println("삭제하니? :" +cvo.getCnum());
    sessionStatus.setComplete();
    managerService.deleteCar(cvo);
    return "redirect:manager.do";
}
```

## 관리자페이지 - 차량 정보 수정 or 삭제

The screenshot shows a car management interface. At the top, it says '관리자페이지 - 차량 정보 수정 or 삭제'. Below that is a section labeled 'Varchar'. The main area displays a red Maserati Ghibli. To the right are input fields for car details: 차량명 (Maserati Ghibli), 상세설명 (보행이력0원/무사고/출고 당시 그대로), 연식 (2017), 연료 (가솔린), 주행거리 (50000), 가격 (12000), 지역 (서울), and 이미지 (선택). At the bottom are two buttons: '수정하기' (highlighted with a yellow box) and '삭제하기' (highlighted with a red box).

- 차량 정보 삭제 시 세션에 저장되어있는 기존 정보를 삭제

# 문제 해결 - 관리자 페이지

## ManagerController.jsp - insert

```

@RequestMapping(value="insertCar.do", method=RequestMethod.POST)
public String insertCar(CarVO cvo, HttpServletRequest request, MultipartRequest multipartRequest) {
    String referer = (String)request.getHeader("Referer");
    System.out.println("referer 값 : "+referer);
    MultipartFile uploadFile = multipartRequest.getFile("file");
    System.out.println("파일 인서트 : "+uploadFile);
    try {
        if(!uploadFile.isEmpty()) { //업로드한 파일 존재여부 확인
            String fileName = uploadFile.getOriginalFilename();
            uploadFile.transferTo(new File(fileRoute+fileName));
            cvo.setCimg("images/"+fileName);
            System.out.println("게시글 작성 파일이름 : "+fileName);
        }
        managerService.insertCar(cvo);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return "manager.do";
}

```



```
return "redirect:manager.do";
```

- Insert 실행 후의 URL이 “insert.do”로 같다.
- 새로고침 시 Insert가 계속해서 실행됨
- 코드 내 경로를 “Redirect:”로 응답해 오류 해결

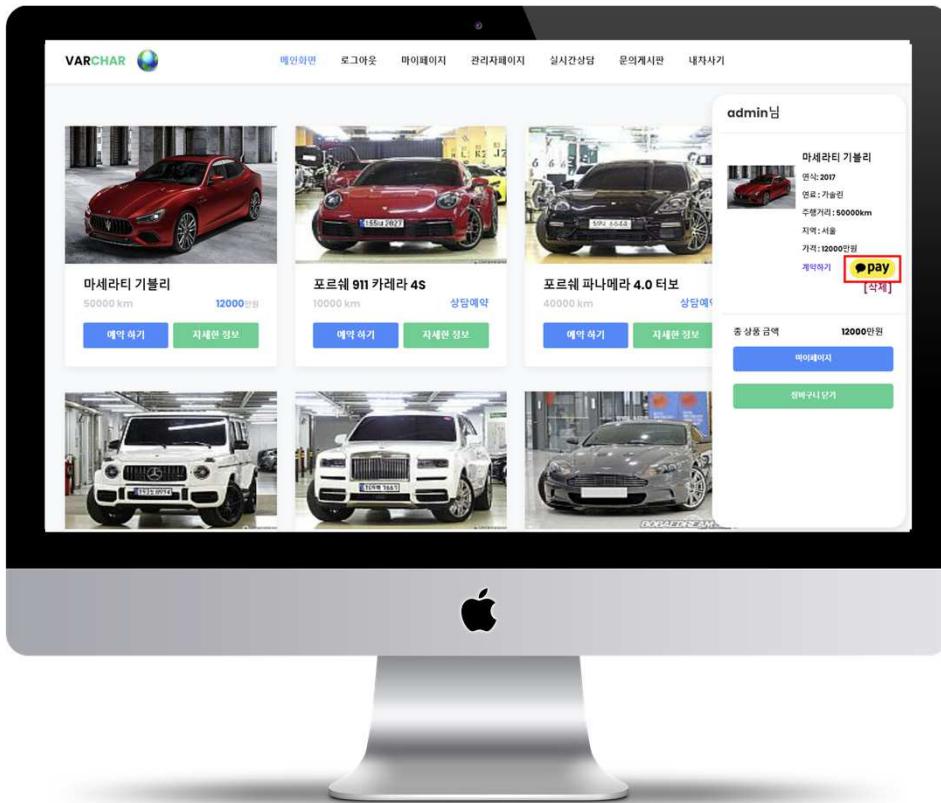
P33

# 주요기능 - 카카오페이지 API

V  
VARCHAR

## zzim.tag - kakaoPay

```
<span id="contractSpan"><spring:message code = "zzim.contractSpan" /></span>
<button id="kakaoPayBtn" type="button" onclick="requestPay('${c.ctitle}')">
    
</button>
```



## zzim.tag - kakaoPay js

```
<script>
var IMP = window.IMP; // 팝업창
/* IMP.init("..."); */
IMP.init("..."); // 아임포트 가맹점 식별코드

var today = new Date(); //일
var hours = today.getHours(); // 시
var minutes = today.getMinutes(); // 분
var seconds = today.getSeconds(); // 초
var milliseconds = today.getMilliseconds(); //밀리초
var makeMerchantUid = hours + minutes + seconds + milliseconds;

function requestPay(itemName) {
    console.log(itemName);
    IMP.request_pay({ // 요청하는 것들
        pg : 'kakaopay', //카카오페이지 API레핑
        /* kakao : 카카오페이지, */
```

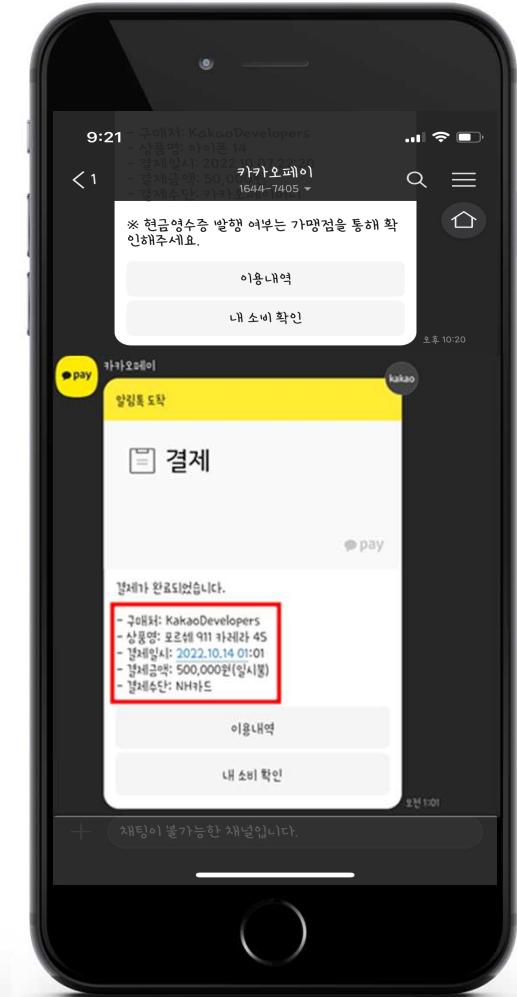
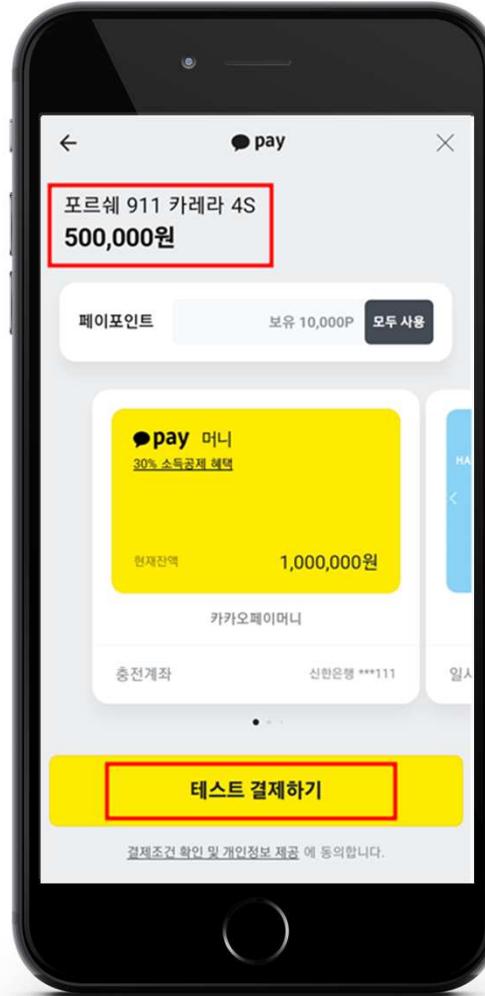


# 주요기능 - 카카오페이지 API

## zzim.tag - kakaoPay.js

```

merchant_uid: "IMP"+makeMerchantUid, // 상점에서 관리하는 상품번호
name : itemNm, // 상품명
amount : 500000, // 가격
buyer_email : 'limport@chai.finance', // 구매자 이메일
buyer_name : '아임포트 기술지원팀', // 구매자 이름
buyer_tel : '010-1234-5678', // 구매자 번호
buyer_addr : '서울특별시 강남구 삼성동', // 구매자 주소
buyer_postcode : '123-456',
/* m_redirect_url : 'https://www.yourdomain.com/payments/complete' */
function (rsp) { // callback
  if (rsp.success) {
    console.log(rsp);
    var msg = '결제가 완료되었습니다.\n';
    msg += '고유ID : ' + rsp.imp_uid+"\n";
    msg += '상점 거래ID : ' + rsp.merchant_uid+"\n";
    msg += '결제 금액 : ' + rsp.paid_amount+'원\n';
    msg += '카드 승인번호 : ' + rsp.apply_num;
  } else {
    console.log(rsp);
    var msg = '결제에 실패하였습니다.\n';
    msg += '에러내용 : ' + rsp.error_msg;
  }
  alert(msg);
}
  
```



- Name에는 인자 값으로 받아온 차량 명 삽입**
- Amount에는 계약금 50만원으로 고정**

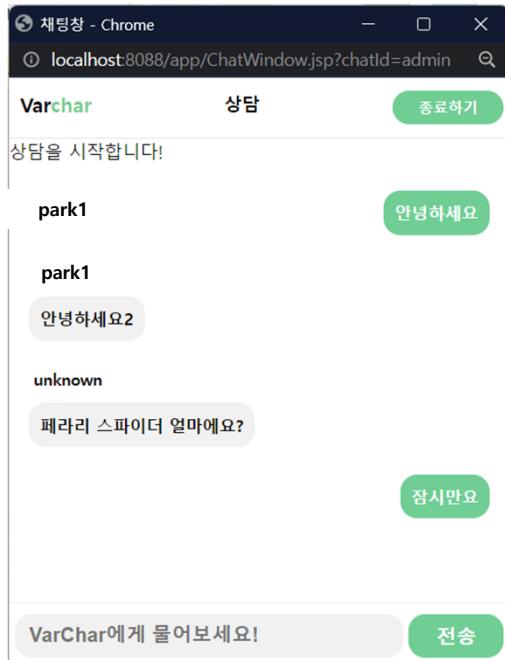
# 주요기능 - WebSocket

- 웹 소켓이란??

➤ 웹소켓은(WebSocket) 클라이언트의 요청에 응답한 후에도 연결을 그대로 유지하는 연결 지향 방식

- 다양한 WebSocket 구현 방식

➤ Polling, LongPolling → WebSocket



tomcat



spring

**Stomp**



STOMP

SockJS

```
web.xml
<context-param>
  <param-name>CHAT_ADDR</param_name>
  <param-value>ws://192.168.0.1:8088/app</param-value>
</context-param>
```

- 설정 파일에서 websocket 프로토콜로 통신할 수 있도록 설정해준다.

**ws://123.456.7.891:8088/app**

Protocol

대표적으로  
HTTP, HTTPS 방식

Host Name

도메인 이름 또는  
IP Address

Port

데이터를 받아야 하는  
프로세스 위치

Path

# 주요기능 - WebSocket

## ChatServer.java

```

@ServerEndpoint("/ChatingServer")
public class ChatServer {
    private static Set<Session> clients
        = Collections.synchronizedSet(new HashSet<Session>());
}

@OnOpen // 웹소켓 클라이언트 접속 시 실행
public void onOpen(Session session) {
    clients.add(session);
    System.out.println("웹소켓 연결:" + session.getId());
}

@OnMessage // 메시지를 받으면 실행
public void onMessage(String message, Session session) throws IOException {
    System.out.println("메시지 전송 :" + session.getId() + ":" + message);
    synchronized (clients) {
        for (Session client : clients) {
            if (!client.equals(session)) {
                client.getBasicRemote().sendText(message);
            }
        }
    }
}

@OnClose
public void onClose(Session session) {
    clients.remove(session);
    System.out.println("웹소켓 종료 :" + session.getId());
}

@OnError // 에러 발생 시 실행
public void onError(Throwable e) {
    System.out.println("에러 발생");
    e.printStackTrace();
}

```

### @ServerEndpoint

- 웹 소켓 서버의 요청명을 설정

EX) ws://123.456.7.89:8088/**/ChatingServer**

- **자바 웹 소켓 모듈 어노테이션**

#### @OnOpen

- 새로운 웹 소켓 세션이 열릴 때 자바 메소드를 호출하는 @

#### @OnMessage

- 자바 메소드가 웹 소켓 메시지를 수신하도록 하는 @

#### @OnClose

- 웹 소켓 세션이 닫힐 때 호출 자바 메소드를 호출하는 @

#### @OnError

- 에러 처리를 위해 자바 메소드를 호출하는 @

# 주요기능 - WebSocket

## ChatWindow.jsp

```
let webSocket
= new WebSocket("=<%= application.getInitParameter("CHAT_ADDR") %>/ChatingServer");
```

## ChatWindow.jsp

```
webSocket.onopen = function(event) {
    chatWindow.innerHTML += "상담을 시작합니다!<br/>";
};
webSocket.onclose = function(event) {
    chatWindow.innerHTML += "상담이 종료되었습니다!<br/>";
};
```

## ChatWindow.jsp

```
function sendMessage() {
    // 대화창에 표시
    chatWindow.innerHTML += "<p class='chatText mychat'><span class='chatTextMsg'>" + chatMessage.value + "</span></p>";
    webSocket.send(chatId + '!' + chatMessage.value);
    chatMessage.value = "";
    messagesWrap.scrollTop = messagesWrap.scrollHeight;
}
```

## ChatWindow.jsp

```
webSocket.onmessage = function(event) {
    let message = event.data.split("!");
    // 대화명과 메시지 분리
    let sender = message[0]; // 보낸 사람의 대화명
    let content = message[1]; // 메시지 내용
    if (content != "") {
```

## ChatWindow.jsp

```
webSocket.onerror = function(event) {
    alert(event.data);
    chatWindow.innerHTML += "상담 중 에러가 발생하였습니다.<br/>";
};
```

- 웹 소켓 세션 객체 생성

- Web.xml 파일에 초기화 매개 변수로 등록한 URL/ChatingServer로 웹소켓 서버 접속

- .onopen

- 자바스크립트의 이벤트 핸들러로 웹 소켓 서버에 연결됐을 때 실행

- .onclose

- 웹 소켓 서버와 연결이 끊을 때 실행

- .send(서버로 보낼 데이터)

- 웹 소켓 객체의 send 메소드를 호출하여 보낼 메시지 지정

- .onmessage

- 자바스크립트의 이벤트 핸들러로 메시지를 받을 때 실행

- .onerror

- 자바 스크립트의 이벤트 핸들러로 에러 발생 시 실행

# 주요기능 - WebSocket

## ChatWindow.jsp

```
window.onload = function() {
    chatWindow = document.getElementById("messages");
    chatMessage = document.getElementById("sender");
```

- document.getElementById()함수로 특정 HTML 태그 저장

## ChatWindow.jsp

```
function sendMessage() {
    // 대화창에 표시
    chatWindow.innerHTML += "<p class='chatText mychat'><span class='chatTextMsg'>" + chatMessage.value + "</span></p>";
    webSocket.send(chatId + ":" + chatMessage.value); // '대화명+메시지' 형태로 가공 후 서버로 전송
    chatMessage.value = ""; // 메시지를 전송한 후 입력창 비우기
    messagesWrap.scrollTop = messagesWrap.scrollHeight; // 스크롤바는 항상 아래로 위치
}
```

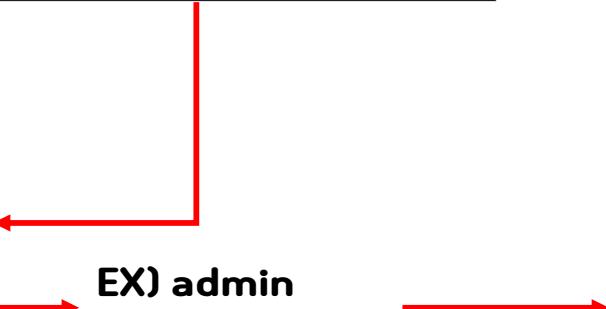
EX) admin 안녕하세요



```
@OnMessage // 메시지를 받으면 실행
public void onMessage(String message, Session session) throws IOException {
    System.out.println("메시지 전송 :" + session.getId() + ":" + message);
    synchronized (clients) {
        for (Session client : clients) {
            if (!client.equals(session)) {
                client.getBasicRemote().sendText(message);
            }
        }
    }
}
```

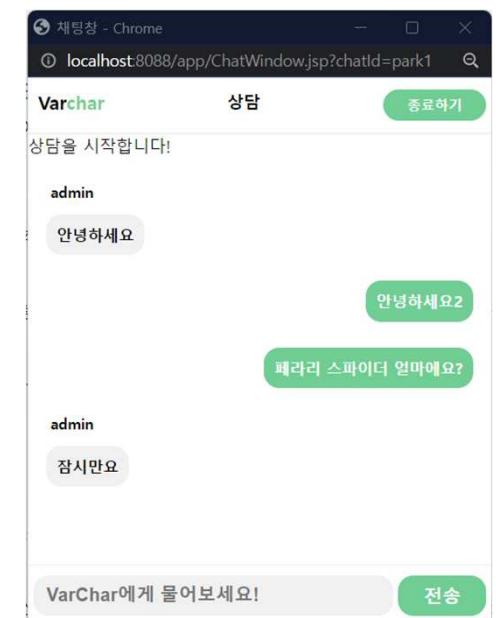
```
webSocket.onmessage = function(event) {
    let message = event.data.split(":");
    // 대화명과 메시지 분리
    let sender = message[0];
    let content = message[1];
    // 메시지 내용
    if (content != "") {
        if (content.match(/\w/)) { // 워드인지 찾는 함수
            if (content.match(chatId)) { // 나에게 보낸 메시지만 출력
                // let temp = content.replace("(" + chatId, "[귓속말] : ");
                let temp = content.replace("(" + chatId, "");
                chatWindow.innerHTML += "<p class='whoSends'><span>" + sender + "[귓속말] : " + temp + "</span></p>";
                chatWindow.innerHTML += "<p class='chatText'><span class='chatTextMsg whisperChat'>" + temp + "</span></p>";
            }
        } else { // 일반 대화
            chatWindow.innerHTML += "<p class='whoSends'><span>" + sender + "</span></p>";
            chatWindow.innerHTML += "<p class='chatText youchat'><span class='chatTextMsg'>" + content + "</span></p>";
        }
        chatWindow.scrollTop = chatWindow.scrollHeight;
        messagesWrap.scrollTop = messagesWrap.scrollHeight;
    }
};
```

EX) admin  
안녕하세요



## innerHTML이란?

- HTML 마크업 언어를 특정id값의 태그에 재할당 또는 추가



```
ChatServer.java
@ServerEndpoint("/ChatingServer")
public class ChatServer {
    private static Set<Session> clients
        = Collections.synchronizedSet(new HashSet<Session>());
```

- **Set배열을 사용한 이유**
  - ✓ 중복을 허용하지 않음
  - ✓ 무작위로 값을 담아 순서가 존재하지 않음

- **문제점**
  - ✓ 하나의 웹 소켓 세션에 하나의 사용자만 저장
- **원인 분석**
  - ✓ 하나의 사용자만 존재할 수 있으므로 상대방과 연결 불가능
- **문제 해결**
  - ✓ 여러 사용자가 동시에 접속해도 문제가 생기지 않도록 하여 문제 해결

# 문제 해결 - WebSocket

## ChatServer.java

```
// 웹소켓 서버에 연결됐을 때 실행
webSocket.onOpen = function(event) {
    // chatWindow.innerHTML += "웹소켓 서버에 연결되었습니다.<br/>";
    chatWindow.innerHTML += "상담을 시작합니다!<br/>";
};

// 웹소켓이 닫혔을 때(서버와의 연결이 끊겼을 때) 실행
webSocket.onClose = function(event) {
    chatWindow.innerHTML += "웹소켓 서버가 종료되었습니다.<br/>";
    chatWindow.innerHTML += "상담이 종료되었습니다!<br/>";
};

// 에러 발생 시 실행
webSocket.onError = function(event) {
    alert(event.data);
    chatWindow.innerHTML += "채팅 중 에러가 발생하였습니다.<br/>";
    chatWindow.innerHTML += "상담 중 에러가 발생하였습니다.<br/>";
};

// 메시지를 받았을 때 실행
webSocket.onMessage = function(event) {
    let message = event.data.split("|");
    // 대화명과 메시지 분리
    let sender = message[0];
    // 보낸 사람의 대화명
    let content = message[1];
    // 메시지 내용
}
```

```
webSocket.onopen = function(event) {
    // chatWindow.innerHTML += "웹소켓 서버에 연결되었습니다.<br/>";
    chatWindow.innerHTML += "상담을 시작합니다!<br/>";
};

// 웹소켓이 닫혔을 때(서버와의 연결이 끊겼을 때) 실행
webSocket.onclose = function(event) {
    chatWindow.innerHTML += "상담이 종료되었습니다!<br/>";
};

// 에러 발생 시 실행
webSocket.onerror = function(event) {
    alert(event.data);
    chatWindow.innerHTML += "상담 중 에러가 발생하였습니다.<br/>";
};

// 메시지를 받았을 때 실행
webSocket.onmessage = function(event) {
    let message = event.data.split("|");
    // 대화명과 메시지 분리
    let sender = message[0];
    // 보낸 사람의 대화명
    let content = message[1];
    // 메시지 내용
}
```

## 문제점

- ✓ 서버와 연결이 안되고 반응 없음

## 원인 분석

- ✓ 웹소켓 객체가 생성되면 서비스를 위해 4개의 이벤트 핸들러가 소켓 통신이 가능

## 문제 해결

- ✓ 이벤트 핸들러의 정확한 함수명을 명시적으로 변경



VarChar에게 물어보세요!

전송

정상

VarChar에게 물어보세요!

전송

비정상



- 1대 N대 아닌 1대 1 방식의 상담
  - 사용자의 개인정보 보호 향상
  
- 채팅방으로 구성되어 있으며 채팅방을 나가도 대화내용 기억
  - SNS DataBase 구축 계획

P42



시연

2

## 1. VARCHAR 사이트 시연

# Q&A

질문 있습니까?!

?????! ??? ??????????????????  
?????? ?? ???

# 소감



김수연

spring에서 JdbcTemplate, Mybatis를 사용하면서 길었던 코드가 간결해지는 것을 보며 spring의 매력을 느낄 수 있었고, 팀원들의 역량과 원활한 커뮤니케이션으로 최종 프로젝트가 잘 완성될 수 있었습니다. 또한, 다른 팀원들이 구현했던 코드들을 보면서 많이 배웠던 시간이었습니다. 코알라조 화이팅...!



임한국

최종 프로젝트를 진행하며 블로그도 다시 한 번 돌아보게 되고 이관작업 및 여러 API를 구현하는 좋은 경험 이었습니다. 항상 팀원들 모두 의욕적으로 프로젝트에 역량을 기여해주어서 함께라는 힘이 얼마나 중요한지 다시 한 번 더 깨닫게 되었고 부족한 부분들도 직접 해보는 시간을 가지며 채워나갈 수 있는 계기가 되었습니다.



이준선

6개월 이란 시간동안 Java 부터 시작하여 Spring 프로젝트 까지 끝 마쳤다는 사실에 뿌듯하고 처음 시작 할때 보다 많은 것 을 알게되고 팀원들과 프로젝트를 만들어 혼자 공부 할 때보다 많은 점을 보완 할 수 있어서 좋았습니다



김종현

6개월 동안 함께 고생하고 많은 도움을 주신 코알라 조원 들, 강사님, 그리고 다른 학우분들께 감사의 말씀 드립니다. 다음 뜻하는 바를 이루시고 언젠가 또 만났으면 좋겠습니다.



이향준

프로젝트를 진행하면서 목표하는 기능을 구현하기 위해 수업 외의 다양한 부분을 학습해볼 수 있는 좋은 기회가 되었습니다. 이로 인해 지금까지 배웠던 것을 기반으로 앞으로 더 많은 부분에 대해 스스로 학습하고 기능을 구현할 수 있겠다는 자신감을 가지게 되었습니다. 또한 팀원들과 구현한 기능에 대해 피드백과 코드 리뷰를 하면서 이전에 비해 많은 성장을 했음을 느꼈습니다..



황지민

기존 프로젝트를 다른 언어로 이관하고 관리자 페이지를 구현하며 유지보수성을 높이는 동시에 결합도가 낮은 자바 로직 작성법을 이해할 수 있었습니다. 학습하지 못했던 로직, 문법들을 학습 및 작성해 코드를 분석할 수 있는 능력이 이전보다 더 성장함을 느껴 백엔드 개발자에 대한 확신을 가지게 됐습니다.

**발표 마치겠습니다.**

**감사합니다.**