

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 ?

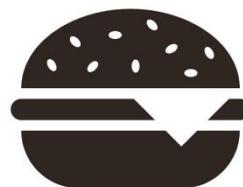
01

02

03

04

05



BURGER KING

»» 위 조건에 맞는 사이트를 벤치마킹 !

구성

Configuration

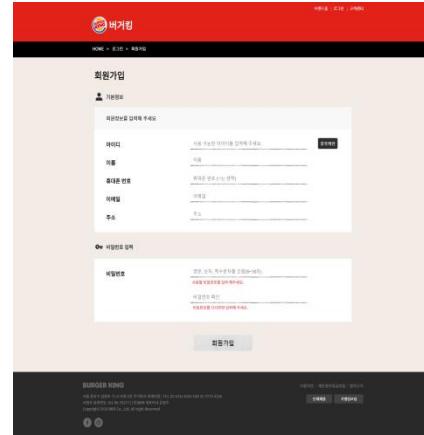
01

>> USER INTERFACE

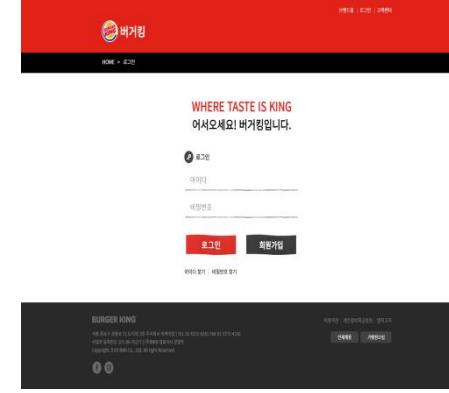
02



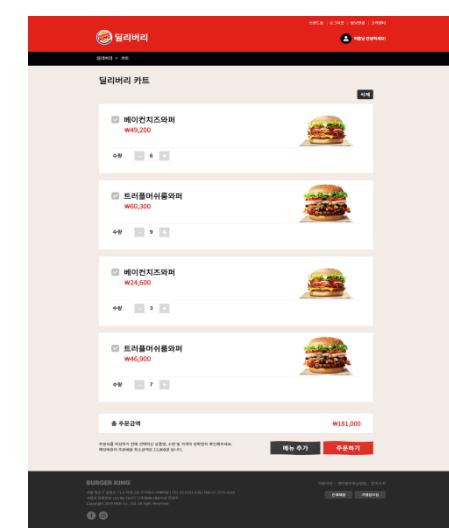
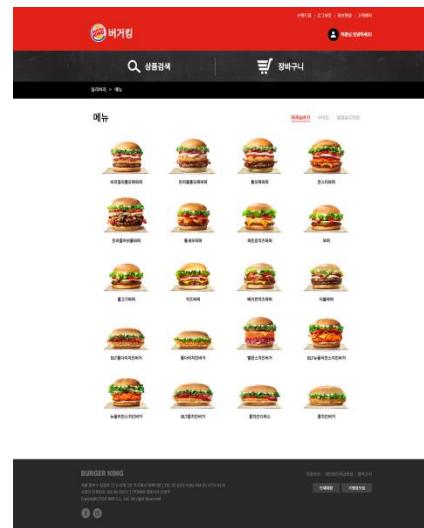
03



04



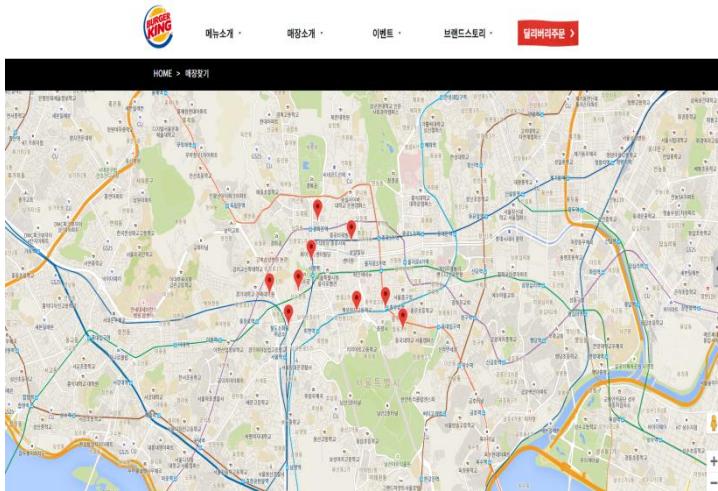
05



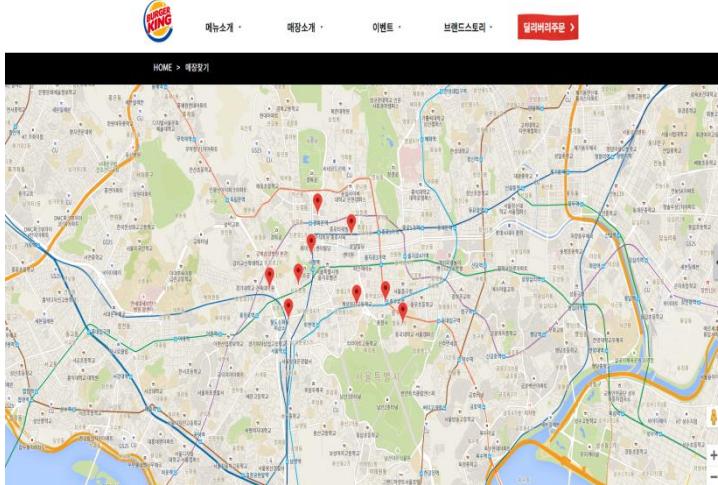
01

>> USER INTERFACE

02



03



04

고객센터

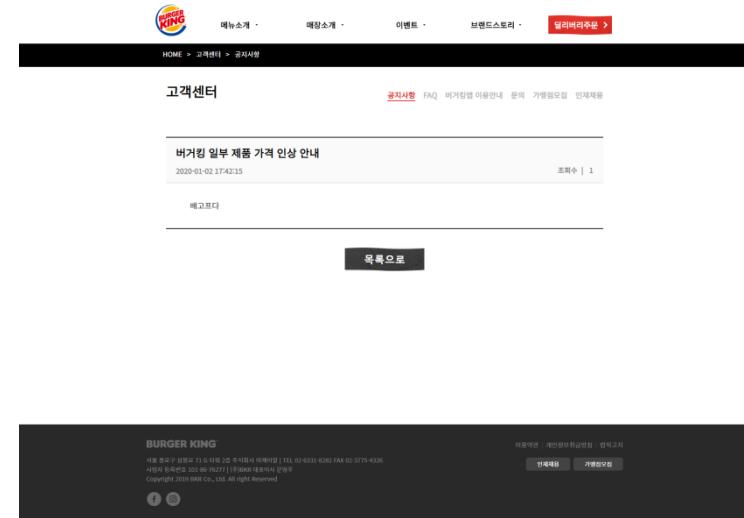
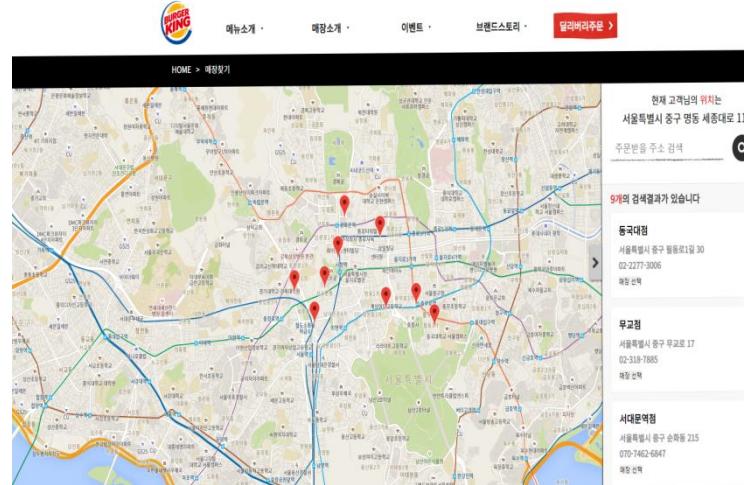
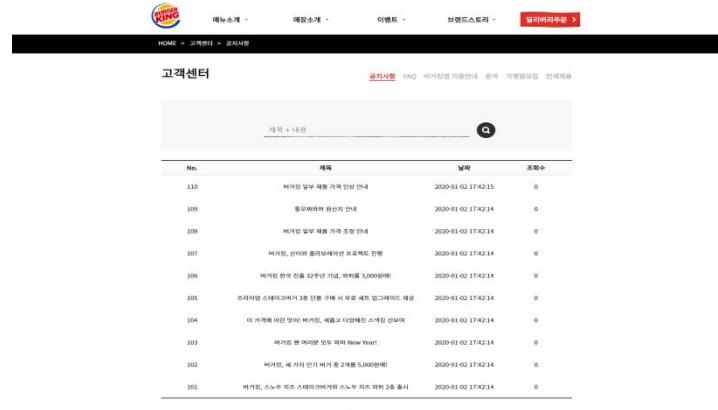
[고객센터](#) FAQ 버거킹앱 이용안내 문의 가맹점문점 인재채용

No.	제목	날짜	조회수
110	버거킹 일부 제품 가격 인상 안내	2020-01-02 17:42:15	0
109	점포별로 판매지 연락	2020-01-02 17:42:14	0
108	버거킹 일부 제품 가격 조정 안내	2020-01-02 17:42:14	0
107	버거킹, 퀸즈와 플라워파크점 조로젝트 진행	2020-01-02 17:42:14	0
106	버거킹 한국 진출 32주년 기념, 배달비 3,000원(한정)	2020-01-02 17:42:14	0
105	프리미엄 스티아이거니 3종 단품 구매 시 무료 입장권 제공	2020-01-02 17:42:14	0
104	이 카페에 대한 멋진 버거킹, 새롭고 다양한 신제품 선보여	2020-01-02 17:42:14	0
103	버거킹 멘 이어리얼 모두 차지 New Year!	2020-01-02 17:42:14	0
102	버거킹, 새 차기 신제품 개발 5,000만원(한정)	2020-01-02 17:42:14	0
101	버거킹, 소노우 카즈 스트리트버거와 스누우 카즈 외회 2종 출시	2020-01-02 17:42:14	0

1 2 3 4 5 >



05



01

>> 로그인 & 회원가입

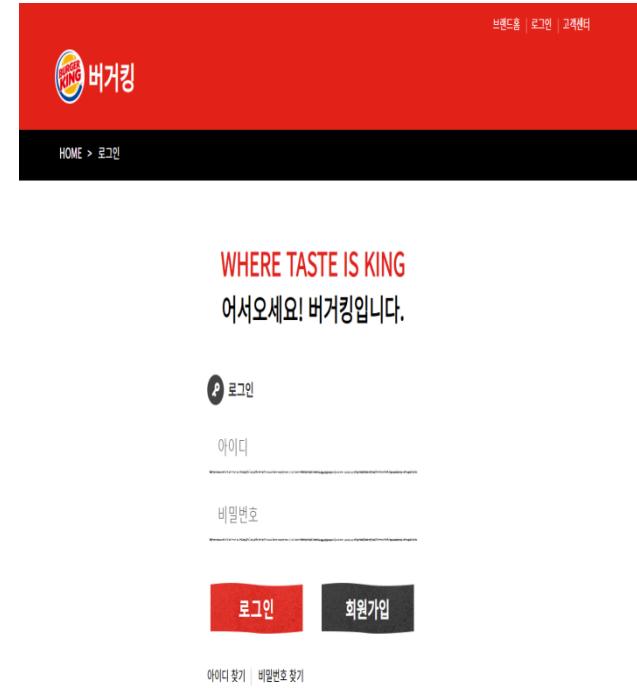
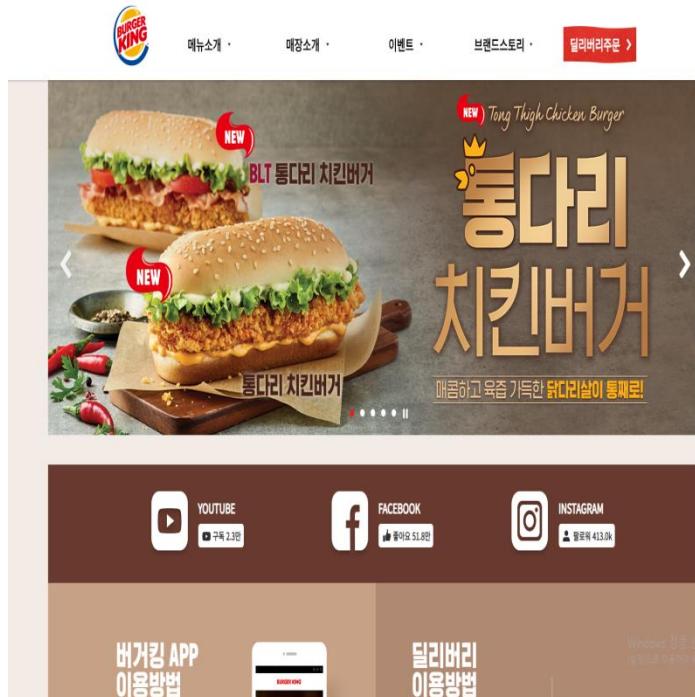
```
UserVO vo = user_dao.select(id);  
HttpSession session = request.getSession();  
session.setAttribute("user_data", vo);
```

02

03

04

05



01

02

03

04

05

»> 로그인 & 회원가입

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

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



The composite screenshot shows the registration process on the Burger King website. It consists of three panels:

- Top Panel:** A summary of the registration steps: HOME > 로그인 > 회원가입.
- Left Panel (Basic Information):** A form titled '회원가입' with '기본정보' (Basic Information). It asks for '회원정보를 입력해 주세요' (Please enter member information). Fields include '아이디' (ID), '이름' (Name), '휴대폰 번호' (Phone Number), '이메일' (Email), and '주소' (Address). Buttons include '중복확인' (Duplicate Check) and 'Windows' (Operating System indicator).
- Right Panel (Password):** A form titled '비밀번호 입력' (Password Input). It asks for '비밀번호' (Password) and '비밀번호 확인' (Password Confirmation). It specifies that the password should be 8~16 characters long and contains letters, numbers, and symbols. Buttons include 'Windows' (Operating System indicator) and '회원가입' (Registration) at the bottom.

01

»> 로그인 & 회원가입

02

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

03

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

04

05

비밀번호 입력

비밀번호

.....

사용할 비밀번호를 입력 해주세요.

.....

비밀번호를 다시한번 입력해 주세요.

회원가입



로그인에 성공하셨습니다.

확인

HOME > 로그인

비밀번호 | 로그인 | 고객센터

WHERE TASTE IS KING
어서오세요! 버거킹입니다.

로그인

xp134

비밀번호

로그인

회원가입

아이디 찾기 | 비밀번호 찾기

01

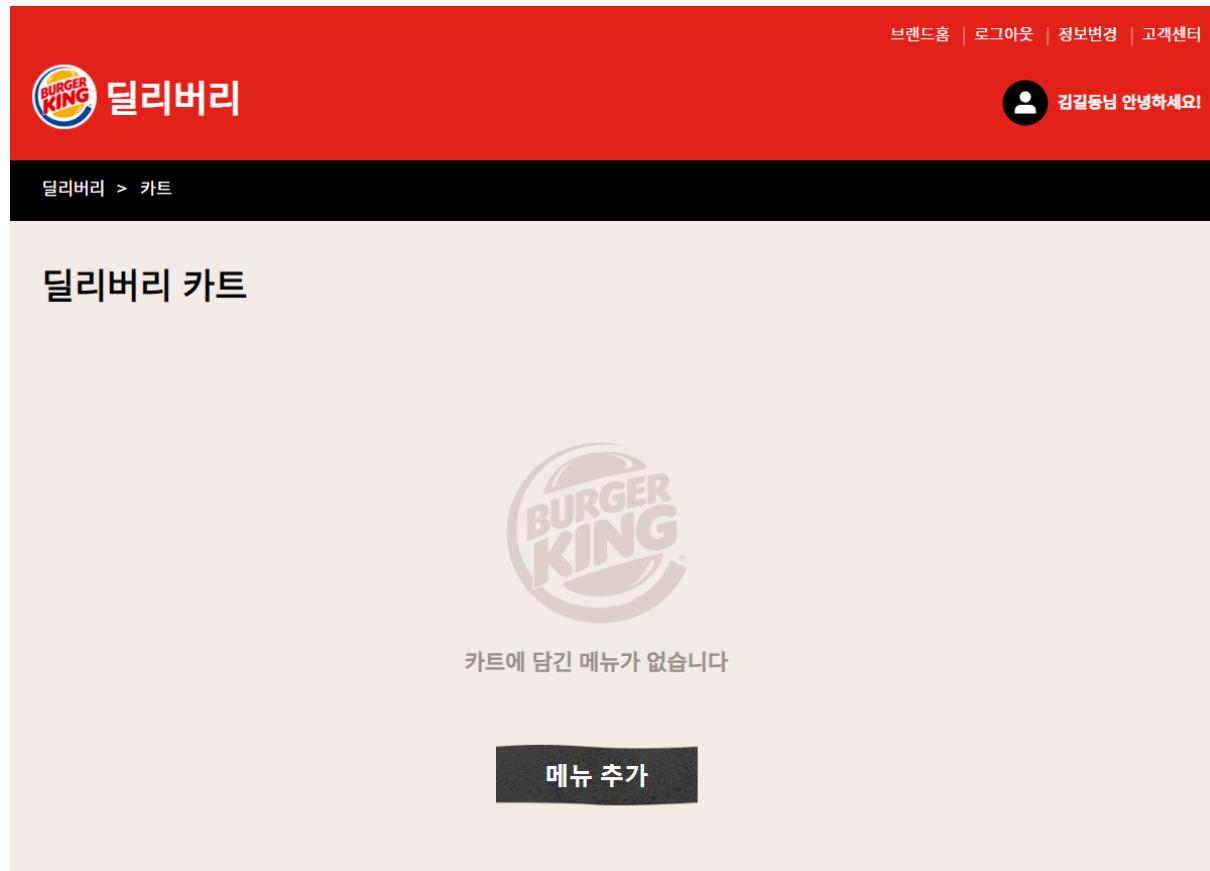
» 카트에 상품이 없을 시 화면

02

03

04

05



01

02

03

04

05

» 카트에 상품을 추가



비프칠리통모짜와퍼

풍미 터지는 비프칠리와 통모짜렐라의 만남!

₩9200

뒤로가기 추가하기

상품을 장바구니에 추가했습니다.

확인

A large, detailed image of a double cheeseburger with various toppings like lettuce, tomato, and cheese. Below the image is a product description in Korean. At the bottom, there are two buttons: '뒤로가기' (Back) and '추가하기' (Add to Cart). A hand cursor is pointing at the '추가하기' button. A callout box to the right of the button says '상품을 장바구니에 추가했습니다.' (Item added to cart) with a '확인' (Confirm) button.

01

» 카트에 상품이 추가된 화면

02

```
<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>
```

03

04

05

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

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 11,000원입니다.

메뉴 추가

주문하기

01
02
03
04
05

» 구글 지도 API 사용

The screenshot shows the Google Cloud Platform API Key Management interface. On the left sidebar, under the 'API 및 서비스' section, 'API 키 제한 및 이름 변경' is selected. In the main area, a new API key has been created with the name 'API key'. The key details are as follows:

- API Key:** AIzaSyA1Kpd9yo1q4m15XuCWj3Xy0xF6pIXGo
- 설명:** key=API_KEY 매개변수로 키를 전달하여 애플리케이션에서 이 키를 사용하세요.
- 생성일:** 2019년 12월 15일 PM 8시 50분 37초 GMT+9
- 생성자:** doubleh819@gmail.com (-)
- 총 사용량(지난 30일):** 2,225

Below the key details, there is a section titled 'API 키를 특정 웹사이트로 제한하려면 어떻게 해야 하나요?' (How do I restrict an API key to a specific website?). It provides instructions for using a proxy to restrict the API key to a specific URL. It also lists several examples of URLs that can be restricted:

- 정확한 경로를 사용한 구체적인 URL: www.example.com/path
- 와일드 카드 범프(*)를 사용한 단일 하위 도메인의 모든 URL: sub.example.com/*
- 와일드 카드 범프(*)를 사용한 단일 도메인의 하위 도메인 또는 경로 URL: *example.com/*
- 비표준 포트가 포함된 URL: www.example.com:8000/*

A note at the bottom states: '참고: 워리 매개변수 및 조작은 현재 지원되지 않으므로 HTTP 리퍼리에 포함하면 무시됩니다.' (Note: Query parameters and manipulation are currently not supported, so they will be ignored if included in the HTTP referer).

At the bottom of the page, there is a warning message: '비워 두면 API 키가 모든 웹사이트의 요청을 수락합니다.' (If left empty, the API key will accept requests from all websites).

Below the warning, there are two dropdown menus showing example URLs:

- localhost:9090/burger/map.do
- localhost:9090/burger/map_get_coords.do

There is also a '항목 추가' (Add item) button.

Google Cloud Platform 에 접속하여
구글 지도 API 발급

01

» 구글 지도 API 사용

```
02
03 <script async defer src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAlKpd9yo1q4mi5XuCWj3XyQxzQFGpIXGo
04 &callback=initialize&sensor=false"></script>
05
<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 userLat = "<c:out value='${results[0].geometry.location.lat}'/>"; // 위도
                let userLon = "<c:out value='${results[0].geometry.location.lng}'/>"; // 경도
            }
        })
    }
</script>
```

- 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

01 »» 지도를 생성하는 방법 1

```
02  
03 <form name="form">  
04   <c:if test="${!empty sessionScope.user_data}">  
05     <input type="hidden" name="address" value="${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

01

» 지도를 생성하는 방법 2

02

03

04

05

```
<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();
            }
        });
    }
</script>
```

- 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

01 » 지도를 생성하는 방법 3

```
02  
03 // 지도 생성 및 전체 매장 목록  
04 @RequestMapping("/map.do")  
05 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)

01 » 지도를 생성하는 방법 4

```
02
03
04
05
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 userLat = "<c:out value='${lat}'/>"; // 입력받은 주소의 위도
            let userLng = "<c:out value='${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);

            "<c: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

01
02
03
04
05

» 지도를 생성하는 방법 5

HOME > 매장찾기

현재 고객님의 위치는
서울특별시 종구 명동 세종대로 110

주문받을 주소 검색

9개의 검색결과가 있습니다

동국대점
서울특별시 종구 펠동로1길 30
02-2277-3006
매장 선택

무교점
서울특별시 종구 무교로 17
02-318-7885
매장 선택

서대문역점
서울특별시 종구 순화동 215
070-7462-6847
매장 선택

주소를 직접 입력했을 때도, 똑같은 과정을 반복!

01
02
03
04
05

» 매장을 선택하는 방법 1

```

function select_store(l_name, u_idx){
    if (u_idx == 0){
        alert("로그인을 하셔야 이용하실 수 있는 서비스입니다.");
        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

01 » 매장을 선택하는 방법 2

```
02                          // 장바구니에 있는 상품 결제  
03                          @RequestMapping("/cart_payment.do")  
04                          public String cart_payment(Model model, HttpServletRequest request, String u_id) {  
05                          // 세션에 있던 기존 정보를 삭제 (매장이 변경되었을 수도 있기 때문)  
06                          HttpSession session = request.getSession();  
07                          session.removeAttribute("user_data");  
08                          // 세션에 사용자 정보를 다시 등록  
09                          UserVO vo = cart_dao.updateSession(u_id);  
10                          session.setAttribute("user_data", vo);  
11                          UserVO user_data = (UserVO)session.getAttribute("user_data");  
12                          List<CartVO> cartList = cart_dao.selectList(user_data.getU_idx());  
13                          int totalAmount = cart_dao.getTotalAmount(user_data.getU_idx());  
14                          model.addAttribute("cartList", cartList);  
15                          model.addAttribute("totalAmount", totalAmount);  
16                          return Common.Cart.VIEW_PATH + "cart_payment.jsp";  
17              }
```

매장 선택을 통해 UserVO의 일부가 변경되었기 때문에

세션에 사용자 정보를 재등록!

01 »» 매장을 선택하는 방법 3

02

03

04

05

주문하기

배달정보

서울특별시 성동구 왕십리로 410

변경

연락처

01029398746

매장

왕십리센트라스점

상세주소

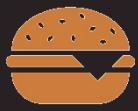
상세주소를 입력하세요

요청사항

요청사항을 입력하세요

최종 결제 확인 창에,

주문할 매장과 주문 받을 주소가 성공적으로 등록된 것을 확인!



시연
Testing