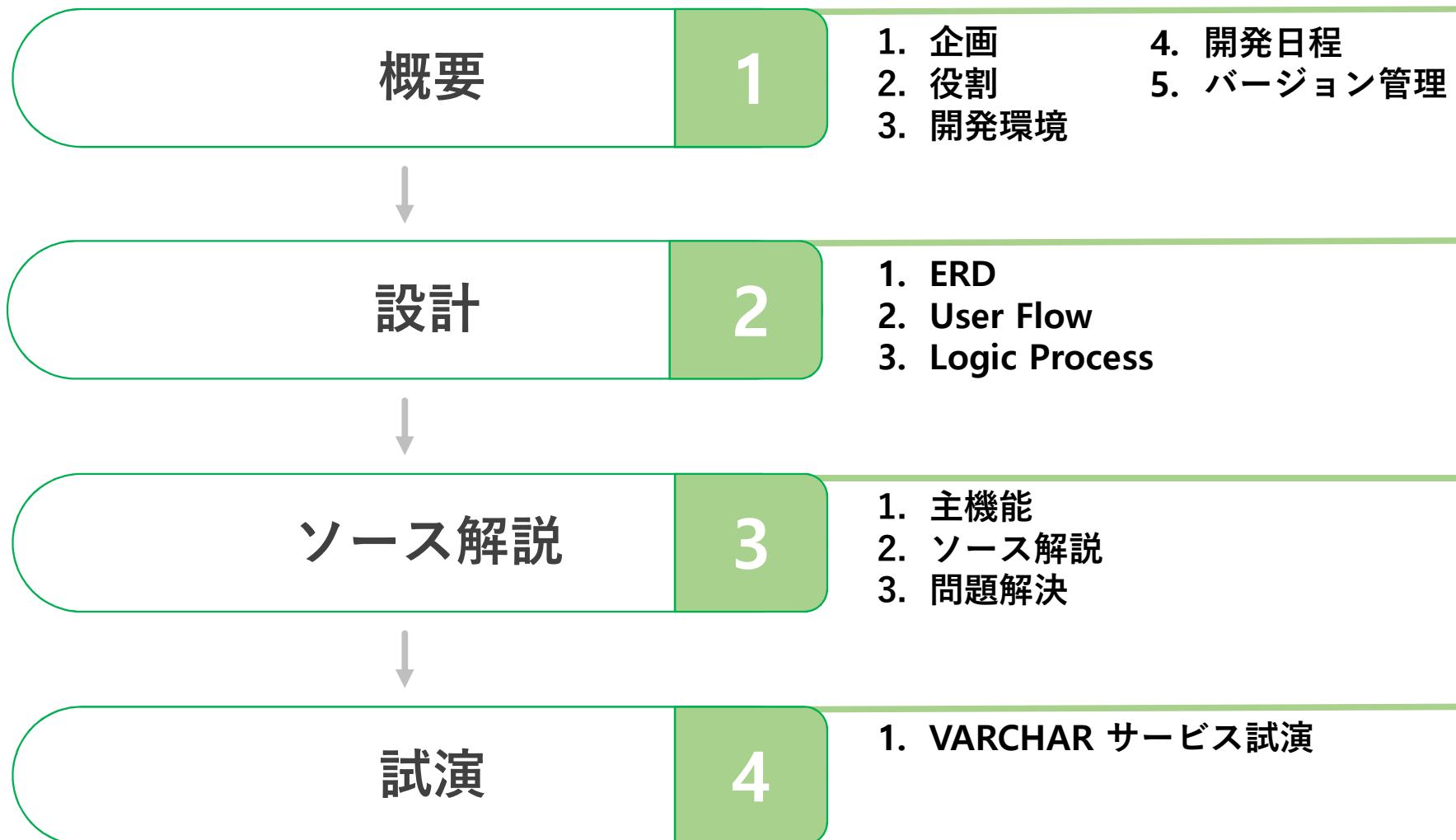


Spring 基盤中古車販売
ウェブプロジェクト

VC
VARCHAR

코알라

김수연, 김종현, 이준선, 이향준, 임환욱, 황지민



P02

V
VARCHAR

概要

1

1. 企画

2. 役割

3. 開発環境

4. 開発日程

5. バージョン管理

6. ページ構成



- **企画背景**
 - 最近オンライン中古車市場の活発化
- **企画目的**
 - 2Layered-architecture based Spring 移管作業
- **期待効果**
 - Mybatisによるより繊細な検索機能
 - WASを利用しリアルタイムチャット機能実現
 - 多国語処理によるウェブ接近性向上

開発環境

DBMS

ORACLE



Server



framework



使用言語



総合開発環境 IDE



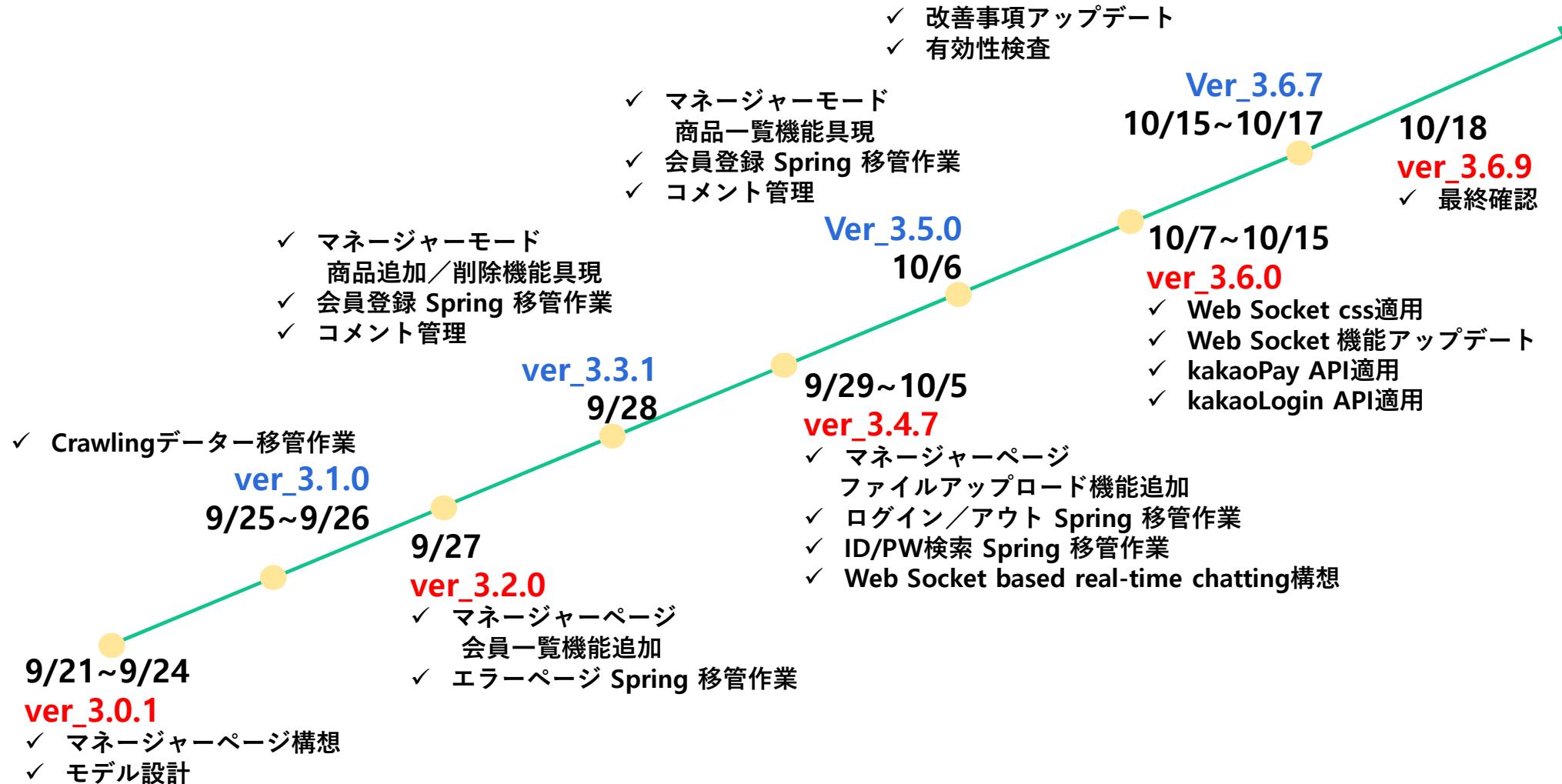
Visual Studio Code

バージョン管理



疎通及び協業





設計

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

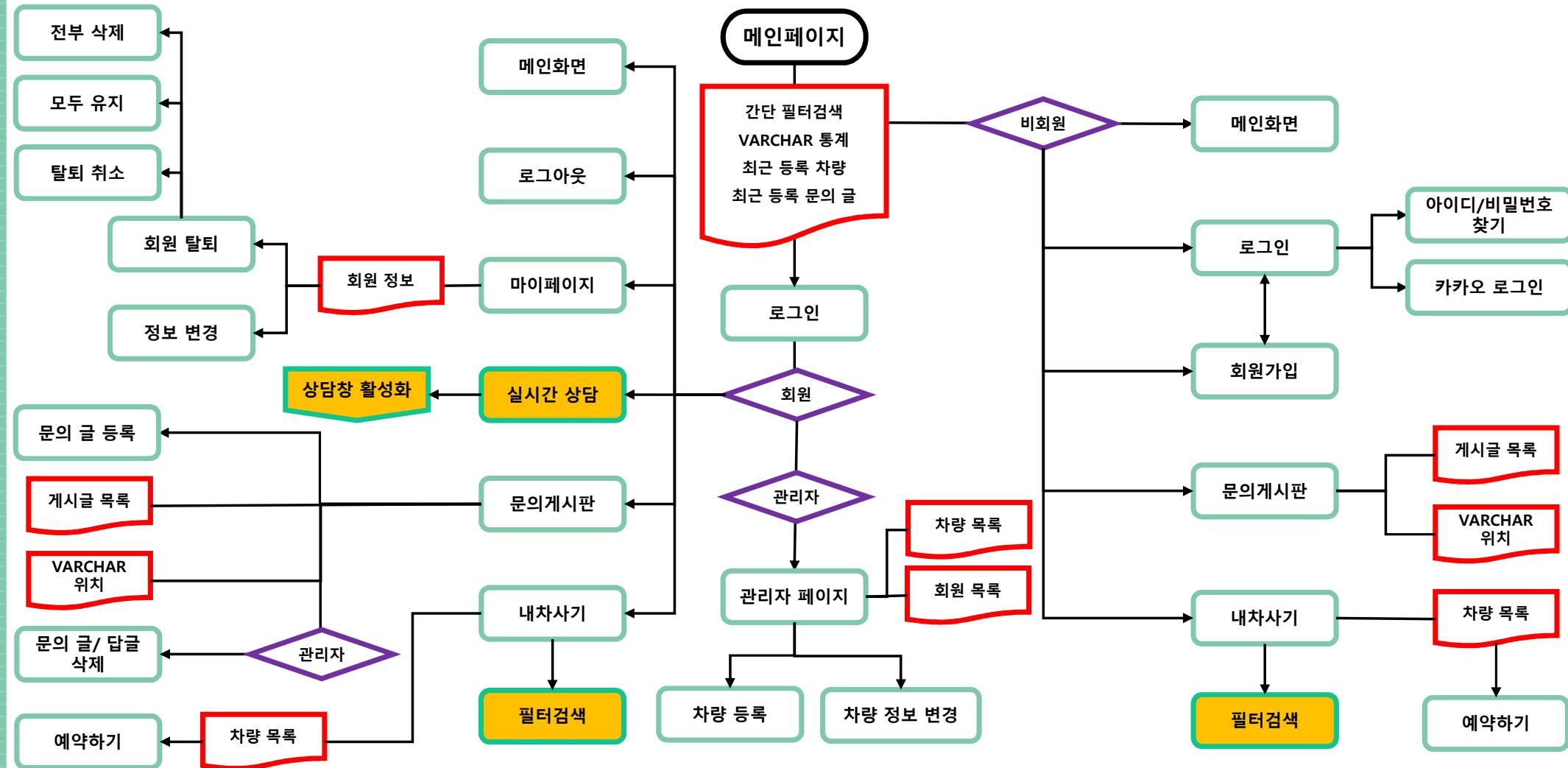
C REPLY			
답글 고유번호	RID	INT	PK
사용자 아이디	MID	VARCHAR(10)	NOT NULL
글 고유번호	BNUM	INT	NOT NULL
답글 내용	CMSG	VARCHAR(500)	NOT NULL
제약조건	CONSTRAINT CBOARD_CREPLY FOREIGN KEY(BNUM) REFERENCES CBOARD(BNUM) ON DELETE CASCADE		

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

P01

User Flow

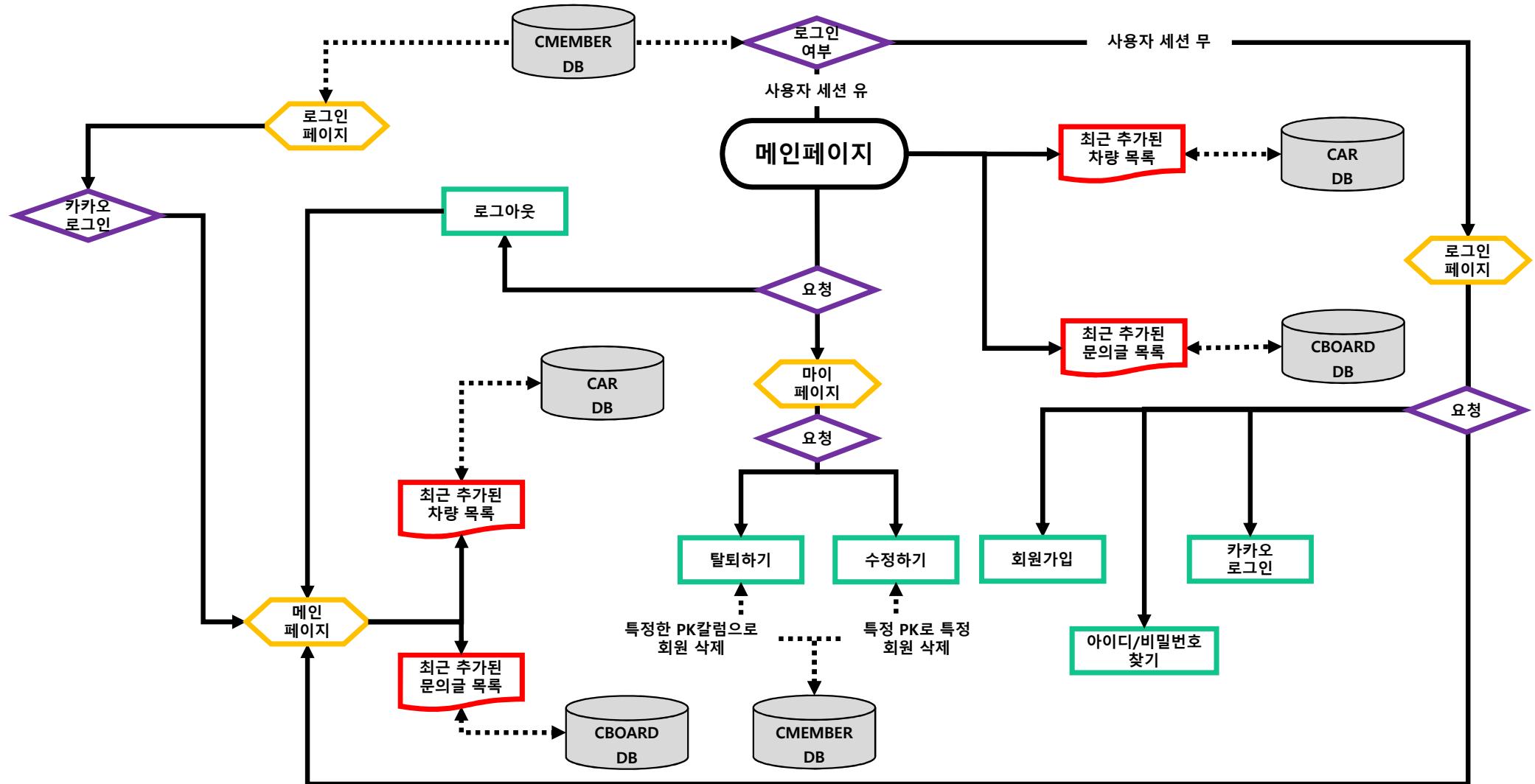
V
VARCHAR



P01

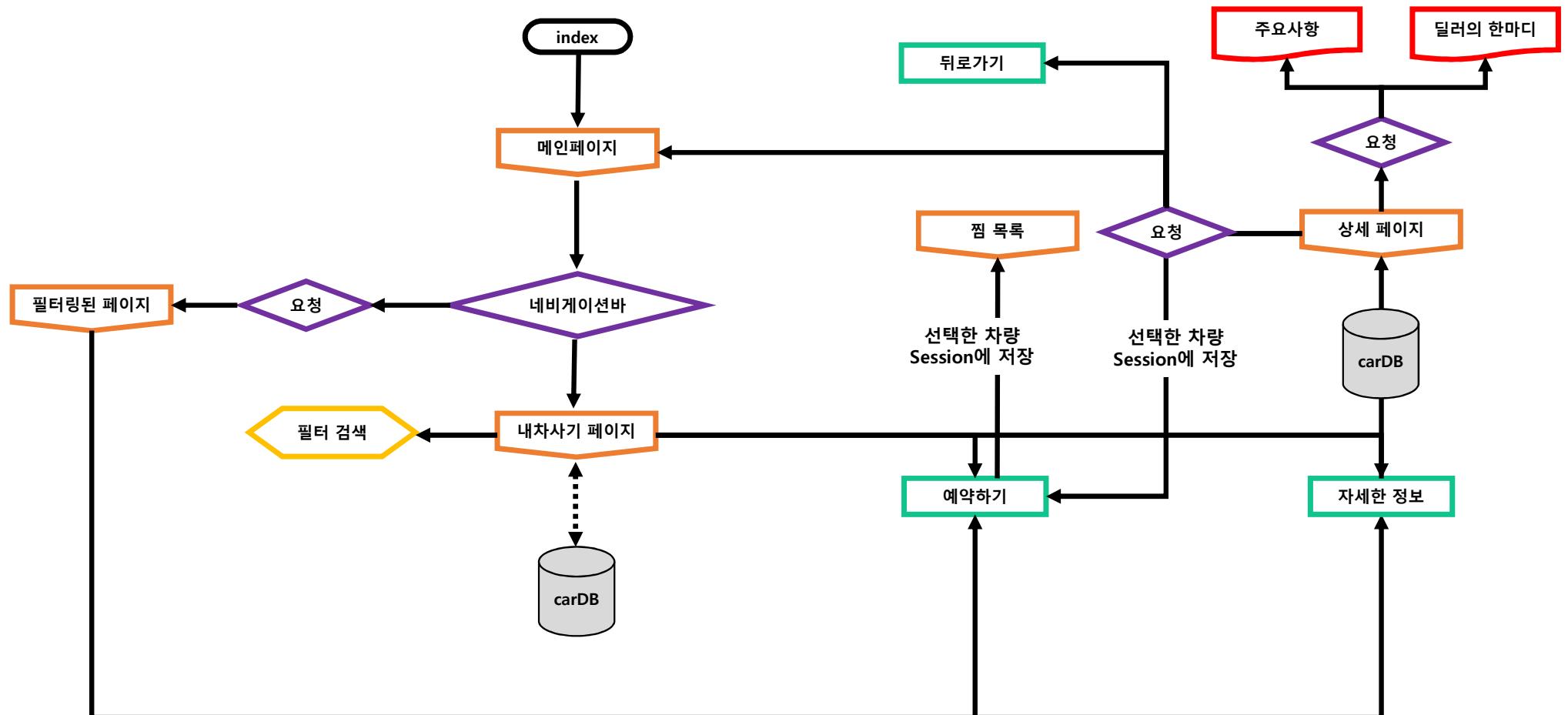
Logic Process – main / member

V
VARCHAR



P01

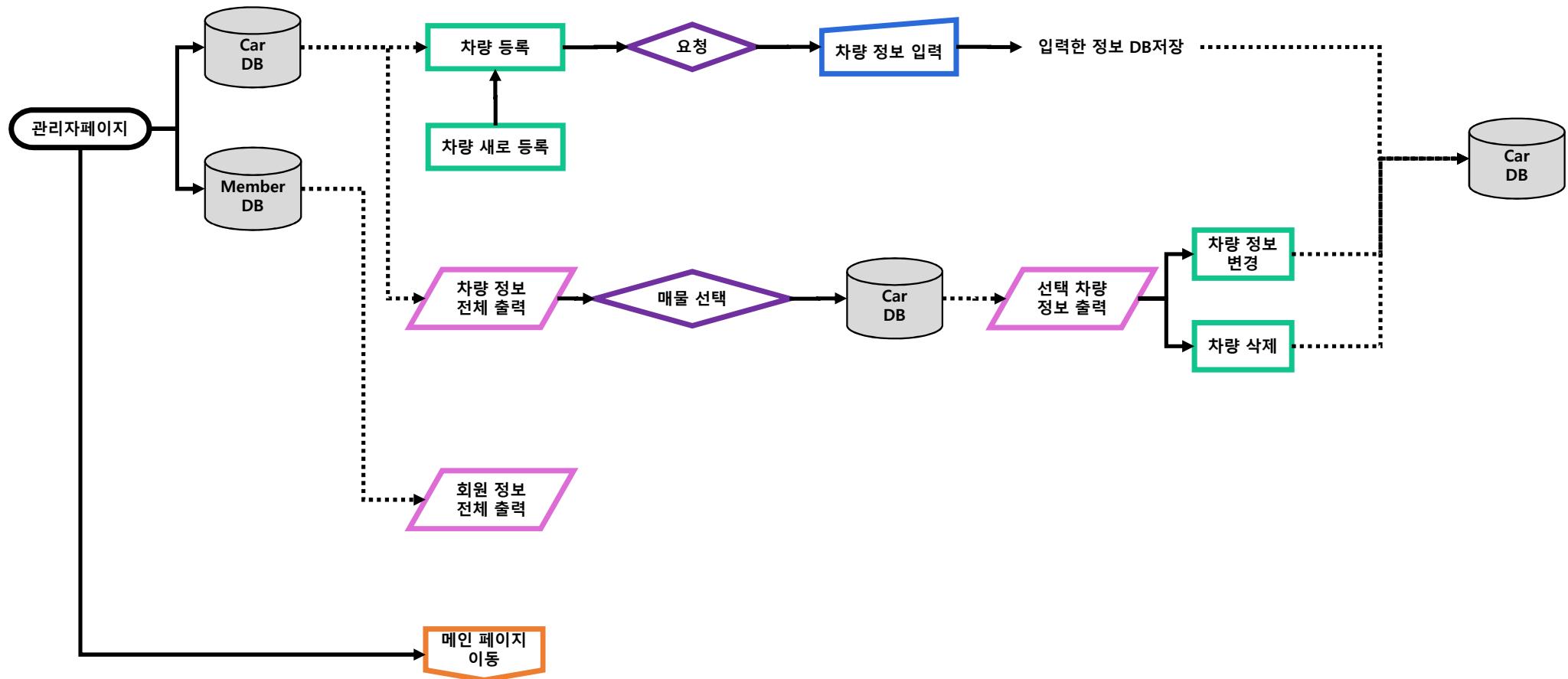
Logic Process – 商品一覧ページ

VC
VARCHAR

P01

Logic Process – マネージャーページ

V
VARCHAR



ソース解説

2

1. 主機能

2. ソース解説

3. 問題解決

P01

主機能 - もっと見る



메인화면

로그인

회원가입

문의게시판

내차사기

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

商品をもっと持ち込む

총 12/70 개의 상품을 보고 있습니다.

View more

現在検索条件反映必須

データーが無ければ非活性

Cart

P01

主機能 – もっと見る_MySQL 移管作業

中古車商品もっと見る : 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();
```



中古車商品もっと見る: MySQL ver.

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

固定値： 1 2

現在検索条件 & 商品範囲 非同期要請 : 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

- ROWNUM into LIMIT
- ROWNUM BETWEEN 1 AND 12 → LIMIT 0, 12
- DB後により送信する範囲データー変更

P01

主機能 – もっと見る_Spring 移管作業

CarSearchAction.java (JSP Project)

```
// 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("가격 : "+price_min+"~"+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("vmax") != null) {
    int year_min = Integer.parseInt(request.getParameter("vmin")); //연식의 최소 값
    int year_max = Integer.parseInt(request.getParameter("vmax")); //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 Objectの自動binding

- JSP Servletでは Parameterをもらい VO Objectに直接set
- Spring ContainerはParameterがVO Fieldに存在すると Command Objectに自動 binding

→ Useless codes in the Spring Framework

P01

主機能 – もっと見る_Spring 移管作業

```

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
@ResponseBody
@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);
}

```

※ JSONParserを通じたデーターパーシング

- 文字列化したJS配列非同期転送
- JSONParserはもらった文字列の JSON, XMLデーターをJavaで利用可能な形に
- 言語の違いによるデーター加工過程を担当

Stringified JS Array ArrayList in Java
 “[‘Seoul’, ‘Busan’]” → [“Seoul”, “Busan”]

JSON Parser

主機能 – もっと見る_Spring 移管作業

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 항목 데이터 개수 계산

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

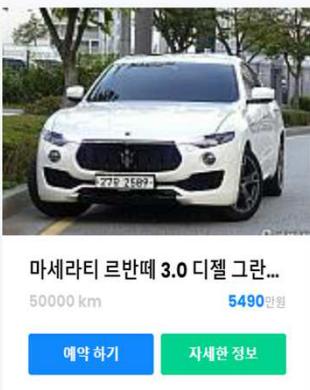
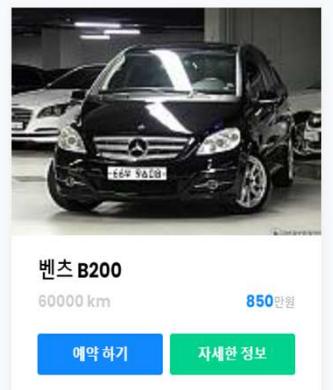
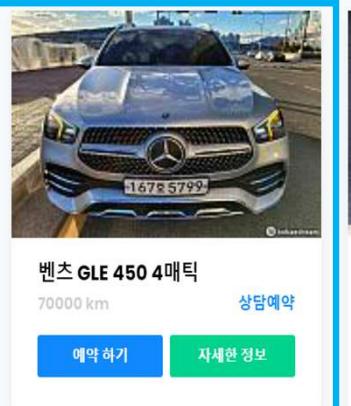
//     svo.setRange2(range2);
List<CarVO> nextDataList = searchService.selectAll(svo);
// 더 보여줄 데이터가 없다면 --> 더보기 버튼 비활성화
if(nextDataList.size() == 0) {
    showMore = false;
}
```

※ ‘もっと見る’ボタン活性化の判定

- 維持補修の為もっと見せる数constant化
- 予め次の範囲の商品数計算 → 活性化判断

P01

主機能 - もっと見る_Spring 移管作業



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";
}
}
```

<div id="viewItems">

主機能 – もっと見る_Spring 移管作業

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
        //필터를 사용할 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(",");
            }
            //cfuelSql 객체 = AND CFUEL IN ('for문을 통해 나온 필터배열값')
            CfuelSql = "AND CFUEL IN ("+cfuelSb.toString()+"')";
        }
        else{
            CfuelSql = "";
        }
    }
}

```

JSP Servletでは、DAO Objectの毎回生成

Springでは検索条件が無い場合、直接空白の値を付与

※ Spring Containerの Singleton Pattern

- 検索条件を消し検索しても、直前の検索結果維持
- JSP Servletでは、DAO Object生成ごとに上部の文字列が空白の値で初期化
- Spring ContainerはSingleton Patternを維持するので改めてObjectを生成しないこと

主機能 - Session_Spring 移管作業

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
    }
}

```

※ Login/Logout Session 利用

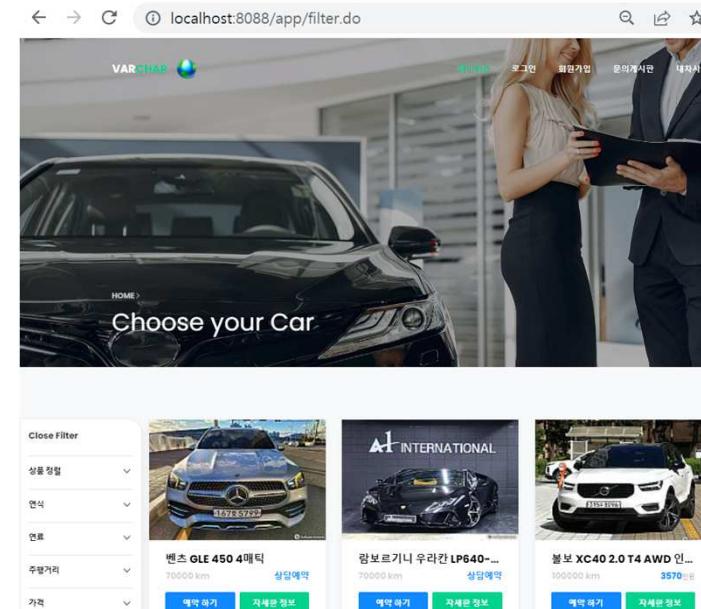
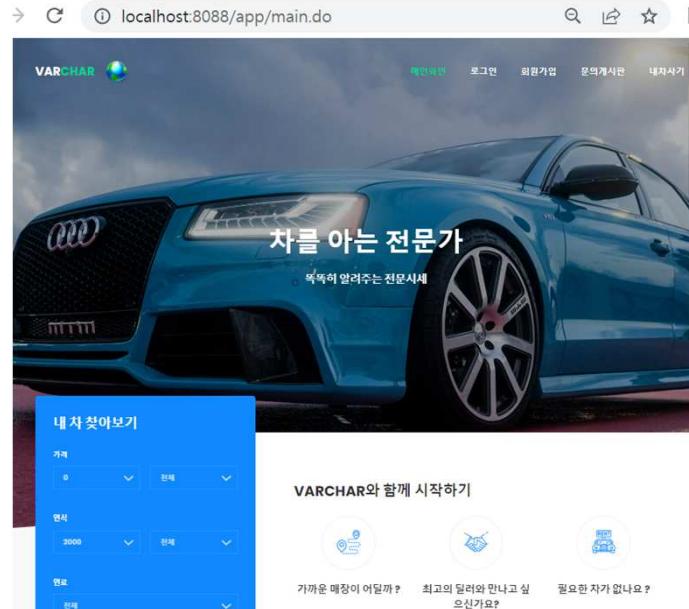
- 保安の為 Sessionには
利用者の情報中IDのみセット
- Spring ContainerのSingleton Pattern

→ @SessionAttributes & Model Objectを
通じてSession活用

→ Session全てを削除することではなく、
中のModel Objectのみクリーンする為
SessionStatus利用

P01

主機能 - ページ維持



```
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 Headersの refer

- 直前の要請確認可能

→ Referの情報で適材適所にページ維持可能

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

P01

主機能 – 多国語処理

イベント追加



- ページ上部に多国語処理機能追加
- ページ維持を利用し全ページにて多国語処理可能

中国語 ver.



日本語 ver.



問題解決 – 多国語処理

エラー詳細

```
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 변경시 기준 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);
```

直前 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];
```

※ 多国語処理 要請 問題

- 要請が環境設定を担当するXMLを必ず経由するべき
- その他、ページ維持や言語パラメータ管理をより用意にするようソース作成

P01

主機能 – 商品検索_MyBatis 移管作業

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]'번
    if(i+1 < svo.getFuelList().size()) //만약 필터 배열의 길이가 i+1보다 크면
        cfuelSb.append(","); // 중간에 ','를 붙여줌

}
System.out.println("DAO 로그 cfuelSb append " + cfuelSb.toString());
}

//cfuelSql 객체 = AND CFUEL IN ('for문을 통해 나온 필터배열값')
CfuelSql = "AND CFUEL IN (" + cfuelSb.toString() + ")";
}

```

- Java 基盤商品検索
- 検索条件多重選択の場合リストを媒介にして
文字列の関数を使い触接SQL文の文字列を完成

filter-mapping.xml - Mybatis 移管

```

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

```

- 既存Java 基盤商品検索MyBatisバージョンに移管
- 検索条件多重選択ケースMyBatisのForeach文法で明瞭に具現
- Separator属性が既存の文字列加工処理を代替

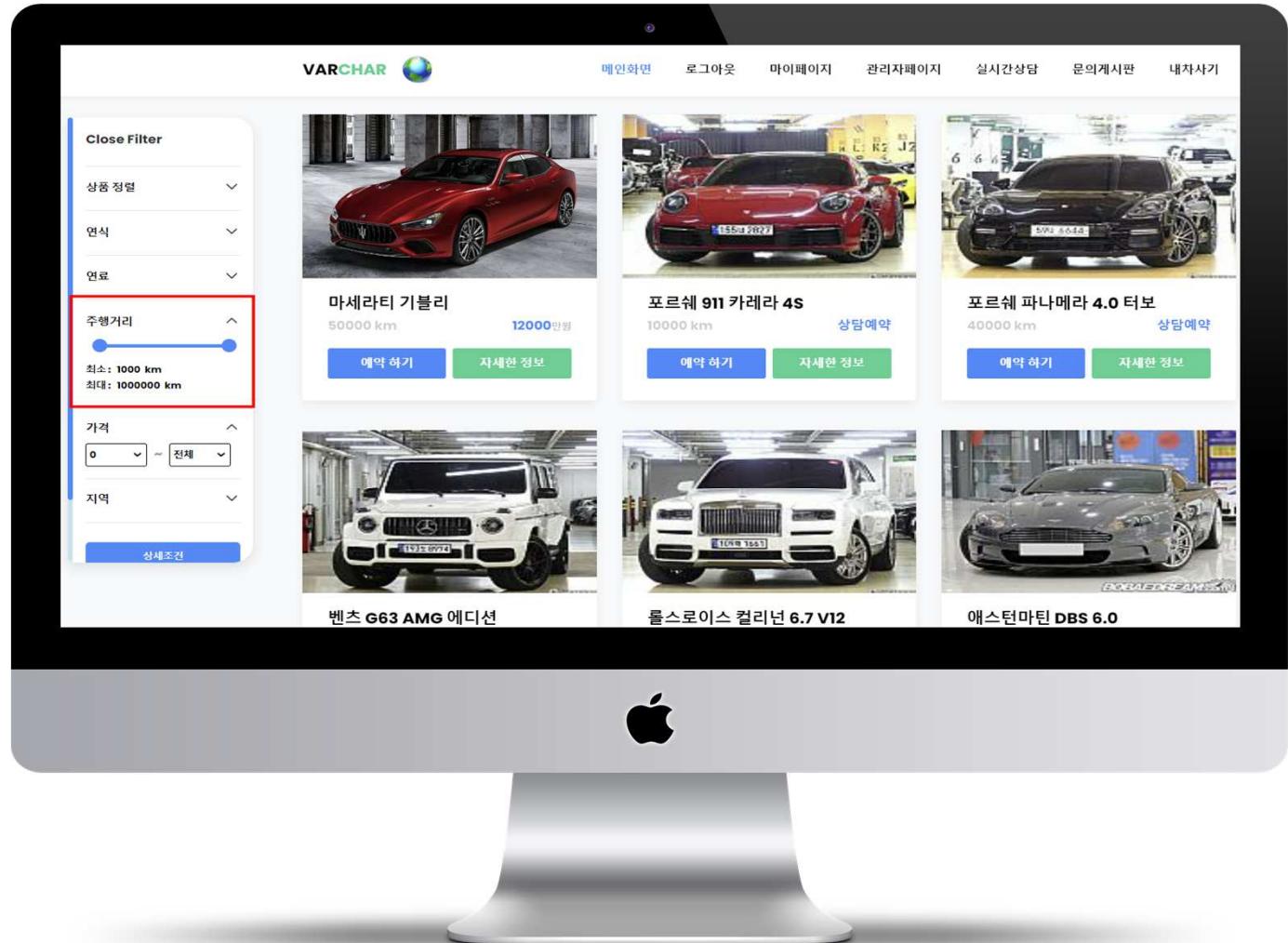
P01

主機能 – 商品検索_MyBatis 移管作業



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

- 配列ではない、
一般パラメータの場合
(走行距離、年式、値段)
- 有効性検査を通過すると
SQL文追加



主機能 – マネージャーページ

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>
```

- 整数データは直接入力
- 他のタイプの値はオプションタグでセット

マネージャーページ

연료	가솔린 가솔린 디젤 LPG 전기
지역	서울
이미지	파일 선택 선택된 파일 없음
등록하기	

マネージャーページ

주행거리	5000
가격	
지역	서울
이미지	파일 선택 선택된 파일 없음
등록하기	

P01

主機能 - マネージャーページ



```

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";
}

```



- 無限 insert 予防を図り redirect 応答
- Image File Uploadの為MultipartFile Interface使用

マネージャーページ						
Varchar						
차량명	상세설명	연식	연도	기술인	주행거리	가격
BMW X5 xDrive 50i	리어 모니터/후진시	2000	기술인	240000	1580	경기
BMW X6 xDrive 50i	제작 후기형 라이트+M 버전 프론트 범퍼 개조	2011	기술인	9000	2147483647	2019
BMW X5 xDrive 50i	BMW X5 xDrive 50i LP770-4	2022	기술인	6000	2147483647	서울
BMW X5 xDrive 50i	BMW X5 xDrive 50i 우수스 4.0 V8	2018	기술인	40000	2147483647	경기
BMW X5 xDrive 50i	BMW X5 xDrive 50i 우수스 4.0 V8	2015	기술인	80000	2147483647	경기
BMW X5 xDrive 50i	BMW X5 xDrive 50i 우수스 4.0 V8	2015	기술인	31000	2147483647	서울
BMW X5 xDrive 50i	BMW X5 xDrive 50i 우수스 4.0 V8	2022	기술인	17000	2147483647	서울
BMW X5 xDrive 50i	BMW X5 xDrive 50i 우수스 4.0 V8	2022	기술인	10000	2147483647	서울
BMW X5 xDrive 50i	BMW X5 xDrive 50i 우수스 4.0 V8	2022	기술인	8000	2147483647	서울
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
park12	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

P01

主機能 - マネージャーページ



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 -->
<li>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>
```

商品詳細一覧

Varchar

새로 등록하기

차량명: 마세라티 기블리
상세설명: 보험이력0원/무사고/출고 당시 그대로
연식: 2017
연료: 가솔린
주행거리: 50000
가격: 12000
지역: 서울
이미지: 파일 선택 선택된 파일 없음

[수정하기](#) [삭제하기](#)

Car Information

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

Showing 71 to 72 of 72 entries

- 商品名クリックで詳細確認可能

主機能 - マネージャーページ

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";
}

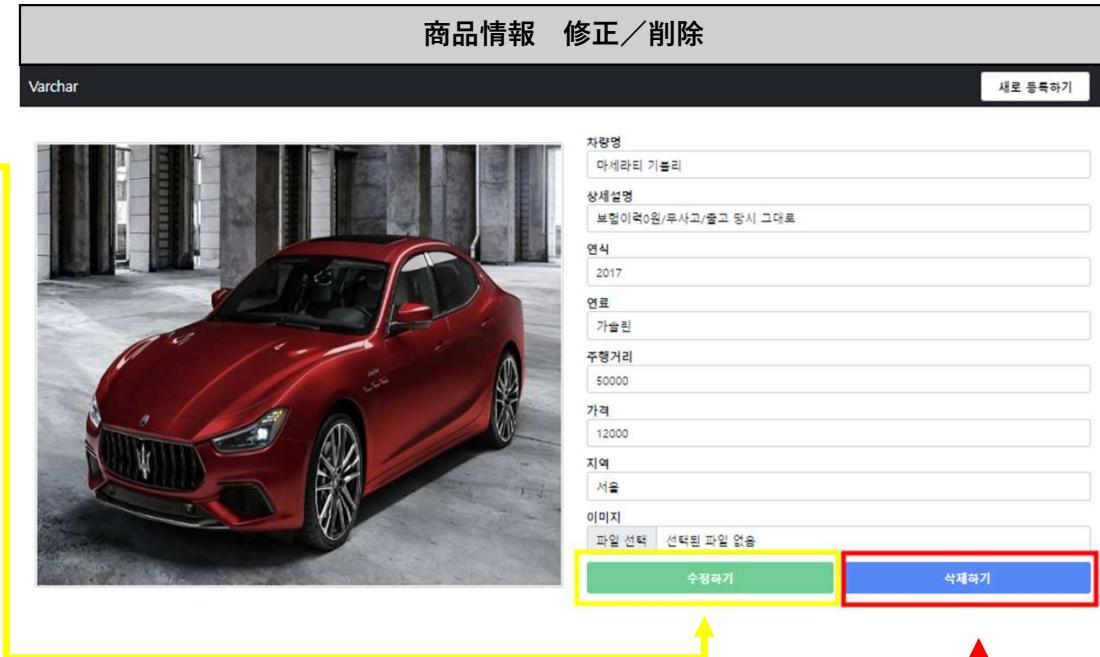
```

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";
}

```



- **@ModelAttributeを利用してNULL UPDATE 予防**

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";
}
```

- データーInsert後URLが以前同様“manager.do”
- 入力したデーターのInsertが続く
- “Redirect:” 応答で解決

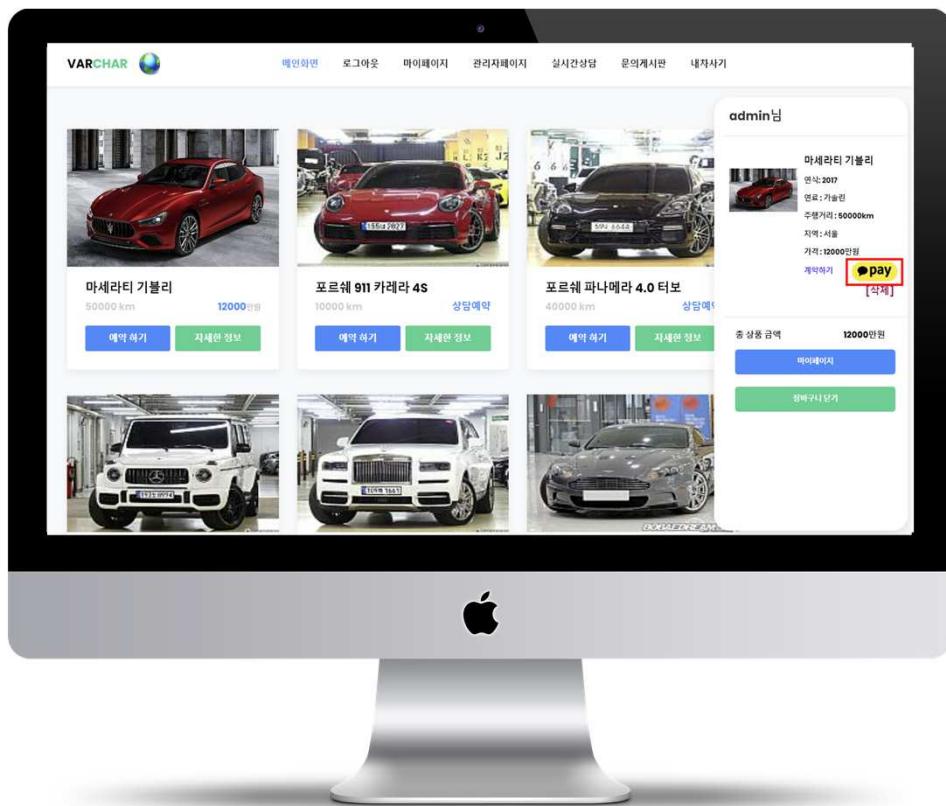
P01

主機能 – Kakao Pay API



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,
        'naver' : 네이버페이 */
```



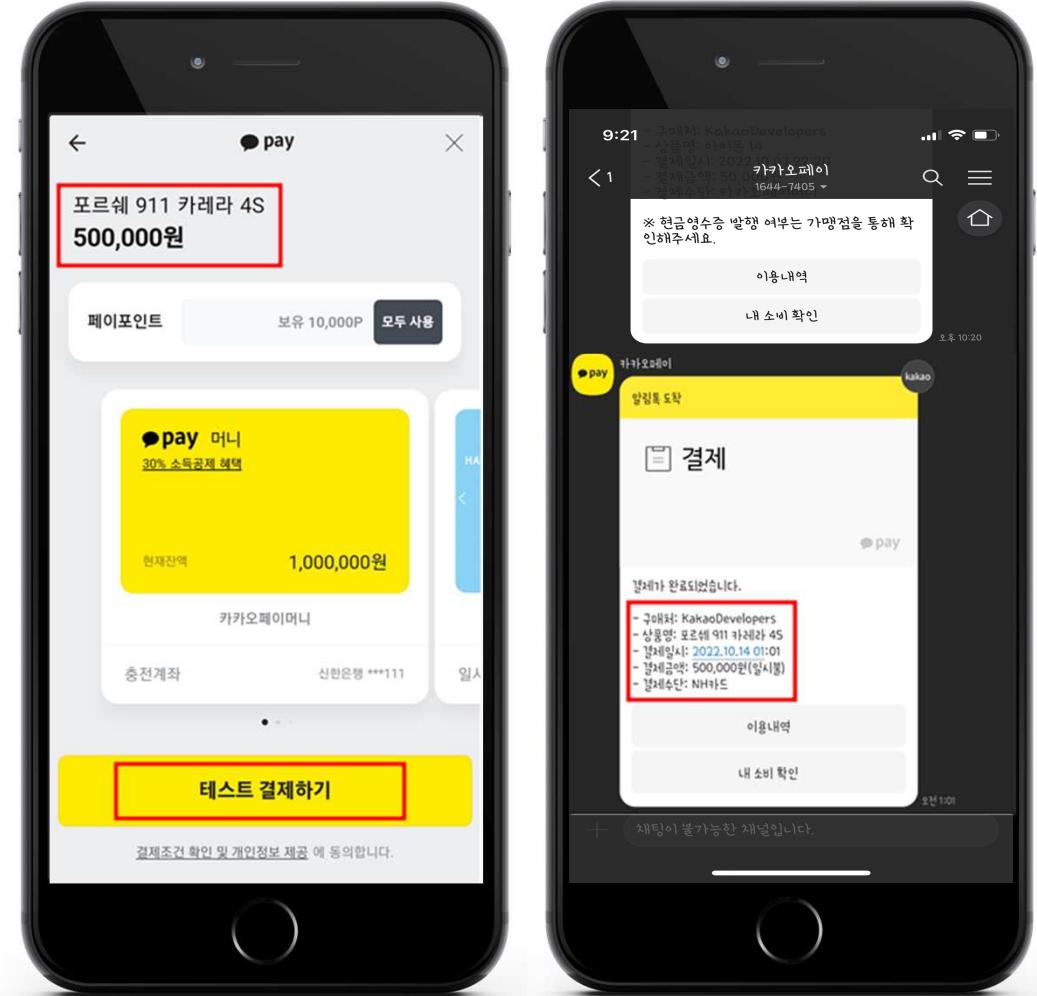
P01

主機能 – Kakao Pay API



```
zzim.tag - kakaoPay js

merchant_uid: "IMP"+makeMerchantUid, // 상점에서 관리하는 상품번호
name : itemName, // 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);
}
```



- Nameにはパラメータとして受信した商品名
- Amountは頭金50万ウォン固定

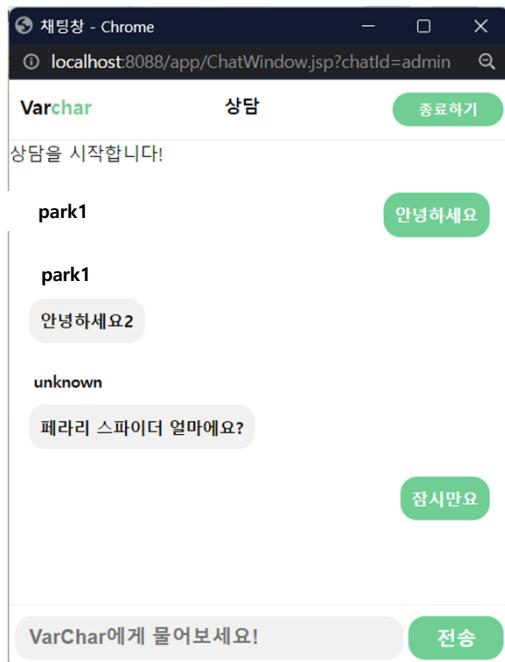
主機能 - WebSocket

- **WebSocket??**

- WebSocketはクライアントの要請に応答した後もそのまま接続を維持する通信

- 色々なWebSocket具現方法

- Polling, LongPolling → WebSocket



tomcat



spring



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>
```

- 設定ファイルでwebsocket protocol
通信設定

ws://123.456.7.891:8088/app

Protocol

主に
HTTP, HTTPS

Host Name

ドメイン名 or
IP Address

Port

データーをもらう
プロセス位置

Path

主機能 - WebSocket

ChatServer.java

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

@ServerEndpoint

- サーバーの要請名を設定

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

```
@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();
}
```

• Java WebSocket Module Annotation

@OnOpen

- 新たなWebSocket Sessionが開かれた時method遂行 @

@OnMessage

- Java methodがWebSocketメッセージを受信するための@

@OnClose

- WebSocket Sessionが閉ざされる時method遂行@

@OnError

- エラー処理の為Java method @

主機能 - 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/>";
};
```

- WebSocket Session Object 生成**

- Web.xml に登録したURL/ChatingServerでサーバー接続

- .onopen**

- サーバーに繋がる時実行

- .onclose**

- サーバーとの接続が途切れる時実行

- .send(サーバーに送るデータ)**

- WebSocket Objectのsend関数を通じて送るメッセージ作成

- .onmessage**

- メッセージをもらう時実行

- .onerror**

- メッセージを受信中エラー発生の際実行

主機能 - WebSocket

ChatWindow.jsp

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

- DOM Selectorを使い入力欄とメッセージボックスを管理

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("!");
    let sender = message[0]; // 대화명과 메시지 분리
    let content = message[1]; // 보낸 사람의 대화명
    if (content != "") {
        if (content.match("/")) { // 컷속말 / match함수는 동일한 단어를 문자열에서 찾는 함수
            if (content.match("(" + chatId)) { // 나에게 보낸 메시지만 출력
                let temp = content.replace("(" + chatId, "[컷속말] : ");
                let temp = content.replace("//" + chatId, "");
                chatWindow.innerHTML += "<p class='whoSends'><span>" + sender + "[컷속말]</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



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

- **Set配列の使用**

- ✓ 同時に同じ値は処理しない
- ✓ 順序が存在しない

- **問題点**

- ✓ 一つのサーバーに一つのセッションのみ

- **原因**

- ✓ セッションは一つの会員情報のみ登録

- **解決**

- ✓ 複数のクライアントが利用できるよう適合するCollection使用

問題解決 - 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
};
```

問題点

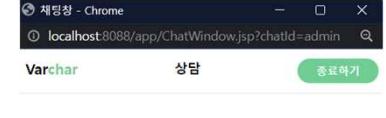
- ✓ サーバーとの繋がりに問題発生

原因

- ✓ Web Socket Objectが生成されるとサービスの為
四つにイベントハンドラー利用可能

解決

- ✓ イベントハンドラーに適切な関数名を作成



VarChar에게 물어보세요!

전송

VarChar에게 물어보세요!

전송

正常

異常

問題解決- 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 받아올 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
}
```

- 問題点

- ✓ サーバーとの繋がりに問題発生

- 原因

- ✓ Web Socket Objectが生成されるとサービスの為
四つにイベントハンドラー利用可能

- 解決

- ✓ イベントハンドラーに適切な命令を挿入



P02

V
VARCHAR

試演

2

1. VARCHAR ウェブサイト試演

Q&A



P02

FINISH

VC
VARCHAR

有難うございました。