

2조 그룹 프로젝트



# Spring MVC Project

- 기획
- 구성
- 시연

- 신은채
- 박승원
- 안병찬
- 허 훈

# 기획

## Planning

01

02

03

04

05

## 주제 선정 조건

1. MVC 모델로 구현할 수 있는가 ?
2. DB를 효율적으로 관리할 수 있는가 ?
3. 보기 좋은 디자인인가 ?

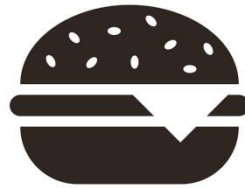
01

02

03

04

05

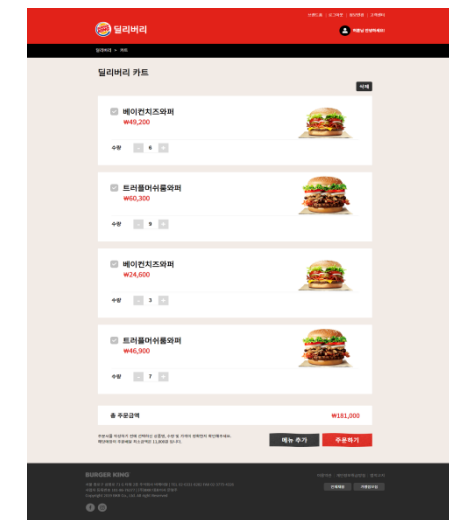
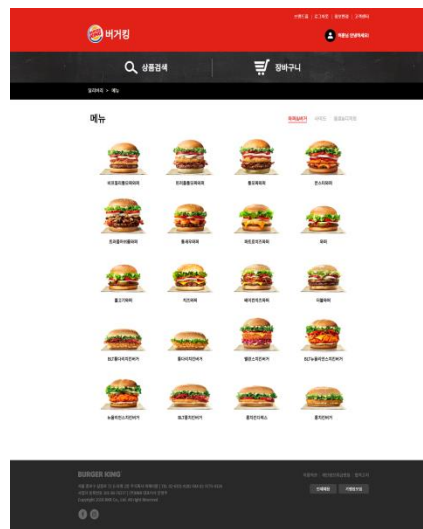
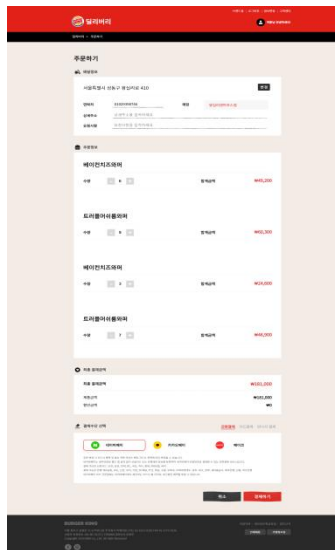
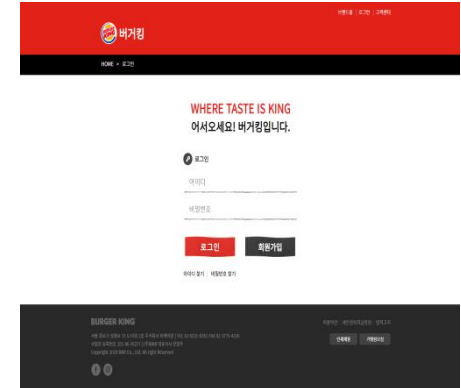
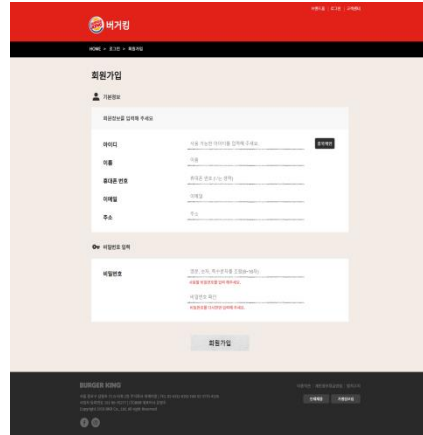
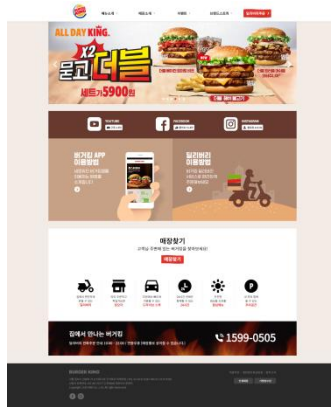
**BURGER KING**

➤➤ 위 조건에 맞는 사이트를 벤치마킹 !

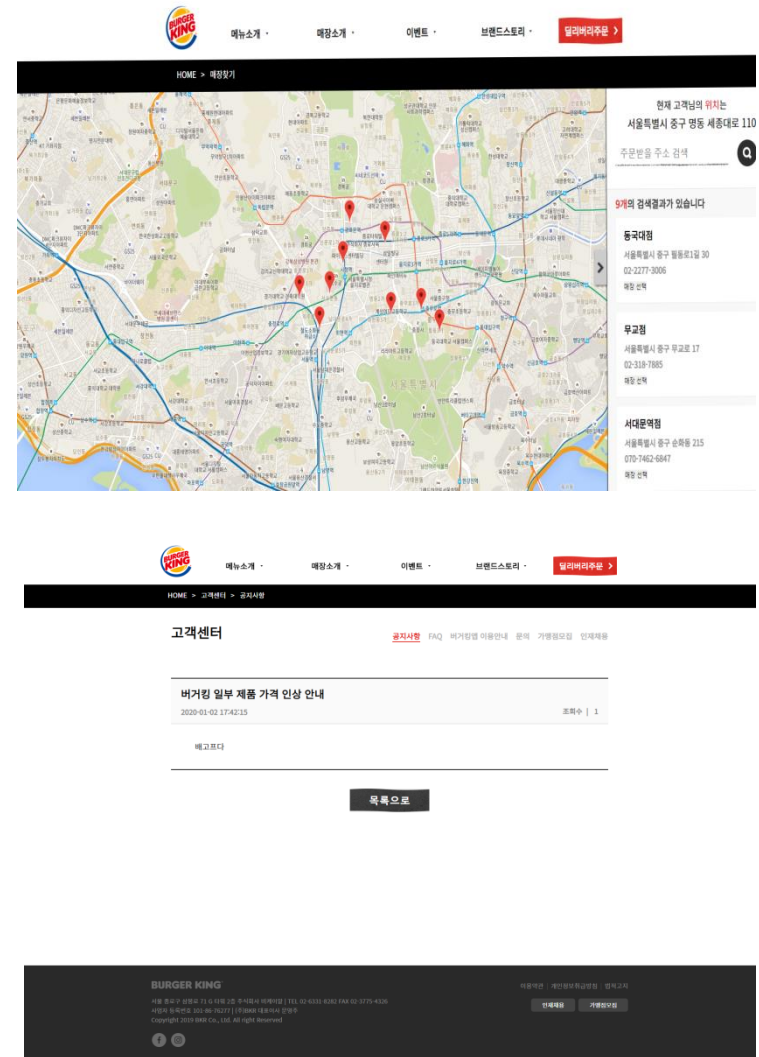
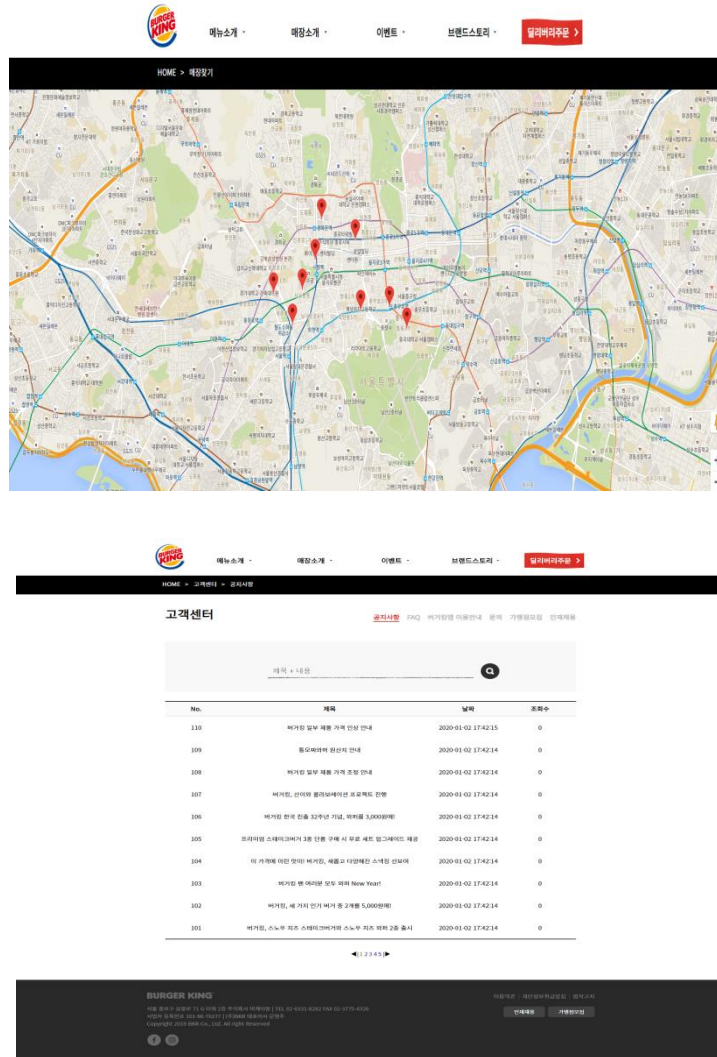
# 구성

## Configuration

# USER INTERFACE

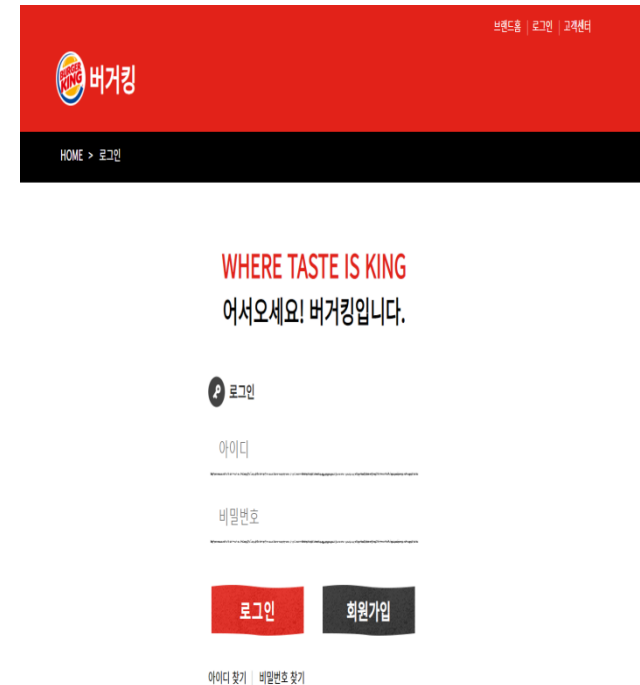
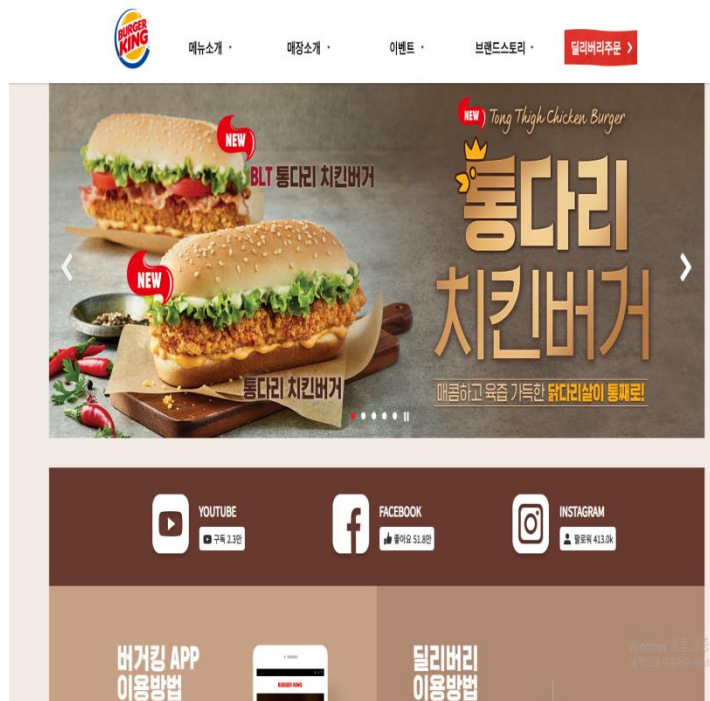


# USER INTERFACE



## 로그인 & 회원가입

```
UserVO vo = user_dao.select(id);  
HttpSession session = request.getSession();  
session.setAttribute("user_data", vo);
```

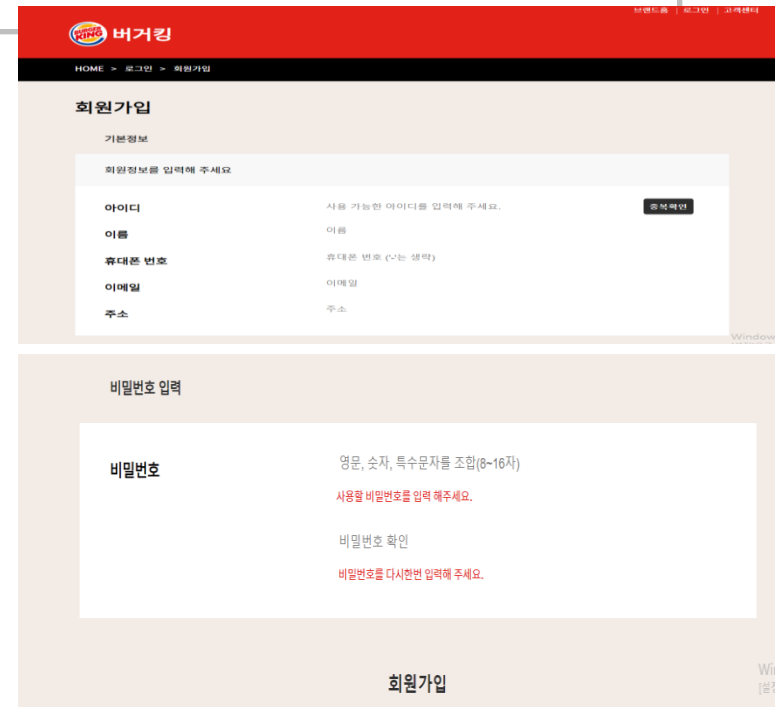




## 01 &gt;&gt; 로그인 &amp; 회원가입

```
int res=user_dao.insert(vo);
```

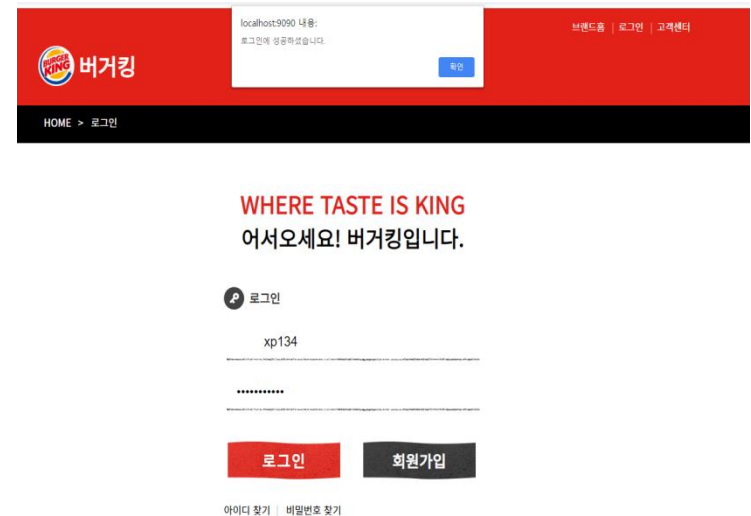
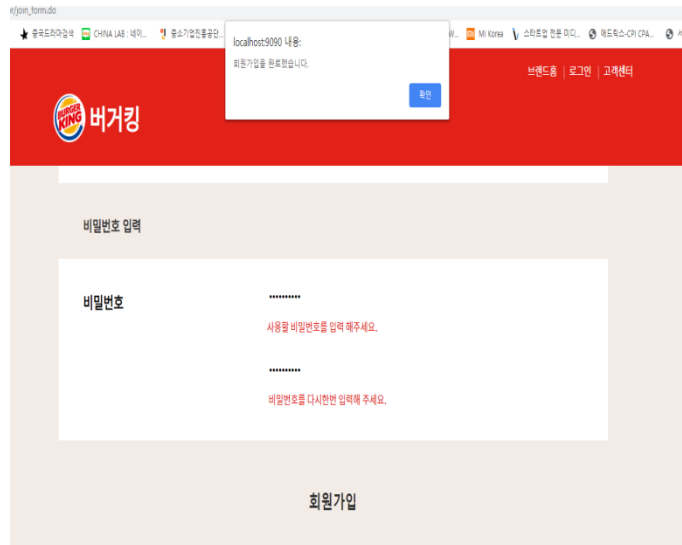
```
String result = "no";  
if( res != 0 ) {  
    result = "yes";  
}
```



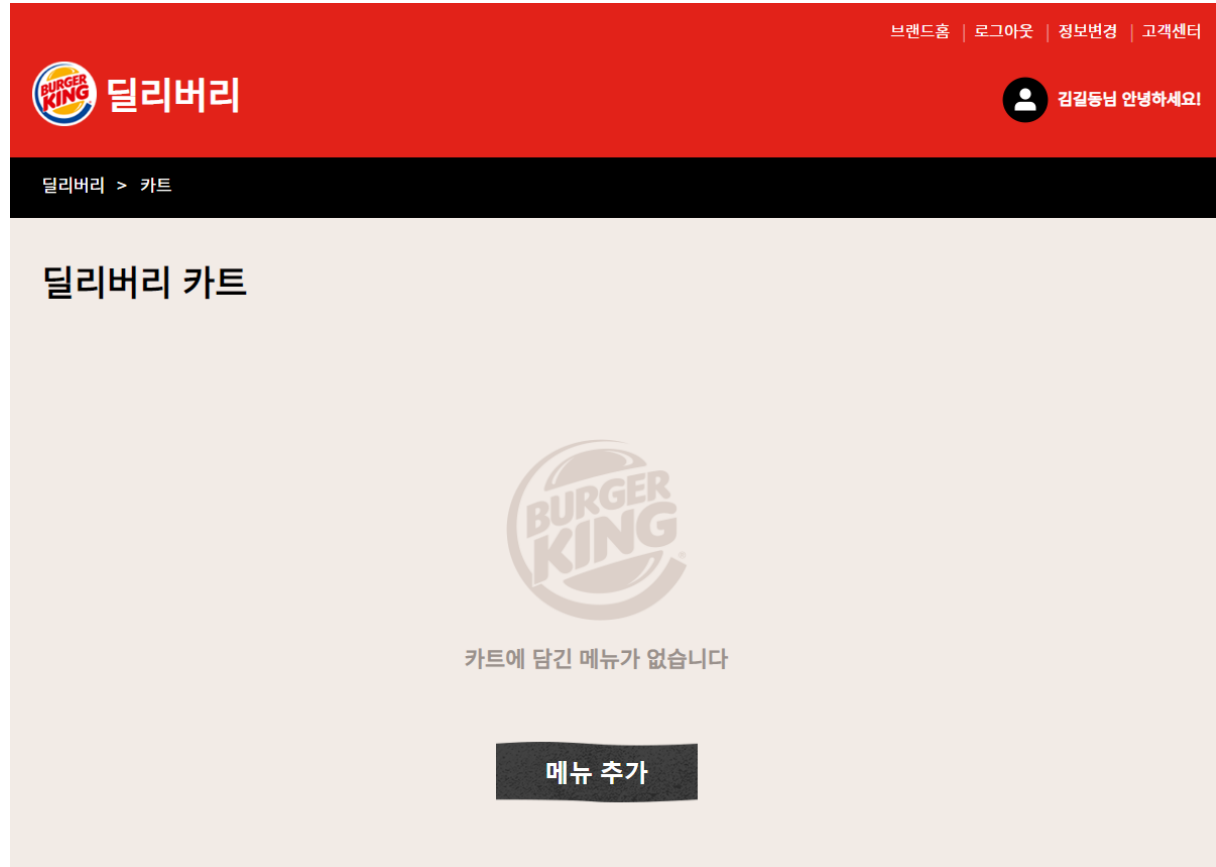
## 로그인 & 회원가입

```
String s_pwd=user_dao.logincheck(id);
```

```
String result="no";  
if(s_pwd.equals(pwd)) {  
result="yes";  
}
```



## »» 카트에 상품이 없을 시 화면



## 04 >> 카트에 상품을 추가



### 비프칠리통모짜와퍼

풍미 터지는 비프칠리와 통모짜렐라의 만남!

₩9200

뒤로가기

추가하기

상품을 장바구니에 추가했습니다.

확인



## ≫≫ 카트에 상품이 추가된 화면

```
<insert id= "cart_insert" parameterType="java.util.HashMap">  
    insert into food_cart values(seq_food_cart_idx.nextVal, 1, #{f_idx}, null, #{u_idx})  
</insert>
```

EDIT	C_IDX	C_COUNT	F_IDX	D_SIZE	U_IDX
	1	2	11	-	1
	2	1	5	-	1
	3	2	11	-	1
	4	1	5	-	1
	21	1	1	-	5
row(s) 1 - 5 of 5					

### 딜리버리 카트

삭제

☒ 비프칠리통모짜와퍼  
₩9,200



수량  1

총 주문금액

₩9,200

주문서를 작성하기 전에 선택하신 상품명, 수량 및 가격이 정확한지 확인해주세요.  
해당매장의 주문배달 최소금액은 11,000원 입니다.

메뉴 추가

주문하기

## 구글 지도 API 사용

The screenshot shows the Google Cloud Platform console interface. The left sidebar contains navigation links: 대시보드, 라이브러리, 사용자 인증 정보 (selected), OAuth 동의 화면, 도메인 확인, and 페이지 사용 동의. The main content area is titled 'API 키 제한 및 이름 변경' and includes sections for '키 제한사항' (Key Restrictions), '애플리케이션 제한사항' (Application Restrictions), and '웹사이트 제한사항' (Website Restrictions). The 'API Key' section displays the key 'AIzaSyA1Kpd9yo1q4m15XuCWj3XyQxzQF6pIXGo' and its creation details. The '웹사이트 제한사항' section shows a list of restricted domains, including 'localhost:9090/burger/map.do' and 'localhost:9090/burger/map\_get\_coords.do'.

Google Cloud Platform 에 접속하여  
구글 지도 API 발급

## »» 구글 지도 API 사용

```
<script async defer src="https://maps.googleapis.com/maps/api/js?key=AIzaSyALKpd9yo1q4mi5XuCWj3XyQxzQFGpIXGo  
&callback=initialize&sensor=false"></script>  
  
<script>  
  function initialize(){  
    let geocoder = new google.maps.Geocoder(); // 지도를 생성하기 위해 필요한 geocoder 객체를 생성  
    let address = "<c:out value='${address}' />"; // 사용자로부터 입력받은 주소  
  
    geocoder.geocode({'address': address}, function(results, status){  
      if (status === google.maps.GeocoderStatus.OK){  
        let location = "<c:out value='${lat+1}' />"; // 이력받은 주소이 이력
```

- script 영역에 발급 받은 지도 API를 링크
- 해당 파일이 실행될 때, initialize() 함수가 가장 먼저 호출됨
- initialize() 함수 안에서만 geocoder 객체를 생성 가능
- geocoder는 지도를 생성하고, 위도와 경도를 계산할 수 있는 객체

## ≫≫ 지도를 생성하는 방법 1

```
<form name="form">
  <c:if test="${!empty sessionScope.user_data}">
    <input type="hidden" name="address" value="${sessionScope.user_data.getU_addr()}">
  </c:if>
  <c:if test="${empty sessionScope.user_data}">
    <input type="hidden" name="address" value="서울특별시 중구 명동 세종대로 110">
  </c:if>
</form>
```

- 사용자가 주문을 받을 주소를 map\_get\_coords.jsp로 전송
- 로그인이 되어 있지 않은 경우, 서울시청(중구 명동 세종대로 110) 주소를 전송
- 로그인이 되어 있는 경우, 사용자가 가입 시 등록한 주소(u\_addr)를 전송



## 05 >> 지도를 생성하는 방법 2

```
<script async defer src="https://maps.googleapis.com/maps/api/js?key=AIzaSyALKpd9yo1q4mi5XuCWj3XyQxzQFGpIXGo
&callback=initialize&sensor=false"></script>
<script>
  function initialize(){
    let address = "<c:out value='${address}'/>";
    let geocoder = new google.maps.Geocoder();

    geocoder.geocode({'address': address}, function(results, status){
      if (status === google.maps.GeocoderStatus.OK){
        let lat = results[0].geometry.location.lat(); // 입력받은 주소의 위도
        let lng = results[0].geometry.location.lng(); // 입력받은 주소의 경도

        let form = document.createElement("form");
        document.body.appendChild(form);

        let path = "map.do";
        let params = {"address": address, "lat": lat, "lng": lng};
        let method = "POST";

        form.setAttribute("action", path);
        form.setAttribute("method", method);

        for (let key in params){
          var hiddenField = document.createElement("input");
          hiddenField.setAttribute("type", "hidden");
          hiddenField.setAttribute("name", key);
          hiddenField.setAttribute("value", params[key]);

          form.appendChild(hiddenField);
        }

        form.submit();
      }
    });
  }
}
```

- 주소, 위도, 경도를 담은 JSON 타입 데이터를 map.do로 전송
- 단순히 매개 변수로 받은 주소의 위도와 경도를 계산하는 임시 파일
- 스크립트 영역에서 <form>을 강제로 생성

## ≫≫ 지도를 생성하는 방법 3

```
// 지도 생성 및 전체 매장 목록
@RequestMapping("/map.do")
public String map(Model model, String address, double lat, double lng) {
    List<MapVO> storeList = map_dao.selectList(); // 버거킹의 모든 매장
    List<MapVO> nearStoreList = new ArrayList<MapVO>(); // 입력 받은 주소에서 주문 가능한 모든 매장 (2km 내에 존재하는 매장)

    for (int i = 0; i < storeList.size(); i++) {
        if (MapUtil.distance(lat, lng, storeList.get(i).getL_lat(), storeList.get(i).getL_lng(),
            "kilometer") <= 2.0) {
            MapVO vo = map_dao.selectOne(storeList.get(i).getL_name());
            nearStoreList.add(vo);
        }
    }
}
```

- MapUtil 클래스의 distance 메서드를 활용하여 거리 계산
- 사용자의 주소에서 2km 내에 존재하는 모든 매장을 nearStoreList에 저장
- 주소, 위도, 경도, 주문 가능 매장 데이터를 map.jsp로 전송 (addAttribute)

## » 지도를 생성하는 방법 4

```
function initialize(){
    let geocoder = new google.maps.Geocoder(); // 지도를 생성하기 위해 필요한 geocoder 객체를 생성
    let address = "<:out value='${address}' />"; // 사용자로부터 입력받은 주소

    geocoder.geocode({'address': address}, function(results, status){
        if (status === google.maps.GeocoderStatus.OK){
            let userLat = "<:out value='${lat}' />"; // 입력받은 주소의 위도
            let userLng = "<:out value='${lng}' />"; // 입력받은 주소의 경도

            let latLng = new google.maps.LatLng(userLat, userLng);
            let mapOption = {zoom: 14, center: latLng};
            let map = new google.maps.Map(document.getElementById("map"), mapOption); // 구글 맵을 생성
            let marker = new google.maps.Marker(null);

            "<:forEach var='vo' items='${nearStoreList}'>"
                marker = new google.maps.Marker({
                    position: new google.maps.LatLng('${vo.getL_lat()}', '${vo.getL_lng()}'),
                    title: '${vo.getL_name()}',
                    map: map});
                marker.setMap(map);
            "</c:forEach>"
        }
    });
}
```

- 주소, 위도, 경도 데이터를 활용하여, 지도 생성
- 주문 가능한 매장 데이터를 활용하여, 지도 위 마커 생성

## 지도를 생성하는 방법 5

HOME > 매장찾기

현재 고객님의 위치는  
서울특별시 중구 명동 세종대로 110

주문받을 주소 검색

9개의 검색결과가 있습니다

**동국대점**  
서울특별시 중구 필동로1길 30  
02-2277-3006  
매장 선택

**무교점**  
서울특별시 중구 무교로 17  
02-318-7885  
매장 선택

**서대문역점**  
서울특별시 중구 순화동 215  
070-7462-6847  
매장 선택

주소를 직접 입력했을 때도, 똑같은 과정을 반복!

## ≫≫ 매장을 선택하는 방법 1

```
function select_store(l_name, u_idx){  
    if (u_idx == 0){  
        alert("로그인을 하셔야 이용하실 수 있는 서비스 입니다.");  
        if (!confirm("로그인 창으로 이동하시겠습니까?")){  
            return;  
        } else{  
            location.href = "login_form.do";  
        }  
        return;  
    } else{  
        let address = "<c:out value='${address}' />";  
  
        let url = "map_store_select.do";  
        let param = "address=" + address + "&l_name=" + l_name + "&u_idx=" + u_idx;  
  
        sendRequest(url, param, resultFn, "POST");  
    }  
}
```

```
public class UserVO {  
    int u_idx; // 회원 번호  
    String u_id; // 회원 아이디  
    String u_pwd; // 회원 비밀번호  
    String u_tel; // 회원 연락처  
    String u_addr; // 회원 주소  
    String u_name; // 회원 이름  
    String u_email; // 회원 이메일  
    String u_order_addr; // 주문 받을 주소 (매 주문마다 계속 업데이트되는 정보)  
    String u_order_store; // 주문할 매장 (매 주문마다 계속 업데이트되는 정보)
```

- 매장 선택은 로그인이 되어 있는 경우에만 가능
- 주문할 매장의 이름과 주문 받을 주소를 map\_store\_select.do로 전송
- UserVO의 u\_order\_addr과 u\_order\_store가 업데이트됨

## ≫≫ 매장을 선택하는 방법 2

```
// 장바구니에 있는 상품 결제
@RequestMapping("/cart_payment.do")
public String cart_payment(Model model, HttpServletRequest request, String u_id) {
    // 세션에 있던 기존 정보를 삭제 (매장이 변경되었을 수도 있기 때문)
    HttpSession session = request.getSession();
    session.removeAttribute("user_data");

    // 세션에 사용자 정보를 다시 등록
    UserVO vo = cart_dao.updateSession(u_id);
    session.setAttribute("user_data", vo);

    UserVO user_data = (UserVO)session.getAttribute("user_data");

    List<CartVO> cartList = cart_dao.selectList(user_data.getU_idx());
    int totalAmount = cart_dao.getTotalAmount(user_data.getU_idx());

    model.addAttribute("cartList", cartList);
    model.addAttribute("totalAmount", totalAmount);

    return Common.Cart.VIEW_PATH + "cart_payment.jsp";
}
```

매장 선택을 통해 UserVO의 일부가 변경되었기 때문에

**세션에 사용자 정보를 재등록!**

## 05 >> 매장을 선택하는 방법 3

### 주문하기

#### 배달정보

서울특별시 성동구 왕십리로 410

변경

연락처

01029398746

매장

왕십리센트라스점

상세주소

상세주소를 입력하세요

요청사항

요청사항을 입력하세요

최종 결제 확인 창에,

주문할 매장과 주문 받을 주소가 성공적으로 등록된 것을 확인!



**시연**  
Testing