Project (프로젝트)

Project Info

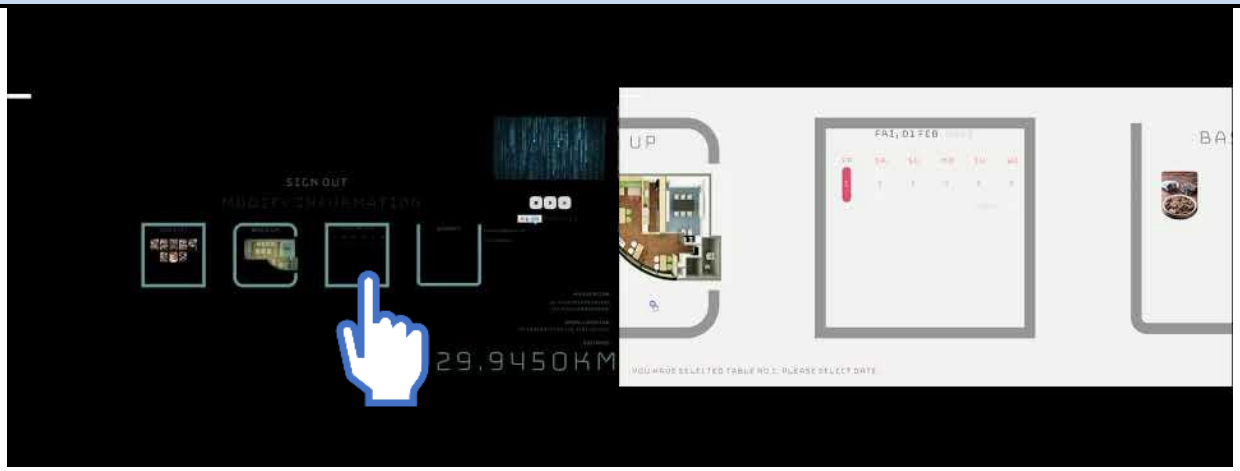
프로젝트 명	PREPARED DINING	
개발 기간	2018.11.25 ~ 2019.01.10	
개발 인원	1 명	
담 당 자	이 름	이현석
	연 락 처	010-4239-2275
	전 자 우 편	exorsa1525@gmail.com
담당업무	제안, 기획, Model 설계, Database 설계, 모듈의 기능구현, 디자인,	

Use Technology

개발 환경	Windows x64, JVM, WAS, DBMS
개발 언어	JAVA, HTML5, CSS3, Servlet/JSP, EL/JSTL, JavaScript, JQuery, BootStrap, SQL
구현 도구	Eclipse, Oracle 11g SQL Developer, AQuery tool, Draw.io, UML
구현 기술	JSP MVC Model2, AJAX, 위치정보기반 보안기술

Introduction

음식점 예약관리 서비스 "PREPARED DINING" Review



웹 브라우저상에서 예약주문 및 결제

위치기반 정보를 활용하여 효율적인 예약관리 및 간편한 인증 기능 구현

무인 결제로 인건비 절감과 매출 증대 효과

Develop Detail

제 안

A 기술분야

- 원격지에서 사용자의 위치정보를 활용하여 식당의 메뉴와 테이블을 시간대별로 예약하기 위한 목적으로 고안된 음식점 예약관리 웹 서비스이다

B 프로젝트의 배경

- 통계청의 서비스업 동향 조사결과에 따르면 2019년 음식점 소매판매액 지수가 작년대비 0.2% 감소하여 2년연속 하락했다. 인건비가 상승하고 재택근무 온라인쇼핑 등이 늘면서 외식 문화의 변화로 인해 소비가 줄어들었기 때문이다. 주 52시간 근무제 시행 이후 PC오프제와 집중근무제가 보편화 되면서 이전보다 업무시간이 줄어 들어 자체적으로 점심시간을 단축하고 업무를 보기 위해 사무실에서 간단하게 한 끼를 때우는 일명 '데스크톱 다이닝' 문화도 떠오르고 있다.
-

C 관련 기술

- 무인 결제 KIOSK는 소비자 스스로 메뉴를 검색하고 주문, 결제까지 가능한 터치스크린 방식의 무인 단말기이다. 전문가들은 소비자들이 KIOSK 사용을 선호하고 매출 증대효과를 볼 수 있다고 분석했다. 2018년 미국 성인 1000명을 대상으로 설문조사 결과 응답자 55%가 줄을 서지 않아도 된다는 이유로 KIOSK를 선호했다. 12%는 직원과 대면하지 않아도 된다는 이유로 KIOSK를 이용한다고 답했다. 이런 상황에서 인건비를 절감하고 인력을 효율적으로 활용하기 위해 국내에도 빠르게 도입되고 있다.

D 기술적 과제

- 종래의 KIOSK나 배달 어플리케이션과 달리 원격지에서 실시간 위치기반 예약관리 제어 방식 기획
- 고객은 웹 상에서 미리 식당메뉴를 예약하고 테이블에 앉자 마자 바로 식사를 할 수 있도록 하여 고객이 기다리는 시간을 최대한 단축시킨다.
- IP주소와 위성항법장치로부터 수신한 고객의 위치정보를 활용하여 해당 목적지까지의 거리를 측정 및 분석하여 효율적인 예약관리가 가능하도록 한다.
- 결제 단계에서 비밀번호를 입력하는 대신 수집한 위치기반 정보를 수집 및 검토하여 인증하는 간편한 보안 인증기술을 구현한다.

E 프로젝트의 비전

- 특허 및 신규사업 영역 진출
- 직장인들의 아침과 점심식사를 겨냥한 영업이익 창출
- 웹 브라우저상에서 예약관리가 필요한 가맹점에 배포
- 서비스 개발을 통한 개인역량 향상

기 획

A 개발전략

- MVC Framework 패턴 Model 2 방식으로 안정성과 유지보수
- 싱글 톤 패턴으로 전역 인스턴스를 생성하여 자원낭비를 줄이고 어디서나 접근
- Ajax를 활용하여 비동기적 환경에서 서버와 통신 및 트래픽 낭비 감소
- 공통 모듈 구현
  - 가독성, 재사용성, 유지보수성
  - 모듈간 결합도는 줄이고 각 모듈의 내부 응집도 향상
  - 보안, 인터페이스, 유효성 검사 스크립트, 로그인, 전자결제, 거리계산기

B 개발환경 구축

- JDK설치, 환경변수 설정, Eclipse 설치, Apache/Tomcat 설치, Eclipse에 Tomcat 연동
- SCM 구성
  - 관리도구: Git / Git Hub
  - 변경사항 관리 및 추적, 문제 발생요인 최소화

C 서버 사이드

- JSP에서 데이터 요청 시 JavaScript에서 데이터 유효성 검사 후 Submit
- 모든 요청은 Front Controller(Servlet)에서 Servlet-mapping하여 해당 Service객체에 작업 위임(RequestDispatcher로 포워딩)
- Service는 DAO를 호출하여 JDBC를 통해 해당 DB에 접근하여 결과 값을 리턴
- JSP페이지에서 결과값을 출력

D 클라이언트 사이드

- 직관적인 UX/UI 구성
  - Zoomooz.js 플러그인을 적용한 PREZI와 유사한 사용자 시점의 확대/축소
  - BootStrap 그리드시스템을 적용한 반응형 웹으로 디자인 시간 단축)
  - Modal 창을 활용하여 화면전환 없이 메뉴 컨텐츠 및 결제페이지 구성
- JavaScript/JQuery 언어를 사용하여 DOM 탐색 및 수정, 이벤트제어
- HTML5/CSS3 스타일 구성 및 애니메이션 효과 적용

E 테스트

- 테스트 도구: Junit
- 테스트 범위: Tomcat위에서 회원, 메뉴, 예약, 결제, 보안 등 Module 및 method 동작
- 테스트 케이스 작성
- 단위테스트 수행 및 검증
  - 각 기능의 요구사항을 분석하고 적정한 검증구문 작성
  - 단위 별로 테스트 수행 후 디버깅을 통해 결함 보완
  - 로직 추가 및 변경 시 지속적으로 CI 수행

F 화면 구현

- 사용자의 접근성을 고려하여 직관적으로 설계한다.
- 누락되는 것이 없이 일관성과 가독성을 고려하고 사용자 시점에서 우선순위를 정한다.
- 변경 시 수정 용이성과 추적 용이성을 고려하여 설계한다.

설 계

Database

Designing ERD with Aquery Tool

URL: http://aquerytool.com:80/aquerymain/index/?rurl=c81b5858-8ce3-4185-b683-310d78ff777f

Password: k227jc

오픈다이닝(ORACLE)

bananabank

은행

PK	AI	FK	Null	Logical Name	Name	Type
				결제 날짜	payment_date	TIMESTAMP
				고객 계정	customer_account	VARCHAR2(16)
✔				신용 카드번호	credit_cdn	VARCHAR2(20)
				신용 유효기간	credit_inval	VARCHAR2(5)
				신용 cvc	credit_cvc	VARCHAR2(3)
				고객 이름	customer_name	VARCHAR2(26)
				고객 이메일	customer_email	VARCHAR2(26)
				은행 거래금액	bank_withdraw	VARCHAR2(20)
				은행 잔고	bank_balance	VARCHAR2(30)

PG_CCP

PG사_신용카드_거래내역

PK	AI	FK	Null	Logical Name	Name	Type
				등록 PG사 날짜	reg_pg_date	TIMESTAMP
				PG사 신용카드회사	pg_ccc	VARCHAR2(20)
		✔		PG사 결제자 카드번호	pg_payer_cdn	VARCHAR2(20)
				PG사 결제자 이름	pg_payer_name	VARCHAR2(26)
		✔		PG사 결제자 이메일	pg_payer_email	VARCHAR2(26)
				신용카드 cvc	ccd_cvc	VARCHAR2(3)
				신용 유효기간	credit_inval	VARCHAR2(5)
				PG사 결제자 전화번호	pg_payer_phone	VARCHAR2(16)
				신용카드 할부	ccd_installment	VARCHAR2(10)
				PG사 거리	pg_distance	VARCHAR2(10)
				PG사 거래금액	pg_withdraw	VARCHAR2(20)

customer

회원

PK	AI	FK	Null	Logical Name	Name	Type
	✔		✔	고객 번호	customer_index	NUMBER
				성	first_name	VARCHAR2(10)
				이름	last_name	VARCHAR2(16)
✔				고객 이메일	customer_email	VARCHAR2(26)
				고객 암호	customer_pwd	VARCHAR2(16)
			✔	고객 전화번호	customer_phone	VARCHAR2(16)
			✔	고객 주소	customer_addr	VARCHAR2(40)
				등록 가입일자	reg_joinTime	TIMESTAMP
			✔	고객 거리 정보	customer_distance_inf	VARCHAR(10)

bookup

예약

PK	AI	FK	Null	Logical Name	Name	Type
✔				예약(약자) 순서	b_seq	TIMESTAMP
				테이블 번호	t_index	VARCHAR2(12)
		✔		고객 이메일	customer_email	VARCHAR2(26)
		✔		모든상품 이름	products_name	VARCHAR2(128)
				거래금액	withdraw	VARCHAR2(20)
				고객 거리 정보	customer_distance_inf	VARCHAR(10)
				예약된 연월일	reserved_wmy	VARCHAR2(20)
		✔		예약된 일자	reserved_day	VARCHAR2(10)
		✔		예약된 시간	reserved_time	VARCHAR210)

product

상품

PK	AI	FK	Null	Logical Name	Name	Type
	✔			상품(약자) 순서	p_seq	NUMBER
				파일 상품(약자) 이미지	file_p_img	VARCHAR2(18)
✔				상품(약자) 이름 날개	p_name_single	VARCHAR2(30)
				상품(약자) 내용	p_content	VARCHAR2(256)
				상품(약자) 가격	p_price	VARCHAR2(10)

calendar

예약일정

PK	AI	FK	Null	Logical Name	Name	Type
✔				일정 일자	calendar_day	VARCHAR2(10)
✔				일정 시간	calendar_time	VARCHAR2(10)

Open Dining	프로그램 명세서_초기화		
	예약시스템 구축	단 계	설 계
	문서번호: 예약시스템 구축_프로그램 명세서_초기화	작 성 일 자	2019/12/10

A 개요

- 페이지가 로드 된 후 각 모듈의 초기화 작업을 수행한다.
- 모듈은 메뉴, 테이블 선택, 예약 날짜 선택, 장바구니가 노출된다
- 로그인 기능은 팝업 윈도우를 메뉴와 결제창은 Modal 윈도우를 실행하며, 주문 이력 검색 기능은 페이지의 이동없이 비동기 통신한다.
- 예약 날짜를 선택하면 관리자에 의해 미리 작성된 예약시간을 조회하여 리스트로 노출된다. 로그인 상태가 아니라면 조회하지 않는다.
- 장바구니를 선택하면 예약 및 결제 버튼이 노출된다.

B 초기화 프로세스

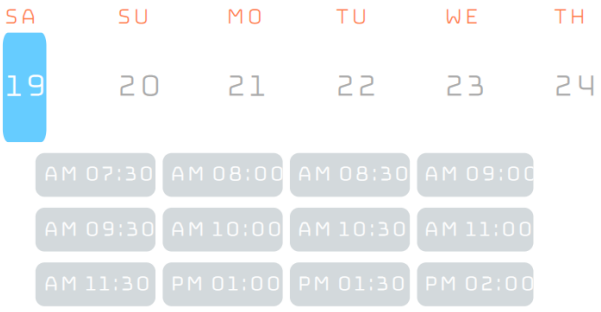
- 로그인 팝업 윈도우
  - 로그인 여부를 확인하고 값이 NULL이면 실행된다.
  - 로그인이 시도될 때마다 현재 사용자의 위치정보가 DB에 갱신된다.

CUSTOMER_EMAIL	CUSTOMER_DISTANCE_INFO	REG_JOINTIME	CUSTOMER_PHONE
exorsa1525@gmail.com	28.9206	190112122354	01042392275
exorsa@naver.com	28.9224	190112122857	01042392275

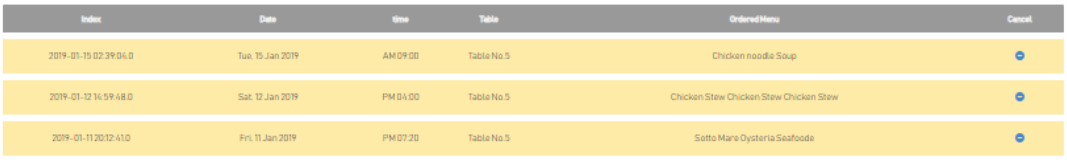
  - 로그인이 완료되면 부모 창을 Reload하여 메뉴와 예약 날짜의 정보를 조회한다.

- 메뉴 조회
  - 현재 등록되어 있는 메뉴의 정보를 조회하여 썸네일 이미지를 화면에 노출시키는 작업이다.
  - 썸 네일 이미지를 클릭하면 Modal 팝업 윈도우가 열리고 선택한 메뉴의 큰 이미지와 품명, 설명, 가격 정보를 보여준다.

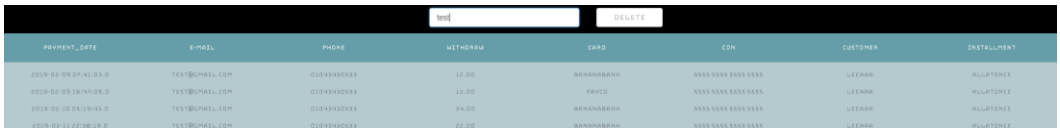
- 예약 가능 일자 조회(Time Picker)
  - 숨김 상태로 초기화된다.
  - 오늘을 기준으로 요일과 날짜가 왼쪽부터 차례로 정렬된다.
  - TimePicker는 중복이 허용되지 않는다.
  - 달력 테이블의 시간 도메인과 예약 테이블의 시간 도메인을 각 날짜별로 'LEFT JOIN'한 값을 반환하여 노출시키는 작업이다.



- 주문 이력 관리
  - 고객이 로그인할 때 볼 수 있는 컨텐츠이다. 고객이 예약을 확인 및 취소할 때 사용한다.
  - 로그인 시 주문한 메뉴의 정보를 조회하여 역순으로 노출시키는 작업이다.
  - 예약 취소 방법은 우측에 있는 Cancel 행의 아이콘을 누르면 해당 줄의 리스트가 취소된다.



- 주문 이력 검색
  - 관리자 페이지에서 고객의 주문내역을 확인하기 위해 활용한다.
  - 관리자로 접속 시 우측 화면 사이드에 숨김 상태로 초기화된다. 마우스 스크롤을 돌리면 전자시계가 사라지고 '주문이력검색모듈'이 슬라이딩된다.
  - 텍스트박스에 값을 입력 또는 Search 버튼을 클릭하면 해당하는 정보가 담긴 릴레이션이 출력된다.



- 위치정보 조회
  - 사용자가 위치정보 수집을 동의하면 조회 가능하나, 위치정보를 측정할 수 없거나 불규칙하게 바뀌는 환경일 경우에는 예약주문 서비스를 제공하기가 어렵다.

  - GPS대신에 HTML5에서 제공하는 Geo-Location 기능을 활용하여 고객이 위치한 위경도와 음식점이 위치한 위경도를 조회한다.
  - 고객이 음식점의 위치를 조회하는 방법은 Store Location을 클릭하면 네이버Cloud 오픈 API의 Map이 팝업창으로 오픈 되고, 현재 음식점의 위치를 지도상으로 보여준다.
  - Distance는 고객으로부터 음식점까지의 거리를 계산하여 km 단위로 출력하는 것을 보여준다.

- 비디오 입출력
  - HTML5의 Video 태그를 활용하여 웹페이지 상에서 비디오를 출력시킨다.
  - 관리자 페이지에서 비디오를 작성하면 클라이언트 페이지에도 동일한 비디오를 출력 된다.

Open Dining	프로그램 명세서_UI/UX		
	예약시스템 구축	단 계	설계
	문서번호: 예약시스템 구축_프로그램 명세서_UI/UX	작 성 일 자	2019/12/11

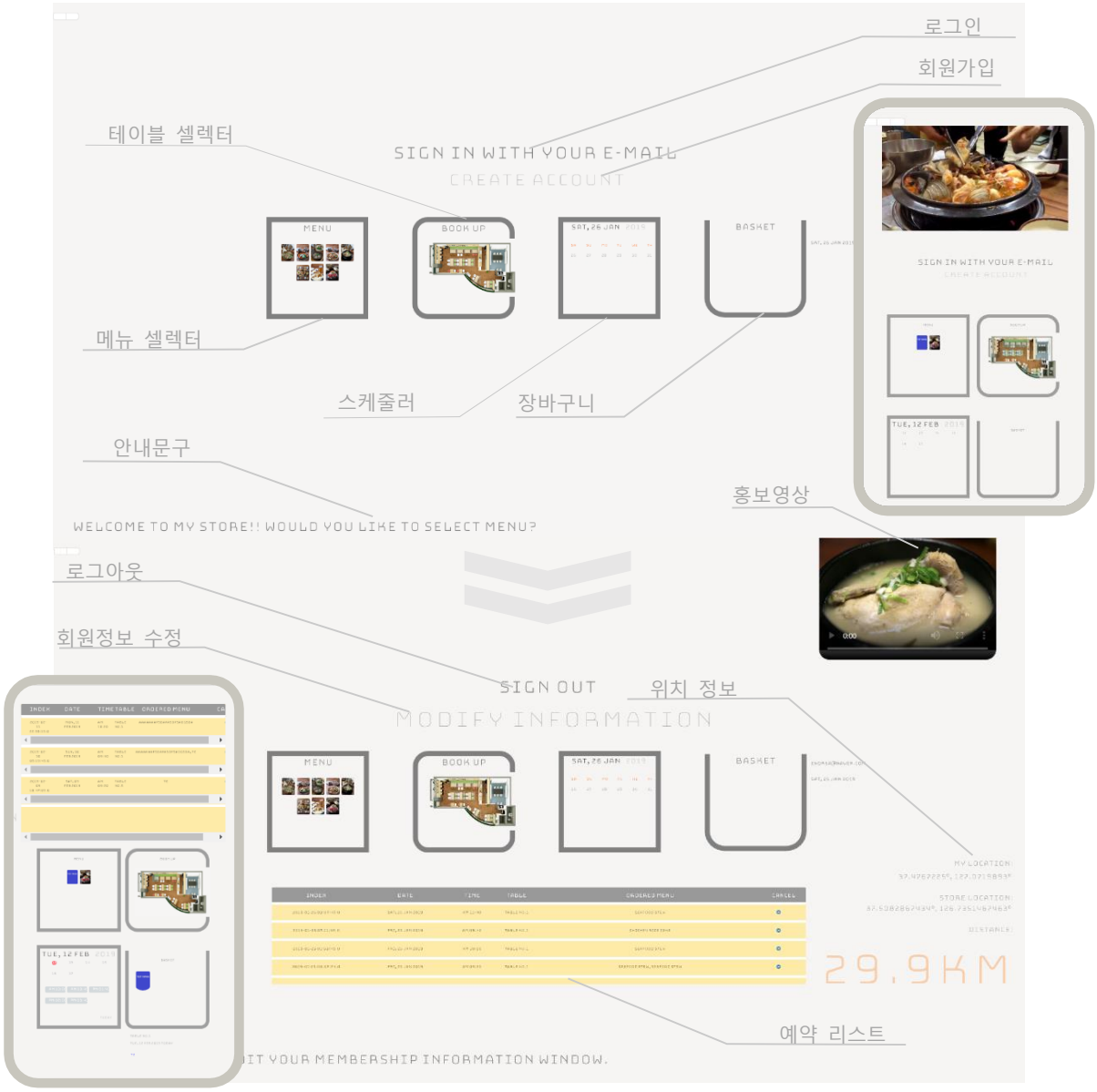
A 개요

- 고객과 접촉하는 클라이언트 사이드에 대한 전략을 수립하는 단계이다.
- HTML5, CSS3, JavaScript, BootStrap, JQuery, zoomooz, HSlider, modal, google-font

B 레이아웃 구성

- 접속 환경에 따른 화면 비율의 최적화를 위해 반응형 웹을 디자인한다.
- 각 모듈을 메인 페이지에 구현하고 모달 및 슬라이딩 처리하여 페이지 이동을 최대한으로 줄인다. 단 로그인, 회원가입, 로그아웃, 회원정보수정, 지도 페이지는 팝업 처리한다.

  - 일반 사용자 접속 시 화면



- 관리자 접속 시 화면

C 모션

- 모듈을 클릭하면 사용자 시점으로 확대되고 여백을 누르면 축소되는 동작 구현
- 광고 영상은 시작과 동시에 실행되고, 여백을 클릭하면 없어진다.
- 마우스 휠을 움직이면 가로 스크롤 형식으로 전환되어 비디오 영상 시청, 전자시계, 위치 정보, 예약 현황관리 게시판 등의 위젯 기능을 활용할 수 있다.

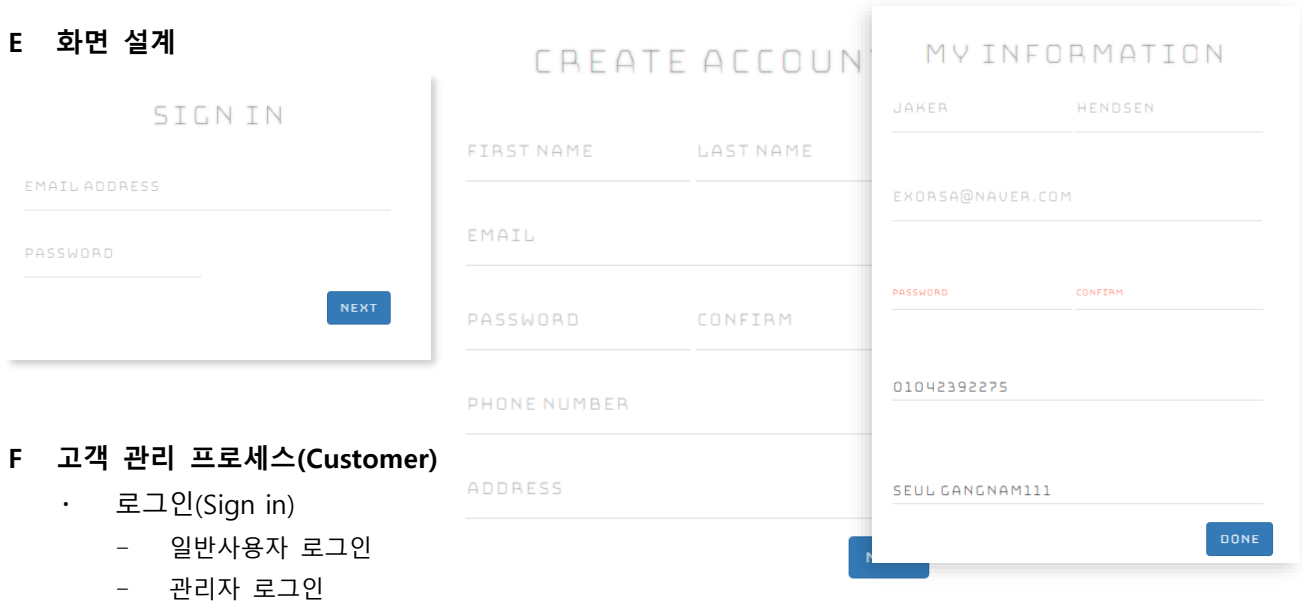
- 고객 시점에서 각 모듈 위에 커서를 올려 놓으면 해당하는 모듈에 대한 안내문구 서비스를 받을 수 있다(JQuery의 mouseenter기능 활용). 타이핑 치는 듯한 효과로 가독성을 높인다.

Open Dining	프로그램 명세서_회원관리		
	예약시스템 구축	단 계	설계
	문서번호: 예약시스템 구축_프로그램 명세서_회원관리	작 성 일 자	2019/12/11

D 개요

- 회원 가입부터 로그인, 회원정보 수정, 로그아웃을 수행하는 페이지이다. 예약 서비스는 로그인 후에 이용할 수 있다. 관리자 계정은 예약 현황을 관리할 수 있다. 로그인 이전에 로그인과 회원가입 링크가 보이며 로그인 후에는 회원정보 수정과 로그아웃 링크가 노출된다. 로그인하지 않은 상태에서 예약주문 및 결제를 할 수 없고, 다른 페이지에 접속할 수 없다. 로그인 할 때마다 접속자의 위치정보가 DB에 새로 갱신된다

E 화면 설계



F 고객 관리 프로세스(Customer)

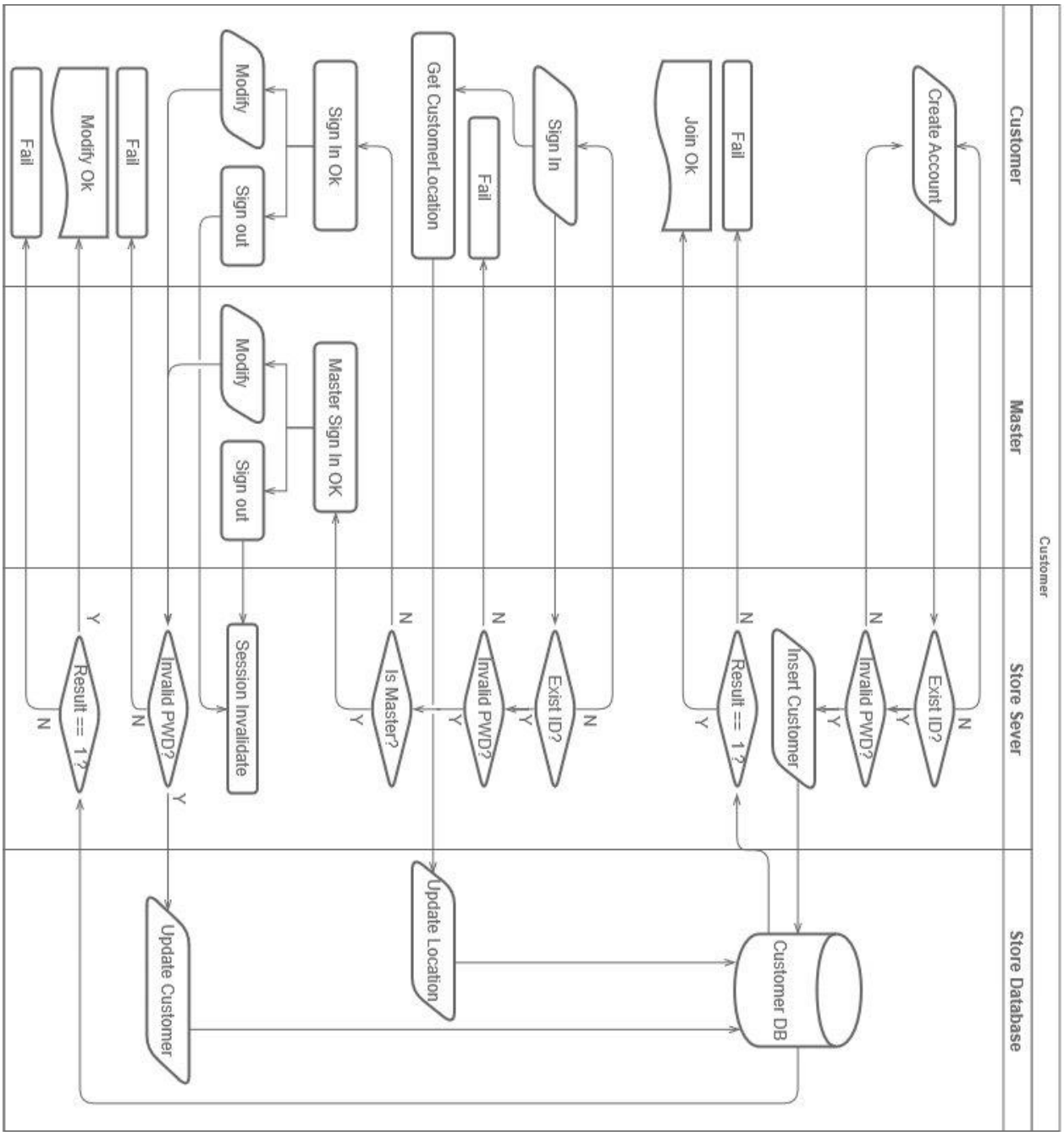
- 로그인(Sign in)
  - 일반사용자 로그인
  - 관리자 로그인
- 회원가입(Create Account)
  - 아이디를 조회한 후 패스워드 일치 여부 검사 수행
  - 로그인 페이지로 복귀하고 아이디 입력 값에 세션 기록이 남는다
- 회원정보 수정(Modify)
  - 패스워드를 조회하여 일치하면 수정 완료
- 로그아웃(Sign Out)
  - 세션이 종료되고 이전 화면으로 초기화된다.

C 관리자 권한 관리

- 일반 사용자용 페이지와 관리자용 페이지를 구분한다.
- 초기 JSP페이지에서 일반 사용자로 접속 시 index.do로 이동하고 관리자 아이디로 접속 시 admin_index.do로 이동한다.
- 관리자로 접속하지 않고 관리자 페이지에 접속하면 일반 사용자 및 로그인 페이지로 이동된다.

D Flow chart

Open Dining	상세 프로세스 흐름도_고객관리		
	예약시스템 구축	단 계	설 계
	문서번호: 예약시스템 구축_상세 프로세스 흐름도_고객관리	작 성 일 자	2019/12/03

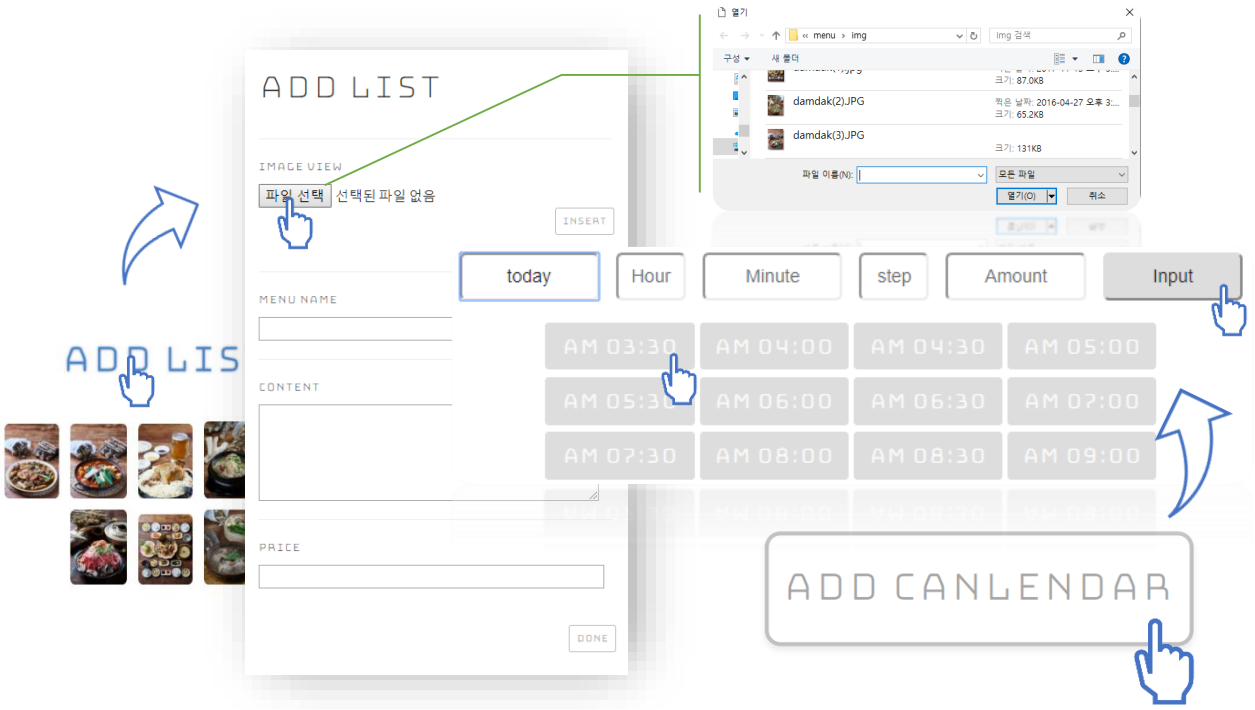


Open Dining	프로그램 명세서_관리자 권한		
	예약시스템 구축	단 계	설계
	문서번호: 예약시스템 구축_프로그램 명세서_관리자 권한	작 성 일 자	2019/12/12

A 개요

- 관리자 권한으로 접속하면 메뉴 파트에 ADD LIST 버튼과 ADD CALENDER 버튼이 노출된다.
- 일반 사용자는 다른 사용자의 예약 리스트에 접근할 수 없지만 관리자는 주문 리스트에서 현재 예약되어 있는 고객의 모든 인스턴스를 삭제할 수 있다.

B 화면설계

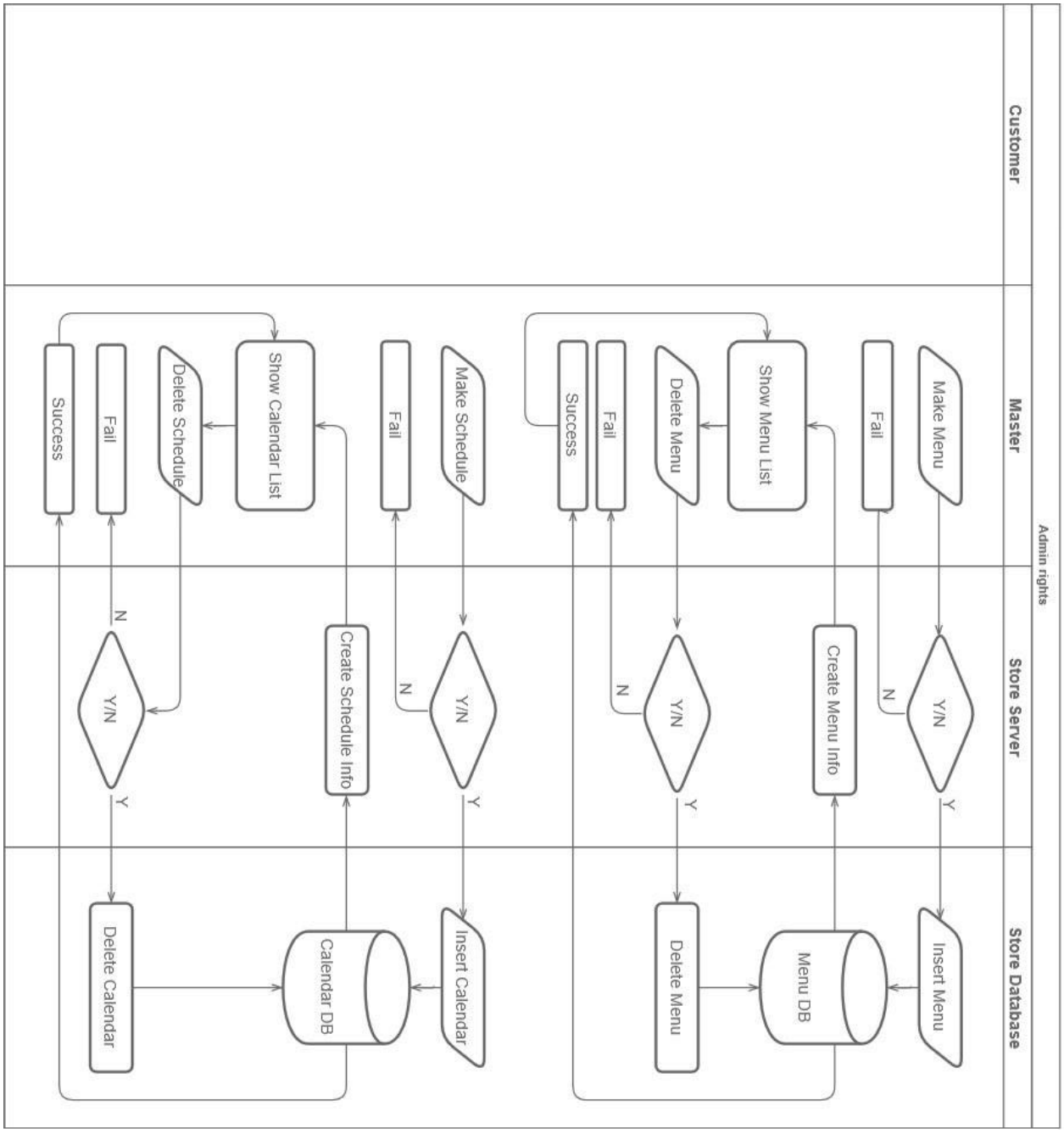


C 관리자 권한 프로세스(Admin rights)

- 메뉴 작성(Make Menu)
  - 메뉴를 등록하면 초기화될 때 메뉴 정보를 조회하여 추가된 메뉴를 리스트 배열에 갱신한다.
- 메뉴 리스트(Menu List)
  - 메뉴 삭제 기능은 선택한 썸네일의 메뉴 컨텐츠 창에서 Delete버튼을 클릭하여 실행된다.
- 예약날짜 작성(Make Schedule)
  - 날짜를 선택하고 ADD CALENDAR 버튼을 클릭하면 스케줄 설정 윈도우가 열린다.
  - 해당 날짜 값이 자동입력 되고 시간과 Term을 조절하여 Submit하면 결과 값이 리스트에 출력되고 셀을 클릭하면 예약날짜가 스케줄링 된다.
- 예약날짜 선택 리스트(Calendar List)
  - 관리자로 로그인했을 때 예약 시간을 클릭하면 예약시간이 선택되지 않고 삭제된다.

D FlowChart

Open Dining	상세 프로세스 흐름도_관리자 권한		
	예약시스템 구축	단 계	설 계
	문서번호: 예약시스템 구축_상세 프로세스 흐름도_관리자 권한	작 성 일 자	2019/12/04

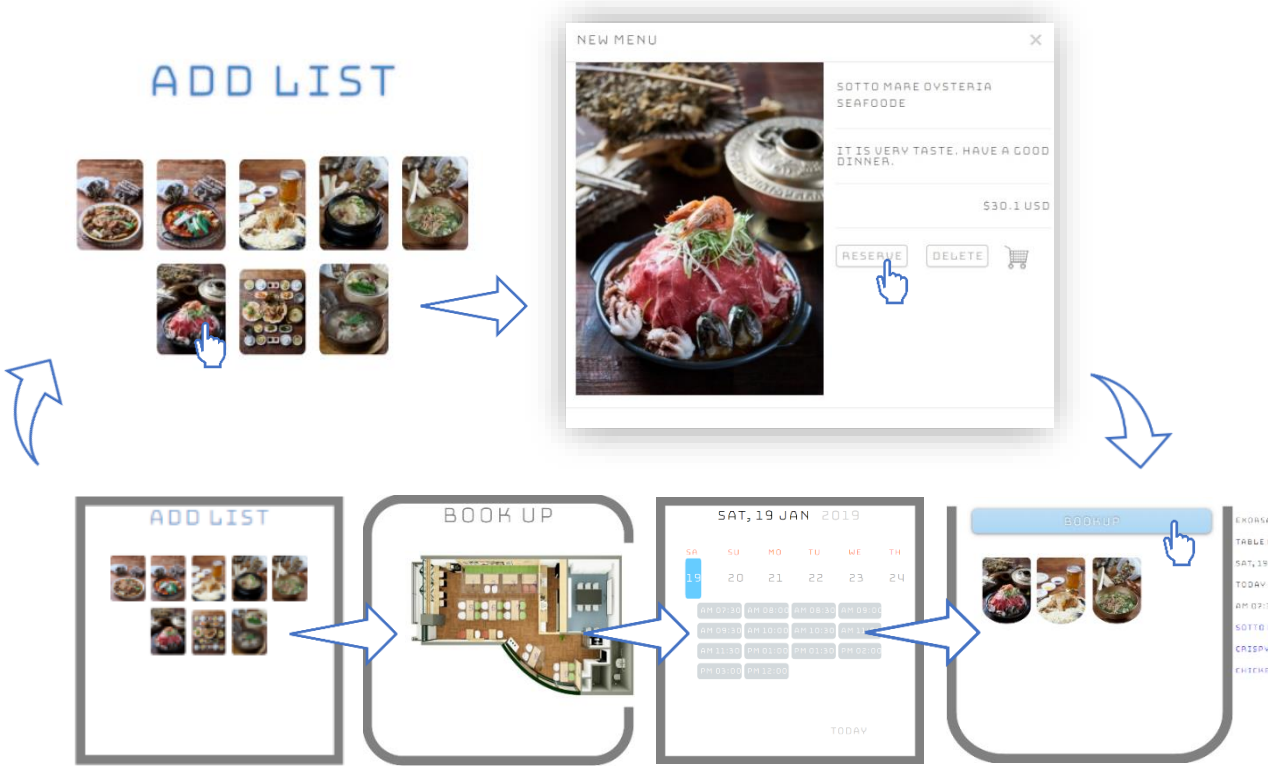


Open Dining	프로그램 명세서_예약		
	예약시스템 구축	단 계	설계
	문서번호: 예약시스템 구축_프로그램 명세서_예약	작 성 일 자	2019/12/13

A 개요

- 주문하기 이전에 임시 저장소인 장바구니에 들어가기까지 예약 과정이다.
- 아이디(이메일), 연월일, 위치 정보는 장바구니에 자동으로 입력되고 아이디는 사용자를 식별하고 위치정보는 결제부에서 올바른 사용자인지 여부를 검사하는데 사용된다.
- 상품 예약, 테이블 번호 예약, 날짜/시간 예약은 필수적으로 해야 하나 예약의 순서는 무의미하다.
-

B 화면설계



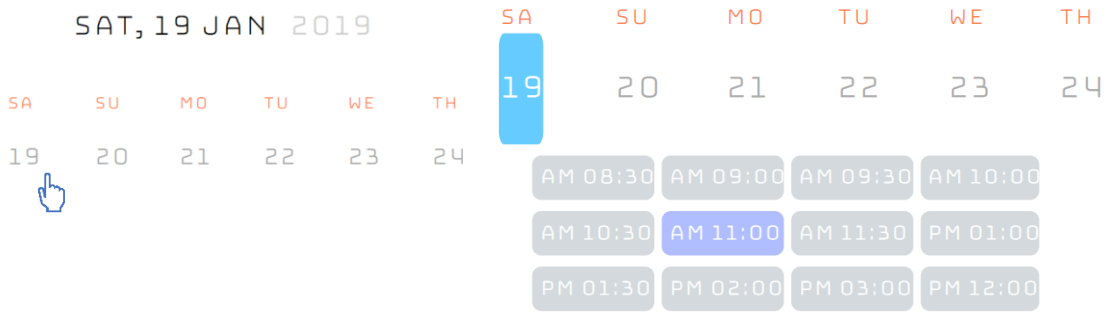
C 예약 프로세스(Book Up)

- 메뉴 선택(Select Menu)
  - 메뉴 리스트에 커서를 올려놓으면 썸네일이 확대되고 선택하면 모달창이 열린다.
- 메뉴 예약(Reserve)

- RESERVE 버튼을 클릭하면 해당 메뉴의 이미지와 상품 명과 가격이 장바구니에 저장되고 모달창이 닫힌다.
- 카트 아이콘을 클릭하면 모달창이 종료되지 않고 클릭 할 때마다 동일한 메뉴가 입력되며 총 액이 증가한다.

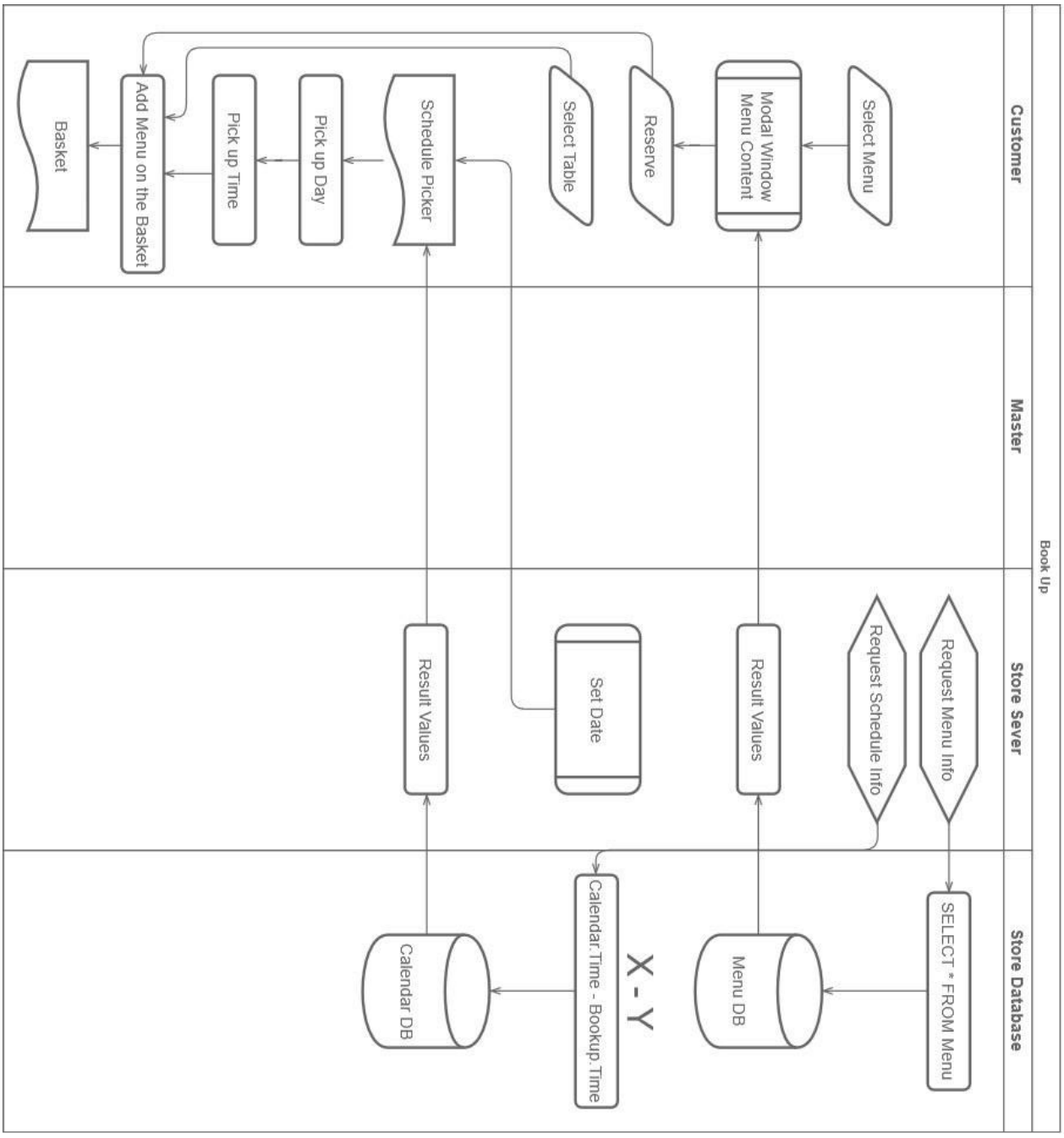
· 테이블 번호 선택(Select Table)
- 평면도를 클릭하면 사용자시점으로 확대
- 테이블 번호를 선택하면 스케줄러 화면으로 시점이 이동된다

· 날짜/시간 예약(Schedule Picker)
- 선택한 테이블에 해당하는 날짜와 시간이 스케줄러에 출력된다.
- 해당 요일의 날짜를 선택하면 예약시간 리스트가 노출되고 선택하면 예약날짜와 시간이 화면에 출력된 후 장바구니로 시점이 이동된다.



· 장바구니(Basket)
- 앞서 지정했던 예약 정보들이 우측에 표기되며 선택한 메뉴와 정보를 확인 가능하다.
- 메뉴는 최대 8개까지 들어간다
- 위에서부터 아이디(이메일), 테이블 번호, 연월일, 예약날짜, 예약시간, 선택한 메뉴
- Book Up 버튼을 클릭하면 주문과 결제를 할 수 있는 모달창이 열린다.

Open Dining	상세 프로세스 흐름도_예약		
	예약시스템 구축	단 계	설 계
	문서번호: 예약시스템 구축_상세 프로세스 흐름도_예약	작 성 일 자	2019/12/04

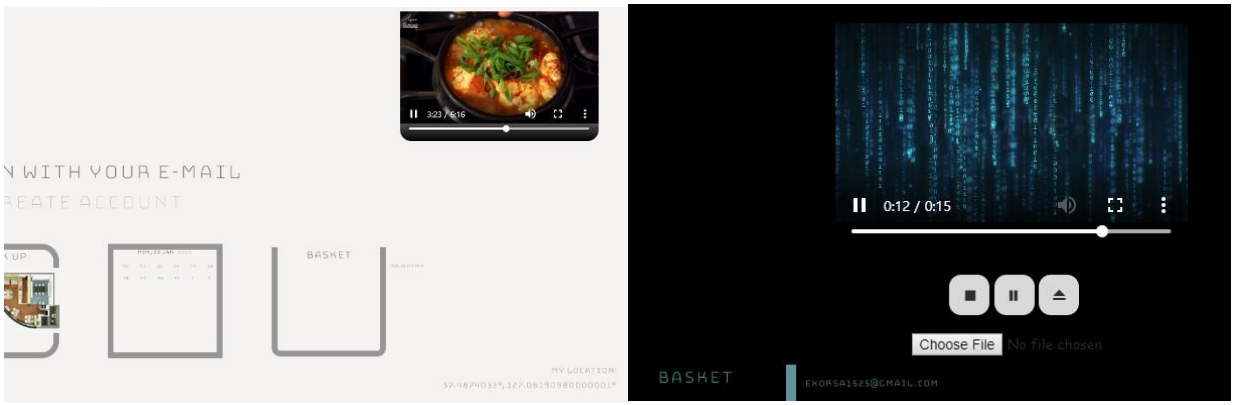


Open Dining	프로그램 명세서_영상 입출력		
	예약시스템 구축	단 계	설계
	문서번호: 예약시스템 구축_프로그램 명세서_영상 입출력	작 성 일 자	2019/12/14

A 개요

- 홍보 영상물을 입/출력하는 과정이다. 마우스를 스크롤 하면 해당 영상이 슬라이딩 되어 상단 우측에 위치하게 된다.
- 관리자 페이지에서 영상물을 삽입하면 데이터를 전달받은 클라이언트 페이지에서 초기화 될 때마다 최신 영상이 노출된다.
- 영상물 관리는 로컬영역의 파일을 입출력 하는 방식과 URI주소를 직접 입력 받아 출력하는 형태가 있다.

B 화면 설계



C 영상 입출력 프로세스

- 영상 업로드
  - 관리자 페이지로 접속하여 로컬영역 안의 파일을 선택하고 영상 삽입 버튼을 누르면 영상이 출력됨과 동시에 영상의 제목이 데이터베이스에 저장된다.
- 영상 출력
  - 초기화시 데이터베이스에 기 저장된 영상의 값을 불러와서 HTML 비디오 태그의 속성 중 파일 경로에 값을 넣는다. 일치하는 값이 나올 때까지 영상 스토리지를 검사한다.
  - 일치하는 영상이 확인되면 비디오 탭에 영상을 송출한다.
  - 영상은 반드시 지정된 스토리지 안에 있어야 실행된다.

D 영상 조작

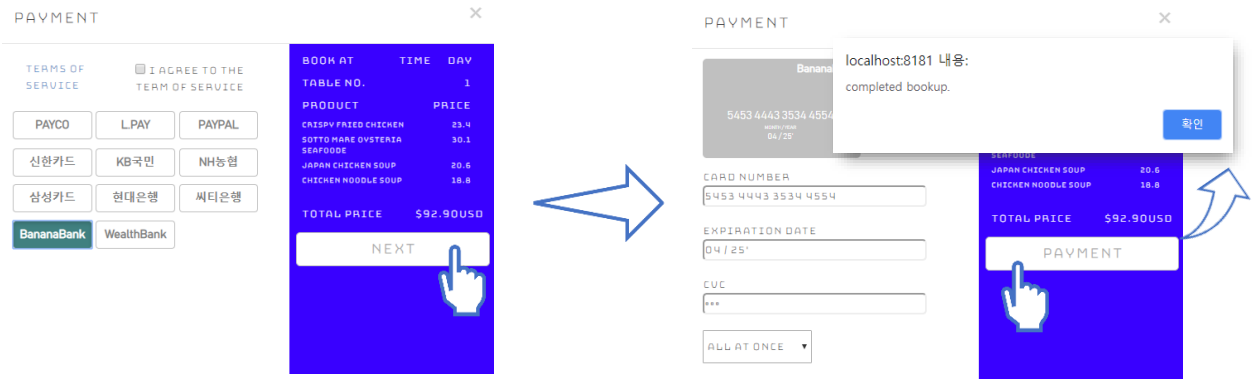
- 영상 조작 메뉴
  - 영상 위에 커서를 올려 놓으면 하단에 영상 조작 메뉴가 나타난다. 왼쪽부터 정지, 일시정지/시작, 영상 삽입 버튼이 위치한다.
  - 삽입 버튼 위에 커서를 올려놓으면 파일 선택(Choose File) 입력폼이 나타난다.
- 영상 중지
  - 정지버튼을 클릭하거나 그 밖에 다른 곳을 클릭하면 정지됨과 동시에 비디오의 뷰가 사라진다.

Open Dining	프로그램 명세서_주문결제		
	예약시스템 구축	단 계	설계
	문서번호: 예약시스템 구축_프로그램 명세서_주문결제	작 성 일 자	2019/12/14

E 개요

- 고객이 예약한 메뉴를 주문/결제하는 페이지이다. 주문서는 고객의 주문 정보를 보여주며 고객이 결제해야 할 금액을 계산한다. 결제는 신용카드의 종류를 선택할 수 있다. 결제 버튼을 누르면 위치기반 보안 검증을 통해 안전을 확보하고 주문과 결제를 트랜잭션 처리한다.

F 화면설계

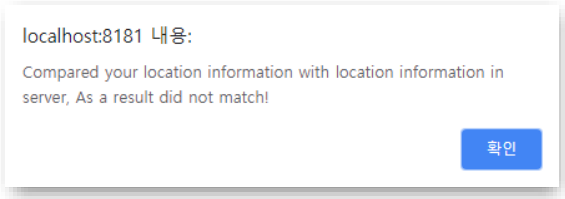


G 주문결제 프로세스(Payment Transaction)

- 예약
  - 장바구니의 Bookup 버튼을 누르면 위와 같은 결제 모달창이 열린다.

- 카드선택
  - 카드사를 선택하고 NEXT버튼을 누르면 선택한 카드사명이 카드 이미지에 기입된다.
- 카드정보 입력
  - 카드번호, 유효기간, CVC번호를 입력하고 할부를 선택한다.
  - 카드번호를 입력하면 카드번호와 유효기간이 카드 이미지에 기입된다.
- 주문결제(Submit)
  - Payment 버튼을 누르면 PG사 측에서 결제 프로세스를 처리한다.
  - 주문과 결제가 동시에 처리되며 해당 고객의 위치정보와 결제가 이루어지는 위치정보를 비교한다(단 결제가 우선 처리되고 결제가 완료되면 주문이 처리된다).

- 현재 위치정보가 NULL값이거나 정보가 일치하지 않은 경우 결제자의 위치정보와 서버상의 위치정보가 일치하지 않는다는 알림창을 띄우고 결제가 취소된다.



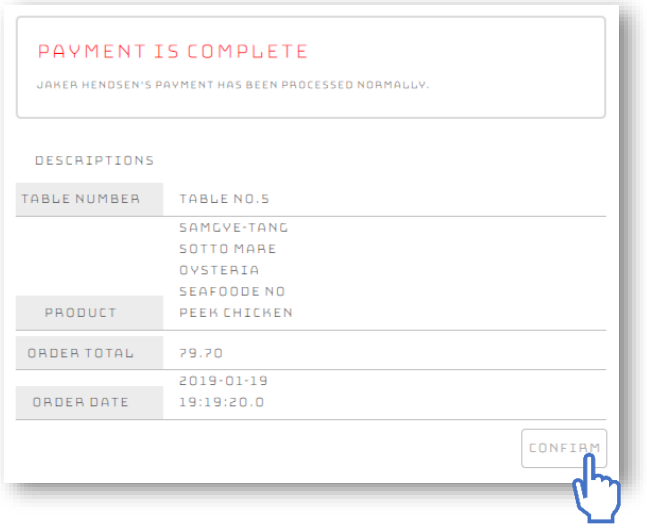
- 결제 혹은 예약주문 트랜잭션 과정에서 문제가 생겼을 경우 알림창을 띄우고 Rollback한다.
- 정보가 일치하면 결제를 진행하고 결제와 예약주문을 동시에 처리한다.
- 결제정보는 PG사 서버에 로그를 남기고 해당 거래소(카드사/은행/이통사)에 대금결제를 요청한다

	PAYMENT_DATE	CREDITCARDFIRM	CDN	CUSTOMER_NAME	EMAIL	INVAL	CVC	PHONE	INSTALLMENT	DISTANCE	WITHDRAW
1	190119191920	BananaBank	77...	Jakerhendsen	exors...	07/20	777	01042...	allatonce	28.9176	79.70
2	190119055236	BananaBank	77...	Jakerhendsen	exors...	07/20	777	01042...	allatonce	28.9164	18.80
3	190115023904	BananaBank	77...	Jakerhendsen	exors...	07/20	777	01042...	allatonce	28.9156	18.80
4	190115023726	BananaBank	77...	Jakerhendsen	exors...	07/20	777	01042...	allatonce	28.9156	18.80
5	190112145948	BananaBank	77...	Jakerhendsen	exors...	07/20	777	01042...	allatonce	28.9174	81.00

- 거래소에서 처리한 결과가 PG사에 통보되면 PG사는 클라이언트에게 결과를 전달한다.

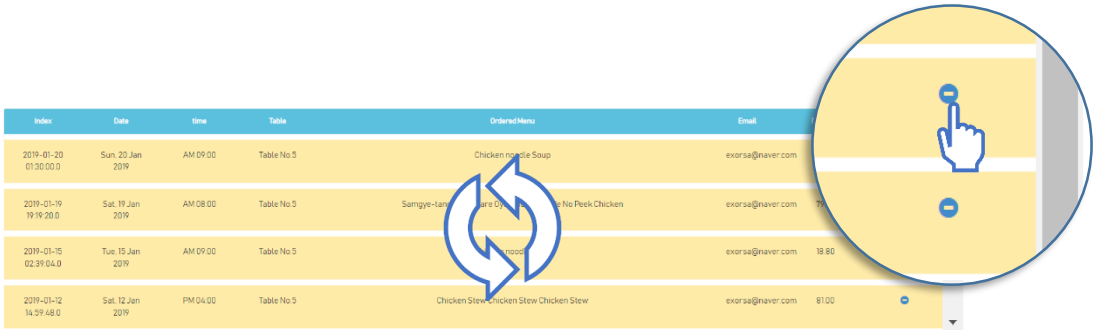
· 결제완료(Complete Payment)

- 트랜잭션 결과 올바른 값이 반환되면 결제완료 팝업이 열리고 주문결제 정보가 출력된다.
- 결제가 정상적으로 완료되면 결제완료 팝업을 띄우고 Confirm 버튼을 누르면 주문이력 정보를 초기화한다.

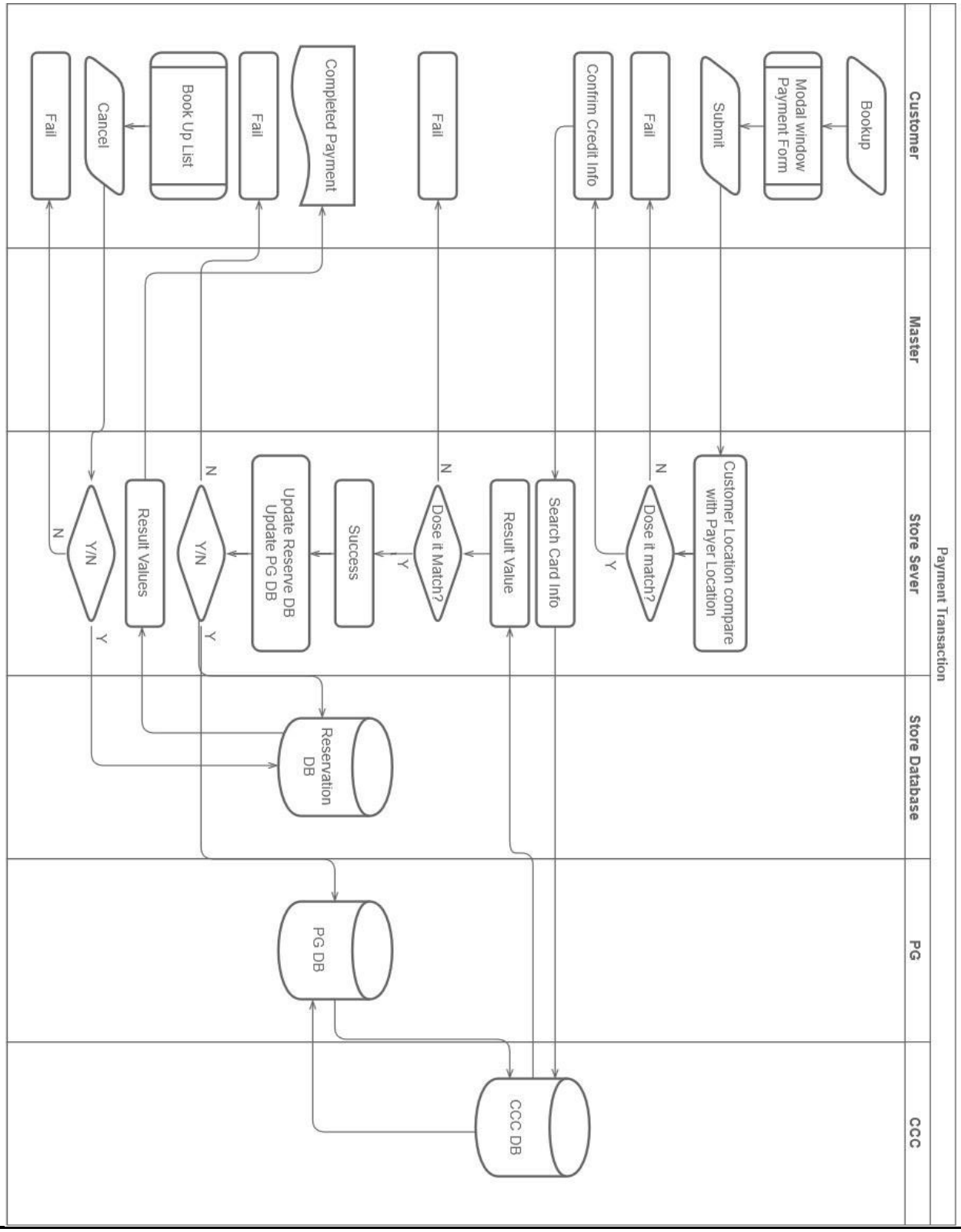


· 예약 취소 기능

- 오른쪽의 Cancel 버튼을 누르면 예약된 메뉴의 Index가 파라미터 값으로 전달되며 AJAX 방식으로 주문DB에 접근하여 해당 품목의 DELETE 쿼리를 수행한다.
- 사용자로부터 취소요청을 받은 PG사에서 거래소에 취소요청을 전달한다
- PG사로부터 취소요청을 받은 거래소에서 이미 결제된 내역을 Rollback하여 PG사에 결과를 리턴하고 PG사는 사용자에게 결과를 전달한다.



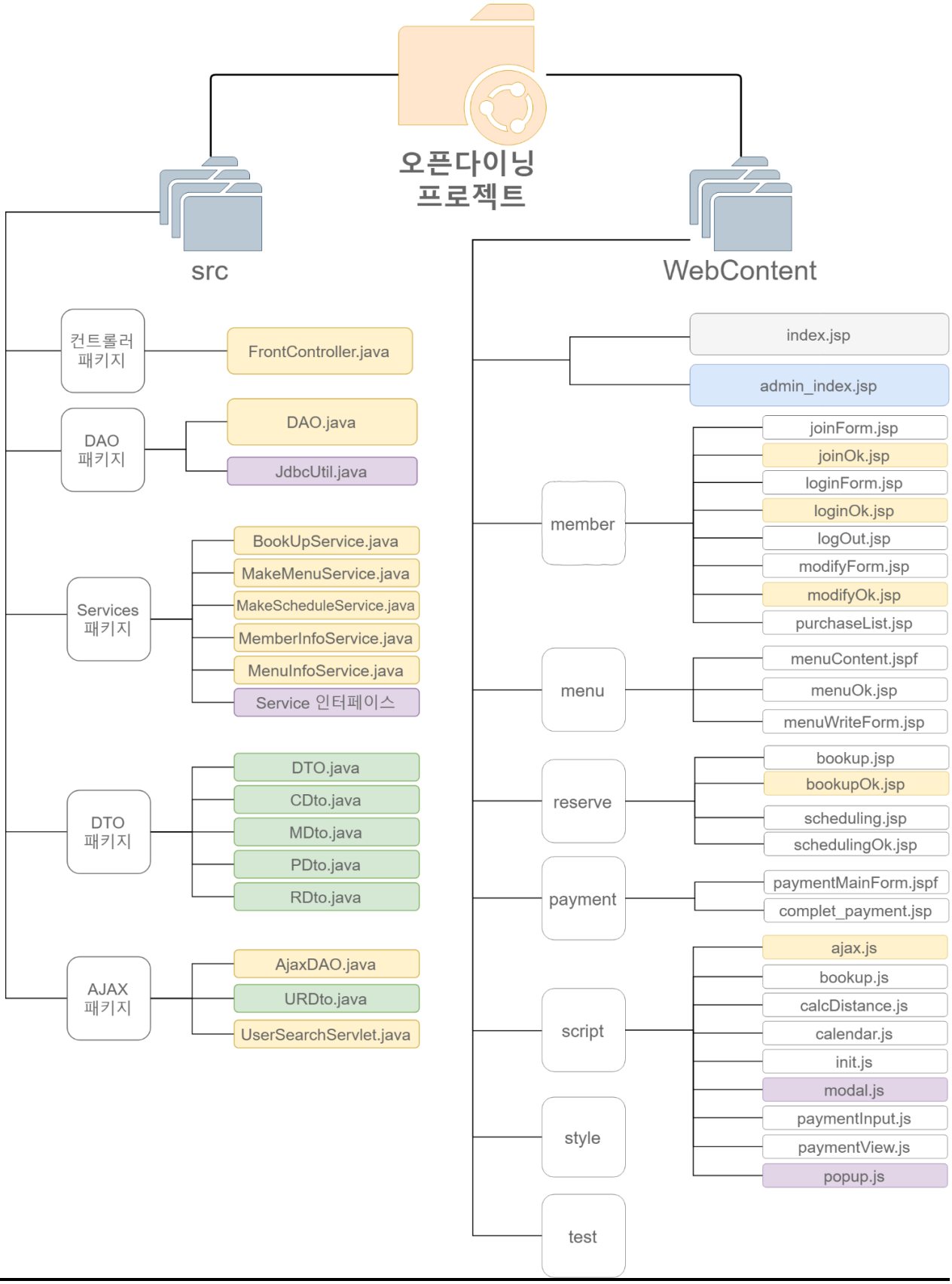
Open Dining	상세 프로세스 흐름도_결제		
	예약시스템 구축	단 계	설계
	문서번호: 예약시스템 구축_상세 프로세스 흐름도_결제	작 성 일 자	2019/12/05



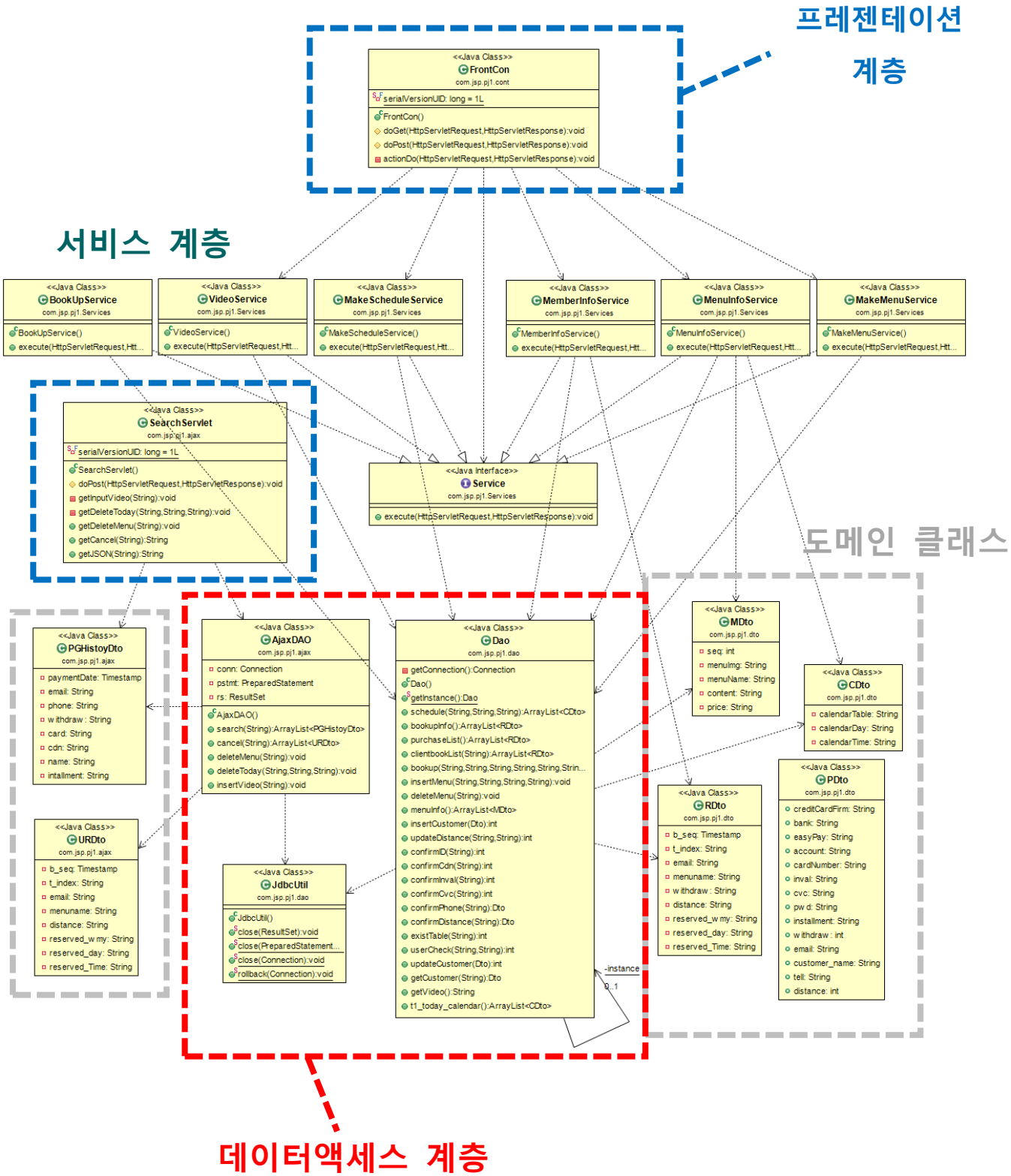
개 발

GitHub 주소 https://github.com/Matt-solver/Open-Dining_2019

A 파일 디렉토리 구조



B 클래스 다이어그램


프레젠테이션 계층
서비스 계층
도메인 클래스
데이터액세스 계층
<<Java Class>>
FrontCon
com.jsp.pj1.cont
serialVersionUID: long = 1L
FrontCon()
doGet(HttpServletRequest,HttpServletResponse):void
doPost(HttpServletRequest,HttpServletResponse):void
actionDo(HttpServletRequest,HttpServletResponse):void
<<Java Class>>
BookUpService
com.jsp.pj1.Services
BookUpService()
execute(HttpServletRequest,Htt...
<<Java Class>>
VideoService
com.jsp.pj1.Services
VideoService()
execute(HttpServletRequest,Htt...
<<Java Class>>
MakeScheduleService
com.jsp.pj1.Services
MakeScheduleService()
execute(HttpServletRequest,Htt...
<<Java Class>>
MemberInfoService
com.jsp.pj1.Services
MemberInfoService()
execute(HttpServletRequest,Htt...
<<Java Class>>
MenuInfoService
com.jsp.pj1.Services
MenuInfoService()
execute(HttpServletRequest,Htt...
<<Java Class>>
MakeMenuService
com.jsp.pj1.Services
MakeMenuService()
execute(HttpServletRequest,Htt...
<<Java Class>>
SearchServlet
com.jsp.pj1.ajax
serialVersionUID: long = 1L
SearchServlet()
doPost(HttpServletRequest,HttpServletResponse):void
getInputVideo(String):void
getDeleteToday(String,String,String):void
getDeleteMenu(String):void
getCancel(String):String
getJSON(String):String
<<Java Interface>>
Service
com.jsp.pj1.Services
execute(HttpServletRequest,HttpServletResponse):void
<<Java Class>>
PGHistoyDto
com.jsp.pj1.ajax
paymentDate: Timestamp
email: String
phone: String
w ithdraw : String
card: String
cdn: String
name: String
intallment: String
<<Java Class>>
AjaxDAO
com.jsp.pj1.ajax
conn: Connection
pstmt: PreparedStatement
rs: ResultSet
AjaxDAO()
search(String):ArrayList<PGHistoyDto>
cancel(String):ArrayList<URDto>
deleteMenu(String):void
deleteToday(String,String,String):void
insertVideo(String):void
<<Java Class>>
Dao
com.jsp.pj1.dao
getConnection():Connection
Dao()
getInstance():Dao
schedule(String,String,String):ArrayList<CDto>
bookupInfo():ArrayList<RDto>
purchaseList():ArrayList<RDto>
clientbookList(String):ArrayList<RDto>
bookup(String,String,String,String,String,Strin...
insertMenu(String,String,String,String):void
deleteMenu(String):void
menuInfo():ArrayList<MDto>
insertCustomer(Dto):int
updateDistance(String,String):int
confirmID(String):int
confirmCdn(String):int
confirmInval(String):int
confirmCvc(String):int
confirmPhone(String):Dto
confirmDistance(String):Dto
existTable(String):int
userCheck(String,String):int
updateCustomer(Dto):int
getCustomer(String):Dto
getVideo():String
t1_today_calendar():ArrayList<CDto>
-instance
0..1
<<Java Class>>
MDto
com.jsp.pj1.dto
seq: int
menuImg: String
menuName: String
content: String
price: String
<<Java Class>>
CDto
com.jsp.pj1.dto
calendarTable: String
calendarDay: String
calendarTime: String
<<Java Class>>
RDto
com.jsp.pj1.dto
b_seq: Timestamp
t_index: String
email: String
menuname: String
w ithdraw : String
distance: String
reserved_w my: String
reserved_day: String
reserved_Time: String
<<Java Class>>
PDto
com.jsp.pj1.dto
creditCardFirm: String
bank: String
easyPay: String
account: String
cardNumber: String
inval: String
cvc: String
pw d: String
installment: String
w ithdraw : int
email: String
customer_name: String
tell: String
distance: int
<<Java Class>>
URDto
com.jsp.pj1.ajax
b_seq: Timestamp
t_index: String
email: String
menuname: String
distance: String
reserved_w my: String
reserved_day: String
reserved_Time: String
<<Java Class>>
JdbcUtil
com.jsp.pj1.dao
JdbcUtil()
close(ResultSet):void
close(PreparedStatement...
close(Connection):void
rollback(Connection):void

C 데이터베이스 생성

· 계정 생성

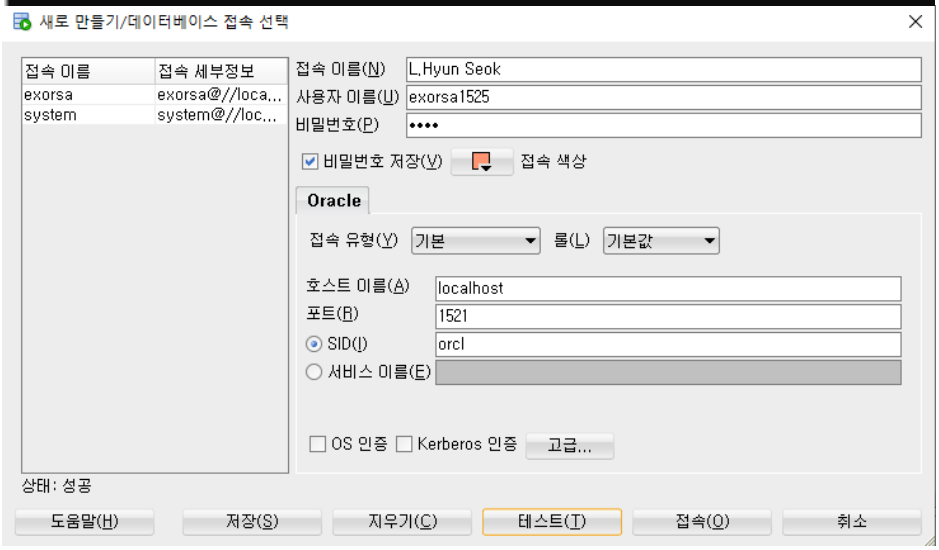
```
SQL> create user exorsa1525 identified by 1234;

User created.

SQL> grant connect, resource, dba to exorsa1525;

Grant succeeded.

SQL>
```



· 테이블 생성

```
/*Add Customer*/
CREATE TABLE customer(
    customer_index          NUMBER          NULL,
    first_name              VARCHAR2(10)    NOT NULL,
    last_name               VARCHAR2(16)    NOT NULL,
    customer_email          VARCHAR2(26)    NOT NULL,
    customer_pwd            VARCHAR2(16)    NOT NULL,
    customer_phone          VARCHAR2(16)    NULL,
    customer_addr           VARCHAR2(40)    NULL,
    reg_joinTime            TIMESTAMP       NOT NULL,
    customer_distance_info  VARCHAR2(10)    NULL,
    CONSTRAINT CUSTOMER_PK PRIMARY KEY (customer_email)
);
```

D DB연동을 위한 Connection Pool 작성

· 성능 문제를 해결하기 위해 Connection Pool 기법을 사용하는 작업

- 매번 커넥션을 생성하지 않고 DB와 연결된 Connection을 미리 만들어서 Pool속에 저장한다.
- Connection을 필요할 때마다 가져다 쓰고 close()를 호출하여 반환한다.

- context.xml에 소스코드 추가

```
<Resource auth="Container"
driverClassName="oracle.jdbc.driver.OracleDriver"
maxActive="50" maxWait="1000"
name="jdbc/oracle" password="tldhstks12"
type="javax.sql.DataSource"
url="jdbc:oracle:thin:@localhost:1521:orcl"
username="exorsa"/>
```

동시에 사용할 수 있는 최대 커넥션 수는 50 으로 설정하고 최대 타임아웃은 1 초로 설정

· Connection Pool 사용
  - getConnection() 메소드를 활용
  - 컨텍스트를 생성하고 lookup()으로 DataSource 객체를 찾는다. DataSource에서 Context를 가져와서 사용한다.

```
private Connection getConnection() {
        Context cont = null;
        DataSource dataSource = null;
        Connection conn = null;

       try {
              cont = new InitialContext();
              dataSource = (DataSource)
              cont.lookup("java:comp/env/jdbc/oracle");
              conn = dataSource.getConnection();

       } catch (Exception e) {
              e.printStackTrace();
       }
              return conn;
 }
```

· **DAO에서 Connection 접근 방식**
· 서비스 클래스에서 접근 : 호출될 때마다 새로운 서비스 객체를 생성해서 사용
· 싱클톤 패턴 적용 : 매번 동일한 객체를 리턴하므로 같은 객체를 참조하게된다.
  - 접근제어 수식어는 private이기때문에 외부 클래스에서는 객체를 생성할 수 없게된다.
  - 유일하게 객체에 접근할 수 있는 getInstance() 정적메소드를 통해서만 DAO에 접근할 수 있게 하였다.

```
private static Dao instance = new Dao();

public Dao() {
}
public static Dao getInstance() {

       return instance;
}
```

  - JSP페이지에서 DAO접근

```
Dao dao = Dao.getInstance();
        dao.confirmID(dto.getCustomerEmail())
```

E 서버 사이드 구현

- Model(DTO) 작성
  - **필드**

```
public class Dto {
    private int customerIndex; // 고객 번호
    private String firstName;    // 성
    private String lastName;     // 이름
    private String customerEmail; // 고객 이메일
    private String customerPwd; // 고객 암호
    private String customerPhone; // 고객 전화번호
    private String customerAddr; // 고객 주소
    private Timestamp regJointime; // 등록 가입일자
    private String customerDistanceInfo; // 고객 거리 정보

    public int getCustomerIndex() {
            return customerIndex;
        }
        public void setCustomerIndex(int customerIndex) {
            this.customerIndex = customerIndex;
        }
        public String getFirstName() {
            return firstName;
        }
        public void setFirstName(String firstName) {
            this.firstName = firstName;
        }
                            .
                            .
                            .
```

- Data Access Object 클래스 작성
  - DB에 대한 CRUD 메서드 작업

```
    public int insertCustomer(Dto dto) {
        int ri = 0;

        Connection conn = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;

        String query = "INSERT INTO customer "
                     + "VALUES(m_seq.nextval,?,?,?,?,?,?,sysdate,0)";

        try {
            conn = getConnection();
            pstmt = conn.prepareStatement(query);
            pstmt.setString(1, dto.getFirstName());
            pstmt.setString(2, dto.getLastName());
            pstmt.setString(3, dto.getCustomerEmail());
            pstmt.setString(4, dto.getCustomerPwd());
            pstmt.setString(5, dto.getCustomerPhone());
            pstmt.setString(6, dto.getCustomerAddr());
            pstmt.executeUpdate();
```

```
                ri = Dao.MEMBER_JOIN_SUCCESS;

            } catch (Exception e) {
                e.printStackTrace();
            }finally {
                JdbcUtil.close(pstmt);
                JdbcUtil.close(conn);
            }
            return ri;
    }
```

- Front Controller 작성
  - Servlet 배치정보 설정(@WebServlet)
  - do, post로 호출되어도 actionDo()로 배치될수 있도록 설정

```
@WebServlet("*.do")
public class FrontCon extends HttpServlet {
private static final long serialVersionUID = 1L;


  public FrontCon() {
       super();
  }
  protected void doGet(HttpServletRequest request, HttpServletResponse
  response) throws ServletException, IOException {
            actionDo(request, response);
  }
  protected void doPost(HttpServletRequest request, HttpServletResponse
  response) throws ServletException, IOException {
            actionDo(request, response);
  }

  private void actionDo(HttpServletRequest request, HttpServletResponse
  response) throws ServletException, IOException {
            request.setCharacterEncoding("UTF-8");

            String viewPage = null;
            Service service = null;
```

  - 전체 uri에서 Context 경로를 제외한 뒷 부분을 가져온다
  - 해당하는 서비스 객체를 지정하여 DAO에서 처리한다.

```
            String uri = request.getRequestURI();
            String conPath = request.getContextPath();
            String com = uri.substring(conPath.length());

            if(com.equals("/index.do")) {
                service = new MenuInfoService();
                service.execute(request, response);
                service = new MemberInfoService();
                service.execute(request, response);
                viewPage = "index.jsp";

            }else if(com.equals("/menu/write.do")) {
                service = new MakeMenuService();
                service.execute(request, response);
                viewPage = "menuOk.jsp";
```

```
		}else if(com.equals("/bookup.do")) {
			viewPage = "reservation/bookupOk.jsp";

		}else if(com.equals("/reservation/schedule.do")) {
			service = new MakeScheduleService();
			service.execute(request, response);
			viewPage = "schedulingOk.jsp";
		}
```

- Dispatcher 포워딩으로 현재 페이지의 request/response 객체를 그대로 가지고 다른 요청 페이지로 이동

```
		RequestDispatcher dispatcher
		= request.getRequestDispatcher(viewPage);
		dispatcher.forward(request, response);

		}
	}
```

- Sub Controller (Interface 클래스) 작성
  - Front Controller 다음에 실제 서비스를 처리하는 컨트롤러 작업
  - 각 Sub Controller(service)에 상속할 공통 인터페이스를 작성한다.

```
public interface Service {
 void execute(HttpServletRequest request, HttpServletResponse response);
 }
```

- Sub Controller (Service 클래스) 작성

```
public class MakeScheduleService implements Service {

	@Override
	public void execute(HttpServletRequest request,
				HttpServletResponse response) {
		String day = request.getParameter("calendarDay");
		String time = request.getParameter("calendarTime");

		Dao dao = Dao.getInstance();
		dao.schedule(day, time);

	}

}
```

F AJAX 구현

- AJAX : Controller(Servlet-mapping) 구현

```
@WebServlet("*.com")
public class UserSearchServlet extends HttpServlet {
  private static final long serialVersionUID = 1L;

  protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
		request.setCharacterEncoding("UTF-8");
		response.setContentType("text/html;charset=UTF-8");

		String uri = request.getRequestURI();
		String conPath = request.getContextPath();
```

```
		String com = uri.substring(conPath.length());

		if(com.equals("/index.com")) {
			String keyword = request.getParameter("keyword");
			response.getWriter().write(getJSON(keyword));
		}else if(com.equals("/cancel.com")) {
			String bseq = request.getParameter("bseq");
			response.getWriter().write(getCancel(bseq));

		}else if(com.equals("/deleteMenu.com")) {
			System.out.println("servlete deletemenu");
			String menuname = request.getParameter("menuname");
			getDeleteMenu(menuname);

		}else if(com.equals("/deleteCalendar.com")) {
			String time = request.getParameter("time");
			String day = request.getParameter("day");
			System.out.println("day : " + day);
			getDeleteToday(time, day);
		}
	}
	private void getDeleteToday(String time, String day) {
		if(time == null) time = "";
		AjaxDAO ajaxDAO = new AjaxDAO();
		ajaxDAO.deleteToday(time, day);
	}
	public void getDeleteMenu(String menuname) {
		System.out.println("getdeletemenu : " + menuname);
		if(menuname == null) menuname = "";
		AjaxDAO ajaxDAO = new AjaxDAO();
		ajaxDAO.deleteMenu(menuname);
	}

	// Delete Reservation
	public String getCancel(String bseq) {
		if(bseq == null) bseq = "";
		String result = null;
		AjaxDAO ajaxDAO = new AjaxDAO();
		ajaxDAO.cancel(bseq);

		return result;
	}

	//제이슨으로 변환
	public String getJSON(String keyword) {

		if(keyword == null) keyword = "";
		StringBuffer result = new StringBuffer("");

		result.append("{\"result\":[");

		AjaxDAO ajaxDAO = new AjaxDAO();
		ArrayList<URDto> userList = ajaxDAO.search(keyword);
		System.out.println(userList);

		for(int i=0; i < userList.size(); i++) {
				result.append("[{\"value\": \"" +
		   userList.get(i).getB_seq() + "\"},");
				result.append("{\"value\": \"" +
		   userList.get(i).getReserved_wmy() + "\"},");
```

```
                                result.append("{\"value\": \"" +
                    userList.get(i).getReserved_Time() + "\"},");
                                result.append("{\"value\": \"" +
                    userList.get(i).getT_index() + "\"},");
                                result.append("{\"value\": \"" +
                    userList.get(i).getMenuname() + "\"},");
                                result.append("{\"value\": \"" +
                    userList.get(i).getEmail() + "\"},");
                                result.append("{\"value\": \"" +
                    userList.get(i).getDistance() + "\"},");
                                result.append("{\"value\": \"" +
                    userList.get(i).getDistance() + "\"}],");
                }

                result.append("]}");
                System.out.println(result);
                return result.toString();
        }

    }
```

- AJAX : Data Access Object 클래스 작성

```
        public ArrayList<URDto> search(String keyword) {
        String query = "SELECT * FROM bookup "
                        + "WHERE t_index||menuname||email||"
                        + "reserved_wmy||reserved_day||reserved_Time LIKE ?";
                ArrayList<URDto> userList = new ArrayList<URDto>();

                try {
                        pstmt = conn.prepareStatement(query);
                       //와일드카드 : 특정한 요소가 포함된 모든 요소가 출력
                        pstmt.setString(1, "%" + keyword + "%");
                        rs = pstmt.executeQuery();

                        while (rs.next()) {
                                URDto dto = new URDto();
                                dto.setB_seq(rs.getTimestamp(1));
                                dto.setT_index(rs.getString(2));
                                dto.setEmail(rs.getString(3));
                                dto.setMenuname(rs.getString(4));
                                dto.setDistance(rs.getString(5));
                                dto.setReserved_wmy(rs.getString(6));
                                dto.setReserved_day(rs.getString(7));
                                dto.setReserved_Time(rs.getString(8));
                                userList.add(dto);
                        }

                } catch (Exception e) {
                        e.printStackTrace();
                }
                        return userList;
            }
```

- AJAX : 프론트엔드 작성
  - 자바스크립트 환경에서 XMLHttpRequest() 호출
  - URI로 데이터를 전달하기 위해 encodeURIComponent()를 사용하여 문자열을 인코딩한다.
  - Request를 통해 Post 방식으로 uri를 날려준다.
  - 접속상태 확인 후 받아온 객체를 제이슨으로 변환

- 제이슨 객체를 JS에서 HTML상에 뿌려준다.

```
var request = new XMLHttpRequest();

function searchProcess() {
        var AT = document.getElementById("ajaxTable");
        AT.innerHTML = "";

        //접속상태 확인
        if(request.readyState = 4 && request.status == 200) {
        var object = eval('(' + request.responseText + ')');

        //받아온 객체를 제이슨으로 변환
        var result = object.result;

        // 제이슨 데이터를 HTML에 적용
        for(var i = 0; i < result.length; i++) {
                var row = AT.insertRow(0);

                for(var j = 0; j < result[i].length; j++) {

                var cell = row.insertCell(j);
               cell.innerHTML = '<br />' + result[i][j].value + '<br />'

          }
          cell.innerHTML = "<a href='' id='' class='cs glyphicon
          glyphicon-minus-sign' style='text-decoration:none'
          onclick='(function(e){e.preventDefault();})(event)'></a>";
        }
    }
}
window.onload = function() {
        searchFunction();
}
$('.cs').click(function (){
        alert('a');
        var url = $(this).attr("id");
        alert(url);

});
function searchFunction() {  // 주문목록 검색(와일드카드)
        request.open("Post", "./index.com?keyword=" +
        encodeURIComponent(document.getElementById("keyword").value),
        true);
        request.onreadystatechange = searchProcess;
        request.send(null);

}
function cancel() {   // 예약주문 취소
        request.open("Post", "./cancel.com?bseq=" +
        encodeURIComponent($('.bseq').val()), true);
        request.onreadystatechange = searchProcess;
        request.send(null);
        alert("Canceled reservation number "+ $('.bseq').val());
        setTimeout(function() {
        self.location.reload();
        }, 500);
```

```
}
function deleteProduct() {   // 품목 삭제
	request.open("Post", "./deleteMenu.com?menuname=" +
	encodeURIComponent($('#mn').val()), true);
	request.send(null);
	alert('Deleted menu ' + $('#mn').val())

}
function deleteCalendar(time, day) {   // 예약가능 시간 삭제
	request.open("Post", "./deleteCalendar.com?time=" +
	encodeURIComponent(time) + "&day=" + encodeURIComponent(day),
	true);
	request.send(null);
	alert("Deleted " + day + " time is "+ time)

	setTimeout(function() {
		self.location.reload();
	}, 500);
 }
```

G 스케줄러(Time Picker)의 구현

- System time 기반의 예약 날짜 및 시간을 작성하기 위한 로직이다.
- 윤달을 계산하여 날짜의 오차를 방지한다
- 왼쪽부터 오른쪽 방향으로 오늘날짜부터 일요일을 제외한 6일을 계산하여 요일과 날짜는 같은 위치에서 작동되도록 HTML상에 렌더링한다.

연/월/일 구현

```
// System time을 기준으로 한다.
Date date= new java.util.Date();

SimpleDateFormat sdf = new SimpleDateFormat("yyyy년 MM월 dd일 HH시 mm분 ss초 E요일");
Date now = new Date();

SimpleDateFormat dayofweek = new SimpleDateFormat("EEE, dd MMM", Locale.ENGLISH);
String dow = dayofweek.format(date);
String EEE = dow.substring(0, 2);   //Day of week(요일)형식을 포맷한다.
// 연,월,일,시,분의 형식을 포맷한다.
String strDate = sdf.format(now);
String yyyy = strDate.substring(0, 4);
String MM = strDate.substring(6, 8);
String dd = strDate.substring(10, 12);
String HH = strDate.substring(15, 18);
String mm = strDate.substring(19, 22);
```

```
//Count day of week
int EEE_n = 0;
//요일을 숫자 형식으로 변환하다.
if(EEE.equals("Su")) {
	EEE_n = 0;
}else if(EEE.equals("Mo")) {
	EEE_n = 1;
}else if(EEE.equals("Tu")) {
	EEE_n = 2;
```

```
// 요일을 문자열 형식으로 다시 변환한다.
String EE[] = new String[10];

if(setEEE[0] == 0) {
	EE[0] = "Su";
}else if(setEEE[0] == 1) {
	EE[0] = "Mo";
}else if(setEEE[0] == 2) {
	EE[0] = "Tu";
```

```
}else if(EEE.equals("We")) {
        EEE_n = 3;
}else if(EEE.equals("Th")) {
        EEE_n = 4;
}else if(EEE.equals("Fr")) {
        EEE_n = 5;
}else if(EEE.equals("Sa")) {
        EEE_n = 6;
}else {
        System.out.println("fail");
}
// 배열을 선언하여 요일을 로테이션한다
int[] setEEE = new int[20];
setEEE[0] = EEE_n;

for(int i = 0; i<=6; i++){

        if(i + EEE_n >= 7){

                for(int j=0; j<=6; j++){
                        setEEE[i++] = j;
                }

        }else{

                setEEE[i] = i + EEE_n;
        }
}


// 30일로 끝나는 달과 31일로 끝나는 달 분류
if((month == 1)||(month ==3 )||(month ==
5)||(month ==
7)||(month==8)||(month==10)||(month==12))
{

for(int i = 0; i<=5; i++) {          if(i
+ today >= 32) {

        for(int j=0; j<=5; j++) {
           af[i++] = j + 1;
        }

        }else{
          af[i] = i + today;
        }
    }

} else if ((month == 4)||(month == 6)
||(month == 9)||(month == 11)){

for(int i = 0; i<=5; i++){

    if(i + today >= 31){

      for(int j=0; j<=5; j++){
          af[i++] = j + 1;

        }
```

```
}else if(setEEE[0] == 3) {
        EE[0] = "We";
}else if(setEEE[0] == 4) {
        EE[0] = "Th";
}else if(setEEE[0] == 5) {
        EE[0] = "Fr";
}else if(setEEE[0] == 6) {
        EE[0] = "Sa";
}else {
        System.out.println("fail");
}
                        .
                        .
                        .


        System.out.println(EE[0]+"요일");
```

today의 값이 날짜와 요일의 기준이 되고
윤달까지 정확하게 계산하여 날짜와 요일을
매칭시킨다

```
//convert into Integer type from String type
int year = Integer.parseInt(yyyy);
int month = Integer.parseInt(MM);
int today = Integer.parseInt(dd);

int af[] = new int[31];
af[0] = today;
  }else{
    af[i] = i + today;
  }
 }
}
}
//스케줄러의 시간 구하기
int[] M_array = new int[1000];
int m_interval = 30;


/* 시간의 간격을 조절하는 변수이다
캘린더를 작성하는 페이지에서 참조된다. */

int h_str = 9; //start hour
int m_str = 20; // start min
int h = 0;
int m = 0;
int rn = 125;

//reservation number
String pmam = "AM";
String[] hours = new String[140];

// array hours
String[] minutes = new String[150];

// array minutes
```

```
      } else {
          af[i] = i + today;
        }
    }
}else if(month == 2){

// 2월 예외처리 윤달 계산
if((year % 4 == 0 && year % 100 !=
0)||(year % 400 == 0)) {

    for(int i = 0; i<=5; i++){

      if(i + today >= 30){

        for(int j=0; j<=5; j++){
         af[i++] = j + 1;

        }

      }else{
        af[i] = i + today;
      }
    }
}else {
    for(int i = 0; i<=5; i++){
      if(i + today >= 29){
        for(int j=0; j<=5; j++){

          af[i++] = j + 1;
}
```

```
//input
for(int i=0; i<=rn; i++) {
  M_array[i] = i * m_interval + m_str;
// increase minute
}
//output
for(int i=0; i<=rn; i++) {

    h = M_array[i] / 60 + h_str; //hours
    m = M_array[i] % 60; //minutes

    if( h >= 13) {
      h = h - 12;
// Change 12 hour system
//pmam = "PM"; // Change AM --> PM
    pmam = "PM";
}

    // attach '0'flag
    String H = String.format("%02d",h);
    String M = String.format("%02d", m);

    hours[i] = H;
    minutes[i] = M;
    }
  }
}
```

예약시간 리스트 구현

```
// 스케줄러에서 시간의 범위를 계산할 때 입력해야 하는 초기값의 속성들을 변수로 선언한다
var time_array = new Array(); // AM/PM hh:mm
var m_array = new Array();   // mm
var m_interval = 0;   //term

var h = 0;
var m = 0;
var rn = 0;    // Amount
var pmam = 'AM';      // PM or AM
var h_str = 0; // initial hour value
var m_str = 0; //initial minute value
var hours = new Array();
var minutes = new Array();

// 분(m) 단위를 기준으로 일정한 간격(step)으로 증감시키고 60 을나누어 시간단위를 구하고 오전/오후(AM/PM)를
구분한다.
function schedule_time() {

       rn = parseInt($('.rn').val());
       step = parseInt($('.step').val());
       m_str =  parseInt($('.mm').val());

       //m_str = $('.m_str').val()

       for(var i=0; i <= rn; i++) {
```

```
                m_array.push(i* step + m_str);
        }
                //input
                for(var i=0; i <= rn-1; i++) {

                        h_str = parseInt($('.hh').val());
                        h = parseInt(m_array[i] / 60 + h_str);
                        m = m_array[i] % 60;

                        if(h >= 13) {
                           h = h - 12;
                        }

                        if(h >= 12) {
                           pmam = 'PM';
                        }else{

                        }

                        var H = pad(h, 2);
                        var M = pad(m, 2);

                        hours[i] = H;
                        minutes[i] = M;

                //output
                $('.record'+i).val(pmam+' '+hours[i] +':'+ minutes[i]);

                }
            }

function pad(n, width) {
          n = n + '';
          return n.length >= width ? n : new Array(width-n.length+1).join('0') + n;
}
```

/* JavaScript 에서 작성한 스케줄 로직의 각 속성(시, 분, 간격)을 조작하기 위하여 폼을 구성하고 Insert 버튼을 누르면 시간 리스트가 출력되며 셀을 선택하면 해당 시간의 정보가 서버에 전송된다. 서버는 예약가능 시간(Calendar) DB 에 입력한 데이터를 저장한다. */

```
<body onload="schedule_time();">
<form id="schedule" action="schedule.do" name="schedule" method="post">
    <div class="container-fluid" >
      <div class="row">
        <div class="col-xs-12">
            <input type="text" name="calendarTable" id="calendarTable"
                   class="calendarTable col-xs-2 qr"
                   value="Table No.<%=request.getParameter("calendarTable")%>">

            <input type="text" name="calendarDay" id="day" class="calendarTable col-xs-
                   2" value="<%=request.getParameter("re_day")%>"
            <input type="text" class="hh col-xs-1" value="" placeholder="Hour"
            <input type="text" class="mm col-xs-2" value="" placeholder="Minute"
            <input type="text" class="step col-xs-1" value="" placeholder="step"
            <input type="text" class="rn col-xs-1" value="" placeholder="Amount"

            <button type="button" class="col-xs-2 qr btn btn-default"
                   onclick="javascript:schedule_time();">Input
            </button>
        </div>
```

```
        <div class="col-xs-12" >
           <div>
              <c:forEach var="i" begin="0" end="27" varStatus="st">

                 <input class="record${st.index } record calendarTime"
                        name="calendarTime" type="submit" value=""/>
           </c:forEach>
           </div>
        </div>
      </div>
    </div>
</form>
</body>
```

- 클라이언트 측에서 초기화될 때마다 서버단에서는 DB에 저장된 예약 가능시간 데이터들을 가져와서 리스트에 출력한다

	CALENDAR_TABLE	CALENDAR_DAY	CALENDAR_TIME
1	Table No.5	today	PM 12:00
2	Table No.6	today	AM 09:30
3	Table No.6	today	AM 10:00
4	Table No.6	today	AM 10:30
5	Table No.6	today	AM 11:00

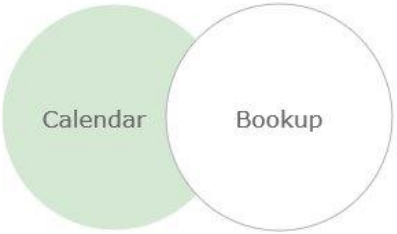
클라이언트 측의 TimePicker 초기화

- TimePicker는 서버 사이드의 스케줄러 페이지에서 전송한 데이터 리스트를 출력하는 클라이언트 사이드의 예약 가능시간 리스트이다.
- TimePicker 리스트와 Bookup 테이블의 특성상 예약시간의 중복을 허용하지 않기 때문에 모든 [예약 가능시간 집합]에서 이미 [예약 완료된 시간 집합]을 제외시켜야 한다.
- TimePicker는 음식점 내부의 테이블 별로 각각 구현되어야 한다.

- 출력 **DML**

  - 관계데이터연산의 일반 집합연산자 중 차집합(-)을 연산을 활용하여 [예약 가능시간 튜플]에서 이미 [예약 완료된 튜플]을 제외시키고 현재 남아있는 레코드들을 출력한다.

  - 달력 테이블의 시간 도메인과 예약 테이블의 시간 도메인을 'LEFT JOIN'한다




```
SELECT * FROM calendar WHERE
calendar_table = 'Table No.1' and
calendar_day='today' and calendar_time NOT
IN (SELECT DISTINCT reserved_time FROM
bookup WHERE t_index = 'Table No.1') ORDER
BY calendar_time ASC;
```

- 각 테이블의 각 날짜별로 TimePicker를 시간단위로 출력한다. 디스플레이는 none으로 초기화하고 고객이 테이블과 날짜를 선택하면 Block으로 전환시킨다.
- JSTL 반복문을 실행하여 List를 출력한다.

```
<!-- Calendar for Table No.1 ~ 9 -->
<div class="table_1" style="display:none;">
<!--     TimePicker No.0 ~ 5    -->
    <div align="left"  class="tp timepicker0">

        <c:forEach var="j" begin="0" items="${t1_today_calendar }"
                   end="15" varStatus="st">

                <input type="button" class="time time_nomal time-sh"
                       value="${j.calendarTime} "
                       onclick="time('${j.calendarTime}')" />

        </c:forEach>
    </div>
</div>
```

- 고객이 예약시간 리스트의 셀을 선택하면 예약시간이 브라우저상에 임시적으로 저장되지만 브라우저가 초기화되면 선택한 정보를 잃는다. 임시정보는 hidden 타입의 입력폼에 저장되어있다가 결제단계 트랜잭션 단계에서 서버로 전송할 때 인자 값으로 활용된다. 값이 NULL이면 트랜잭션에 실패하고 Rollback된다.

H 위치기반 기술 구현

· 음식점으로부터 고객 간의 거리 계산 기능(script/calcDistance.js)

- Geolocation API를 활용하여 사용자의 방위각을 추출하여 좌표와 소요거리를 측정한 값을 반환하는 메소드 작성

```
function getDistance() {
      var startPos;
      if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(function(position) {
          startPos = position;
```

- 위도(latitude)와 경도(longtitude) 추출하는 작업

```
document.getElementById("startLat").innerHTML
= startPos.coords.latitude;
        document.getElementById("startLon").innerHTML
        = startPos.coords.longitude;
        document.getElementById("a").value = startPos.coords.latitude;
        document.getElementById("b").value = startPos.coords.longitude;
      }, function(error) {
        alert("Error occurred. Error code: "+error.code);
      });

      navigator.geolocation.watchPosition(function(position) {
```

- 음식점의 경로 설정하는 작업

```
        var calc =  calculateDistance(startPos.coords.latitude,
                    startPos.coords.longitude,.508286743400000,
                    126.73514674630000);
        calc = calc.toString();
        document.getElementById("currentLat").innerHTML
          = 37.508286743400000;
        document.getElementById("currentLon").innerHTML
```

```
          = 126.73514674630000;
        calc = calc.substr(0,7);

            document.getElementById("distance").value = calc;
            document.getElementById("dist").value = calc;
          });
        }
      };
```

- 음식점으로부터 고객 사이의 거리 계산하는 작업

```
      function calculateDistance(lat1, lon1, lat2, lon2) {
        var R = 6371; // km
        var dLat = (lat2-lat1).toRad();
        var dLon = (lon2-lon1).toRad();
        var a = Math.sin(dLat/2) * Math.sin(dLat/2) +
                Math.cos(lat1.toRad()) * Math.cos(lat2.toRad()) *
                Math.sin(dLon/2) * Math.sin(dLon/2);
        var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
        var d = R * c;
        return d;
      }
      Number.prototype.toRad = function() {
        return this * Math.PI / 180;
      }

      function distance() {
             alert($('.distance').val());
             $('.dist').val($('.distance').val());
      }
```

- 로그인시 위치정보의 활용(member/loginOk.jsp)
  - 매번 로그인시마다 고객의 현재 위치정보를 고객 DB에 업데이트 시킨다.

```
dao.updateDistance(dist, email);
```

  - 고객 DB에서 가져온 사용자의 위치정보를 세션으로 처리한다(세션은 로그아웃시 종료한다).

```
String userDistance = null;
userDistance = dao.confirmDistance(dto.getCustomerEmail())
               .getCustomerDistanceInfo();
session.setAttribute("userDistance", userDistance);
```

- 예약 주문 결제시 위치정보의 활용(reservation/bookupOk.jsp)
  - 위치 정보 기반의 인증키를 보조인증 수단으로 사용한다. 위치정보 기반이라 함은 사용자의 단말기의 HTML의 GeoLocation으로 실시간 추출하는 가변적인 방위각과 음식점의 고정된 방위각 사이의 거리를 계산한 측정 수치를 Km단위로 환산한 결과 값이다.
  - 결제 시점의 위치정보의 값과, 세션에서 가져온 값(로그인 시 초기화된 위치정보)과 고객 DB에서 가져온 위치정보 값을 비교하여 정당한 사용자인지 여부를 검사한다.

```
// 결제 시점의 위치정보
String payerdistance = request.getParameter("distance");
// 서버단의 위치정보
sessionDistance = (String)session.getAttribute("userDistance");
// DB상의 위치정보
serverDistance =
dao.confirmDistance(dto.getCustomerEmail()).getCustomerDistanceInfo();
```

- 위치정보는 고유한 정보이므로 올바른 사용자가 접속한다면 아래 세 가지 결과 값이 일치하게 된다. 인증이 성공했다는 신호를 보낸 후 결제를 시도한다.

```
1월 29, 2019 11:44:21 오후 org.apache.catalina.core.ApplicationContext log
INFO: jsp: payerdistance: 28.9071
1월 29, 2019 11:44:21 오후 org.apache.catalina.core.ApplicationContext log
INFO: jsp: sessionDistance: 28.9071
1월 29, 2019 11:44:21 오후 org.apache.catalina.core.ApplicationContext log
INFO: jsp: serverDistance: 28.9071
```

- 결과 값이 일치하지 않은 경우이다. 경고창을 띄우고 메인 페이지로 돌아간다.

```
1월 29, 2019 11:42:36 오후 org.apache.catalina.core.ApplicationContext log
INFO: jsp: payerdistance: 28.9067
dispatcher : org.apache.catalina.core.ApplicationDispatcher@4cb8343c
1월 29, 2019 11:42:36 오후 org.apache.catalina.core.ApplicationContext log
INFO: jsp: sessionDistance: 28.9211
1월 29, 2019 11:42:36 오후 org.apache.catalina.core.ApplicationContext log
INFO: jsp: serverDistance: 28.9098
```

I 결제 트랜젝션

· 인증 프로세스(reservation/bookupOk.jsp)

- 사용자 고유의 위치정보를 조회하여 일치여부를 판독한 결과에 따라 처리한다.
- 값이 일치하지 않은 경우 인증에 실패하였다는 알림창을 띄우고 종료한다.

```
if (!payerdistance.equals(sessionDistance) || !payerdistance.equals(serverDistance)) {
    %>
        <script type="text/javascript">
                alert('Fail : Compared your location information'
                            + ' with location information in server,'
                            + 'As a result did not match!');
                history.go(-1);
        </script>
    <%
```

- 값이 일치하는 경우 결제 페이지에서 입력한 값을 넘겨받아 아이디를 인자값으로 DAO에서 거래소에 해당하는 DB 내에 있는 신용 정보를 조회하고 리턴한 값에 따라 처리한다.

```
}else if (payerdistance.equals(sessionDistance)
        && payerdistance.equals(serverDistance)) {
                // confirm Credit information
                int ri = dao.confirmCdn(cdn);
                ri += dao.confirmInval(inval);
                ri += dao.confirmCvc(cvc);
        if( ri != 3) {
        %>
        <script language="javascript">
                alert("Fail : Not exist credit card informaion.");
                history.go(-1);
        </script>
        <%
```

- 거래소 DB내의 카드번호 일치여부를 조회하기 위해 DAO에서 쿼리를 수행하는 작업이다.

```
 public int confirmCdn(String cdn) {
        int ri = 0;
        Connection conn = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;
```

```
		String query = "SELECT credit_cdn FROM bananabank "
				+ "WHERE credit_cdn = ?";
		try {
			conn = getConnection();
			pstmt = conn.prepareStatement(query);
			pstmt.setString(1, cdn);

			rs = pstmt.executeQuery();

			if(rs.next()) {
				ri = Dao.CDN_EXISTENT;
			}else {
				ri = CDN_NONEXISTENT;
			}

		} catch (Exception e) {
			e.printStackTrace();
		}finally {
			JdbcUtil.close(rs);
			JdbcUtil.close(pstmt);
			JdbcUtil.close(conn);
		}
		return ri;
	}
```

- 위치정보와 신용정보 인증과정이 완료되면 인자 값을 DAO의 bookup()에 전송하고 예약과 결제를 수행하여 반환 된 값에 따라 처리한다.

```
		@SuppressWarnings("resource")
		public int bookup(String t_index, String email, String
				menuname, String withdraw, String distance, String
				reserved_wmy, String reserved_day, String
				reserved_time, String creditCardFirm, String cdn,
				String customer_name, String inval, String cvc,
				String phone, String installment, String p_withdraw)
				{
		int ri = 0;
		Connection conn = null;
		PreparedStatement pstmt = null;

		String query1 = "INSERT INTO bookup VALUES(sysdate,?,?,?,?,?,?,?,?)";

		String query2 = "INSERT INTO creditcardcompany VALUES(sysdate, ?, ?,"
				+ " ?, ?, ?, ?, ?, ?, ?)";
		try {
			conn = getConnection();

			/* Transaction
			 * Change autoCommit function manually. */

			conn.setAutoCommit(false);
			System.out.println("getConnection");
			pstmt = conn.prepareStatement(query1);
			System.out.println("prepareStatement for query values");
			pstmt.setString(1, t_index);
				.
				.
				.

			ri = pstmt.executeUpdate();
			pstmt = conn.prepareStatement(query2);
```

```
                pstmt.setString(1, creditCardFirm);
                        .
                        .
                        .

                ri += pstmt.executeUpdate();
                System.out.println("ri ::" + ri);

                // It can only be commit when all the work is completed
                without any error.
                if (ri != 2) {
                        try {
                                conn.rollback();
                                conn.setAutoCommit(true);

                        } catch (SQLException sqle) {
                                sqle.printStackTrace();
                        }

                }else {
                        conn.commit();
                }

        } catch (Exception e) {
                e.printStackTrace();

        }finally {
                        JdbcUtil.rollback(conn);
                        JdbcUtil.close(pstmt);
                        JdbcUtil.close(conn);
        }

        return ri;
    }
```

- 트랜잭션에 성공하면 정수 2를 반환하고 결제완료 팝업창이 열린다. 트랜잭션에 실패하면 롤백 처리하고 이전페이지로 돌아간다.

```
<%
        ri = 0;
        ri = dao.bookup(t_index, email, menuname, withdraw, payerdistance,
              reserved_wmy, reserved_day, reserved_time, creditCardFirm,
              cdn,  customer_name, inval, cvc, phone, installment, withdraw);

        if(ri == 2){
%>
    <script type="text/javascript">
        alert("completed bookup.");
        history.go(-1);

        var popUrl = "payment/complet_payment.jsp";
        var popOption;
        window.open(popUrl, "", popOption);
    </script>
<%
        }else{
%>
     <script type="text/javascript">
        alert("Failed bookup.");
        history.go(-1);
     </script>
```