

# JSP Servlet프로젝트

## - 음식 주문 태블릿

홍한별

---

# 목차

1

프로젝트 계획 및 설명

2

프로젝트의 전체적인 흐름

3

프로젝트 코드설명

4

프로젝트에 대한 리뷰

# Part 1

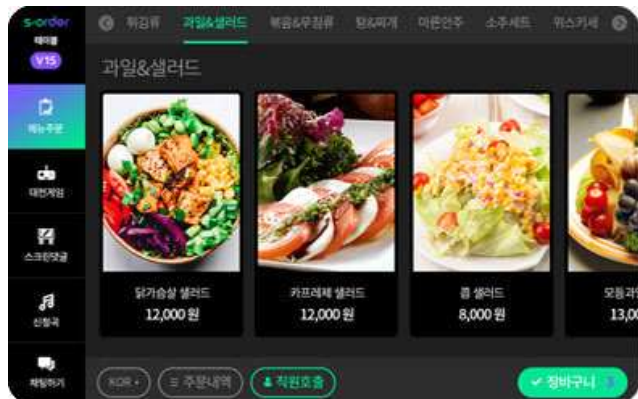
## 프로젝트 계획 및 설명




음식 주문 태블릿

+

카운터 포스기



**개발언어/환경** Eclipse IDE 4.25.0 / MySQL Workbench 8.0.13 / Java / Javascript /  
ApacheTomcat 9.0 /  
( JDBC 드라이버 : mysql-connector 8.0.13v  )

**개요** 음식점에서 활용할 수 있는 웹사이트

**구현기능** 메뉴/테이블 추가 , 삭제 , 수정 , 주문하기, 주문내역확인, 주문상태 업데이트 , 결제하기 등

## Part 1

# 프로젝트 계획 및 설명 - 개발환경(context.xml, web.xml)

```

1 <Context
2
3 <Resource name="jdbc/JSP_Project"
4           auth="Container" type="javax.sql.DataSource"
5             maxActive="20" maxIdle="5" maxWait="10000"
6             username="restaurant" password="restaurant96"
7             driverClassName="com.mysql.cj.jdbc.Driver"
8             url="jdbc:mysql://localhost:3306/jsp_project?useSSL=false&serverTimezone=UTC"/>
9
10 </Context>

```

Query 1 Administration - Users and Privileges

restaurant  
Users and Privileges


User Accounts

User	From Host
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
restaurant	%
root	localhost
webstudent	localhost

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
3   <display-name>JSP_Project</display-name>
4   <welcome-file-list>
5     <welcome-file>ProjectServlet1</welcome-file>
6   </welcome-file-list>
7
8   <error-page>
9     <error-code>404</error-code>
10    <location>/ErrorPage.jsp</location>
11  </error-page>
12  <error-page>
13    <error-code>405</error-code>
14    <location>/ErrorPage.jsp</location>
15  </error-page>
16  <error-page>
17    <error-code>406</error-code>
18    <location>/ErrorPage.jsp</location>
19  </error-page>
20  <error-page>
21    <error-code>500</error-code>
22    <location>/ErrorPage.jsp</location>
23  </error-page>
24 </web-app>

```



예기치 못한 문제로 에러가 발생했습니다.

구체적인 에러의 원인은 시스템개발팀에 보고가 되었으며,  
현재 처리 중에 있습니다.

불편함을 드려서 죄송합니다. 조속히 해결하도록 하겠습니다.

화면을 클릭하시면 첫 화면으로 돌아갑니다.

```

<script>
  document.addEventListener('click',()=>{
    window.location.href = 'ProjectServlet1; return false;';
  })
</script>
<style>
  img{
    width : 80%;
    height : 80%;
    margin : 0 auto 0 auto;
  }
  div{
    text-align : center;
    font-size : 25px;
  }
</style>
</head>
<body>
  <div>
    <br>
    화면을 클릭하시면 첫 화면으로 돌아갑니다.</div>
</body>
</html>

```

## 프로젝트 계획 및 설명 - 프로젝트 진행 Gantt Chart

절차	12/12	12/13	12/14	12/15	12/16	12/17	12/18	12/19	12/20	12/21	12/22
프로젝트 구상 ( 화면 구상 / 기능 구상 )											
기능 순서도 작성											
프론트 화면 구현											
기능 구현											
단위 테스트											
통합 테스트											
PPT 작성 및 발표 준비											



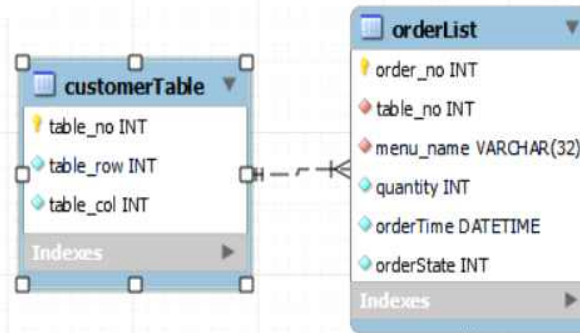




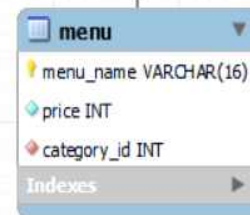
## Part 1

# 프로젝트 계획 및 설명 - DB구성

```
CREATE TABLE `jsp_project`.`customerTable` (
  `table_no` INT NOT NULL,
  `table_row` INT NOT NULL,
  `table_col` INT NOT NULL,
  PRIMARY KEY (`table_no`));
```



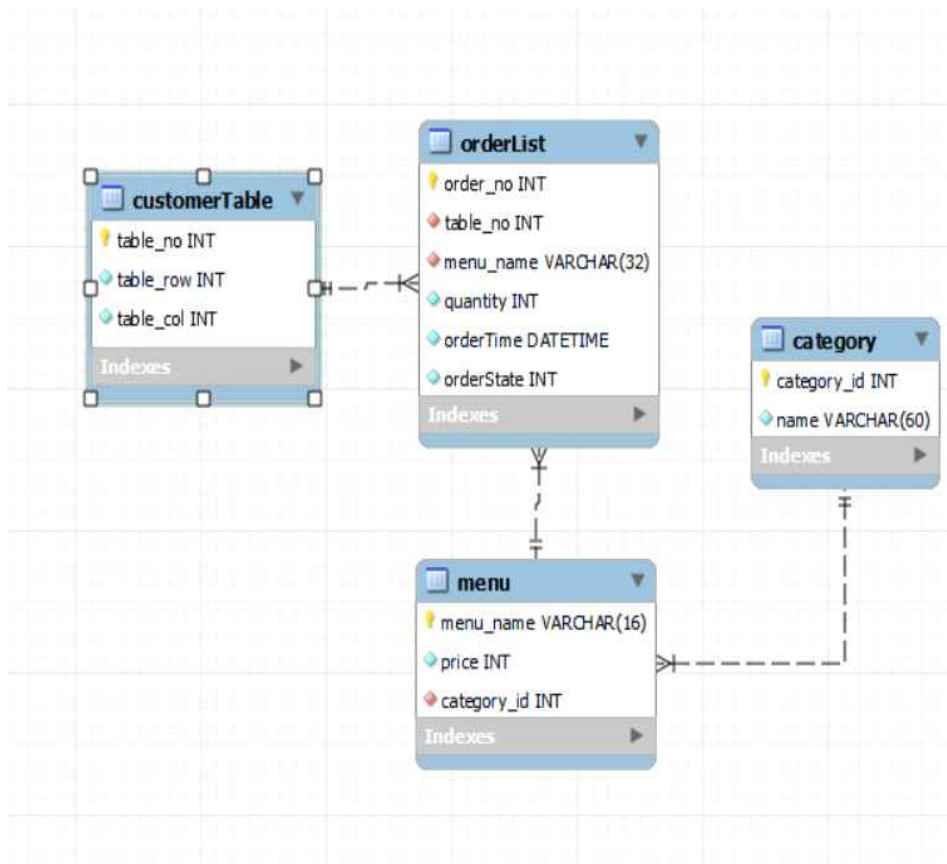
```
CREATE TABLE `jsp_project`.`menu` (
  `menu_name` VARCHAR(16) NOT NULL,
  `price` INT NOT NULL,
  `category_id` INT NOT NULL,
  PRIMARY KEY (`menu_name`),
  FOREIGN KEY (`category_id`)
  REFERENCES `jsp_project`.`category` (`category_id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION);
```



```
CREATE TABLE `jsp_project`.`orderList` (
  `order_no` INT NOT NULL DEFAULT AUTO_INCREMENT,
  `table_no` INT NOT NULL,
  `menu_name` VARCHAR(32) NOT NULL,
  `quantity` INT NOT NULL,
  `orderTime` DATETIME NOT NULL DEFAULT NOW(),
  `orderState` INT NOT NULL DEFAULT 1,
  PRIMARY KEY (`order_no`),
  CONSTRAINT `table_no`
  FOREIGN KEY (`table_no`)
  REFERENCES `jsp_project`.`customerTable` (`table_no`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
  CONSTRAINT `menu_name`
  FOREIGN KEY (`menu_name`)
  REFERENCES `jsp_project`.`menu` (`menu_name`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION);
```



```
CREATE TABLE `jsp_project`.`category` (
  `category_id` INT NOT NULL,
  `name` VARCHAR(60) NOT NULL,
  PRIMARY KEY (`category_id`));
```



### orderList 테이블의 orderState

- 1 : 주문들어온 상태
- 2 : 요리중인 상태
- 3 : 요리 완성한 상태
- 4 : 결제 완료한 상태

### customerTable

- table\_row 와 table\_col은 위치를 나타냄

1					12
주문상태 주문번호					주문상태
2	10	11	12	13	14
주문상태 주문번호	주문상태 주문번호	주문상태 주문번호	주문상태 주문번호	주문상태 주문번호	주문상태 주문번호
3	5	6	7	8	10
주문상태 주문번호	주문상태 주문번호	주문상태 주문번호	주문상태 주문번호	주문상태 주문번호	주문상태 주문번호
4					9
주문상태 주문번호	주문상태 주문번호	주문상태 주문번호	주문상태 주문번호	주문상태 주문번호	주문상태 주문번호



## Part 2

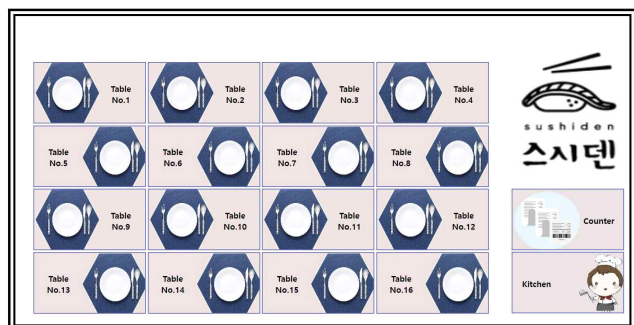
프로젝트 전체적인 흐름





## Part 2

## 프로젝트의 전체적인 흐름



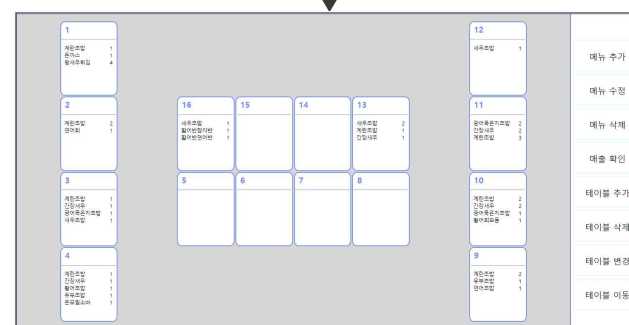
```
document.addEventListener("DOMContentLoaded", () => {
  const movePage = (event) => {
    let moveLocation;
    let targetString = event.target.textContent;
    if (targetString.includes('Table')) {
      const idx = targetString.indexOf('.') + 1;
      moveLocation = 'ProjectServlet1?command=movePage&location=Table' + targetString.substr(idx);
    } else {
      moveLocation = 'ProjectServlet1?command='+targetString+'Data';
    }
    window.location.href = moveLocation;
  }
});
```



Orders Awaiting Confirmation			
Table Number	Menu	Quantity	Button
3	계란초밥	1	Cook
3	간장새우	1	Cook
3	광어묵은지초밥	1	Cook
2	연어회	1	Cook
4	활어초밥	1	Cook
4	유부초밥	1	Cook
4	온도일소배	1	Cook
11	광어묵은지초밥	2	Cook
13	계란초밥	1	Cook
13	간장새우	1	Cook
9	계란초밥	2	Cook
9	유부초밥	1	Cook
9	연어초밥	1	Cook
16	새우초밥	1	Cook
16	활어반면어반	1	Cook

Orders Being Cooked			
Table Number	Menu	Quantity	Button
4	계란초밥	1	Complete
4	간장새우	1	Complete
3	새우초밥	1	Complete
1	계란초밥	1	Complete
1	돈까스	1	Complete
1	왕새우튀김	4	Complete
2	계란초밥	2	Complete
10	계란초밥	2	Complete
10	간장새우	2	Complete
10	광어묵은지초밥	1	Complete
10	활어회도둑	1	Complete
11	간장새우	2	Complete
11	계란초밥	3	Complete
12	새우초밥	1	Complete
13	새우초밥	2	Complete



## Part 2

# 프로젝트의 전체적인 흐름

Orders Awaiting Confirmation				Orders Being Cooked			
Table Number	Menu	Quantity	Button	Table Number	Menu	Quantity	Button
3	계란초밥	1	Cook	4	계란초밥	1	Complete
3	간장새우	1	Cook	4	간장새우	1	Complete
3	광어묵은지초밥	1	Cook	3	새우초밥	1	Complete
2	연어회	1	Cook	1	계란초밥	1	Complete
4	활어초밥	1	Cook	1	돈까스	1	Complete
4	유부초밥	1	Cook	1	왕새우튀김	4	Complete
4	온모밀소바	1	Cook	2	계란초밥	2	Complete
11	광어묵은지초밥	2	Cook	10	계란초밥	2	Complete
13	계란초밥	1	Cook	10	간장새우	2	Complete
13	간장새우	1	Cook	10	광어묵은지초밥	1	Complete
9	계란초밥	2	Cook	10	활어회모듬	1	Complete
9	유부초밥	1	Cook	11	간장새우	2	Complete
9	연어초밥	1	Cook	11	계란초밥	3	Complete
16	새우초밥	1	Cook	12	새우초밥	1	Complete
16	활어반참치반	1	Cook	13	새우초밥	2	Complete
16	활어반연어반	1	Cook				

주방에서 보는 페이지로  
왼쪽은 테이블에서 주문한 메뉴를 오른쪽에는  
만들고 있는 메뉴들을 표시해주는 페이지 입니다.

각 버튼을 클릭할 경우 이벤트가 발생합니다.

왼쪽 테이블

```
SELECT order_no,table_no,menu_name,quantity  
FROM orderlist WHERE orderState=1
```

오른쪽 테이블

```
SELECT order_no,table_no,menu_name,quantity  
FROM orderlist WHERE orderState=2
```

## Part 2

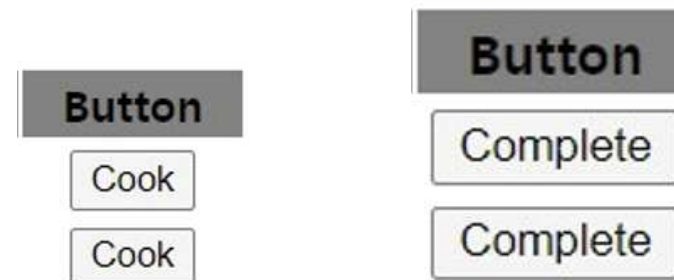
# 프로젝트의 전체적인 흐름

```
<c:forEach var="confirm" items="${orderConfirm}">
  <tr>
    <td>${confirm.tableNumber}</td>
    <td>${confirm.menuName}</td>
    <td>${confirm.orderQuantity}</td>
    <td><input type="button" value="Cook"
      onclick="window.location.href='ProjectServlet1?
      command=updateState&OrderNumber=${confirm.orderNumber}'; return false;"
      class="update-State"/></td>
  </tr>
</c:forEach>
```

```
public void updateState(int orderNumber) {
    // TODO Auto-generated method stub
    Connection conn = null;
    Statement mySt = null;

    try {
        conn = dataSource.getConnection();
        String sql = "UPDATE orderlist SET orderState = IF( (orderState = 1) "
            + "OR (orderState = 2) , orderState+1 ,orderState) WHERE order_no ="
            + orderNumber;
        mySt = conn.createStatement();

        mySt.executeUpdate(sql);
    } catch (Exception e) {
        System.out.println("updateState 실행중 에러발생");
        e.printStackTrace();
    } finally {
        close(conn, mySt, null);
    }
}
```

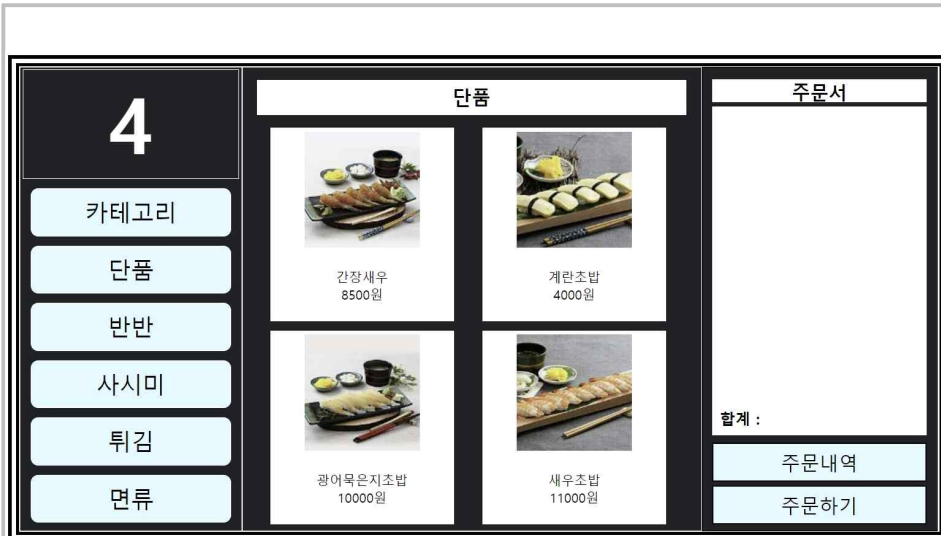


두 버튼 모두 클릭시 주문의 상태를 업데이트 하는 함수가 실행되면 어떤 주문인지 확인하기 위해서 각 Row 에 해당하는 주문의 주문번호가 변수에 담겨 같이 넘어갑니다.



## Part 2

## 프로젝트의 전체적인 흐름



각 테이블에서 보는 페이지로  
모든 메뉴에 대한 정보가 넘어와 카테고리 별로  
Div 에 작성되어 있지만 1개의 카테고리의 Div만  
보이도록 작성했습니다.



왼쪽 카테고리 부분을 클릭하면 보여지던 Div가  
사라지고 클릭한 카테고리의 Div가 보이도록  
스크립트 부분을 설정했습니다.

## Part 2

## 프로젝트의 전체적인 흐름

4

카테고리

단품


반반

사시미


튀김

면류


사시미




연어회  
33000원



특모듬회  
60000원



활어회모듬  
35000원



활어회모듬L  
55000원

주문서

연어회

활어회모듬L

활어회모듬

특모듬회

합계 : 405000

주문내역

주문하기

주문서와 합계 경우 따로 DB를 만들어 운영하지 않고 해당 페이지에서 자바스크립트를 이용해서 만들었습니다.

4

카테고리

단품


반반

사시미


튀김

면류


사시미




연어회  
33000원



특모듬회  
60000원



활어회모듬  
35000원



활어회모듬L  
55000원

주문서

연어회

활어회모듬L

활어회모듬

특모듬회

합계 : 405000

주문내역

주문하기

주문하기 버튼의 경우 주문서에 저장된 데이터들을 메뉴명 과 양 두개의 String 으로 저장하여 Servlet으로 전달하여 주문내역 테이블에 Insert 해주었습니다.

## Part 2

# 프로젝트의 전체적인 흐름

```
const orderButtonClick = () =>{  
  
  if (document.querySelectorAll('.totalOrderDate').length == 0) return;  
  $totalOrderDateDiv = document.querySelector('.totalOrderDate');  
  
  let menuNames = '';  
  let menuQuantity = '';  
  
  for(let i = 0 ; i < $totalOrderDateDiv.length ; i ++){  
  
    if(menuNames){  
  
      menuNames = menuNames + ',';  
      menuQuantity = menuQuantity + ',';  
    }  
    menuNames = menuNames + $totalOrderDateDiv[i].childNodes[1].textContent;  
    menuQuantity = menuQuantity + $totalOrderDateDiv[i].childNodes[3].childNodes[3].textContent  
  
  }  
  
  const tableNumber = parseInt(document.querySelector('#tableNumberDiv').textContent);  
  
  console.log(menuNames + ' , ' + menuQuantity);  
  
  window.location.href='ProjectServlet1?command=Order&tableNum='+tableNumber+'&orderMenuNames='+  
+menuNames+'&orderMenuQuantity='+menuQuantity;  
  
}
```

주문서	
연어회	+ 5 - x
참어회모듬L	+ 2 - x
참어회모듬	+ 2 - x
특모듬회	+ 1 - x
합계 : 405000	
주문내역	
주문하기	

주문하기 버튼을 클릭할 경우 왼쪽의 함수가 실행되도록 되어 있는데

왼쪽의 함수는 주문서에 포함되어 있는 메뉴들과 양들을 각각 menuNames와 menuQuantity 두개의 변수에 ,를 구분자로 이용해서 하나의 String으로 만들고

이동할때 파라미터의 값으로 두개의 String을 넘겨주어 처리하도록 설정했습니다.

## Part 2

# 프로젝트의 전체적인 흐름



주문 내역의 경우 팝업창으로 열리게 했으며  
주문 확인전에만 주문이 취소 가능하게 만들었습니다.

## View -> Controller

```
<input type="checkbox" name="deleteMenus" value='${order.menuName},${order.orderTime}' />  
<input type="hidden" name="tableNumber" value='${TableNumber}' />
```

## Controller -> Model

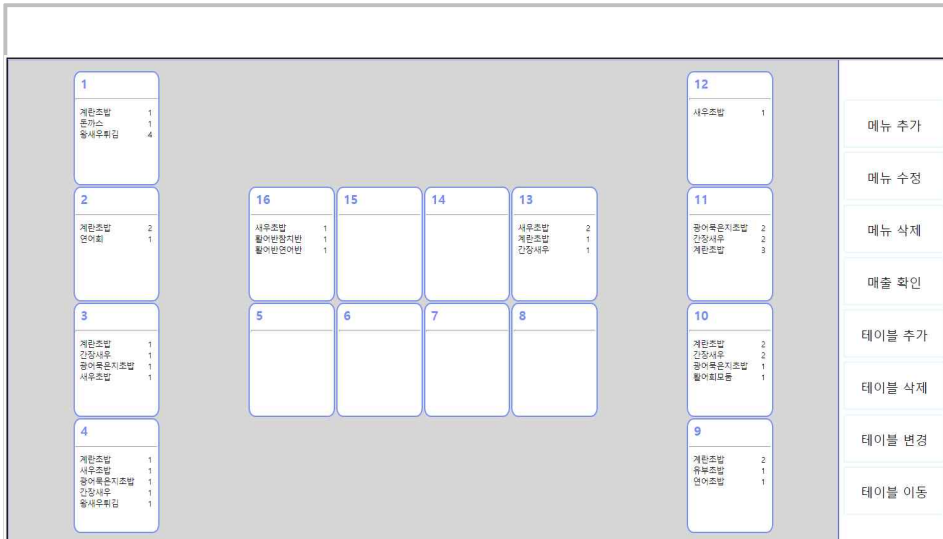
```
String[] deleteMenuNames = request.getParameterValues("deleteMenus");  
int tableNumber= Integer.parseInt(request.getParameter("tableNumber"));  
ArrayList<String> deleteTimes = new ArrayList<>();  
ArrayList<String> deleteMenus = new ArrayList<>();  
for(String a : deleteMenuNames) {  
    deleteMenus.add(a.split(",")[0]);  
    deleteTimes.add(a.split(",")[1]);  
}  
  
dataUtil.deleteOrders(tableNumber,deleteMenus,deleteTimes);
```

## Model -> Controller

```
public void deleteOrders(int tableNumber, ArrayList<String> deleteMenus, ArrayList<String> deleteTimes) {  
    // TODO Auto-generated method stub  
    Connection conn = null;  
    PreparedStatement mySt = null;  
    try {  
        conn = dataSource.getConnection();  
        StringBuffer sql = new StringBuffer("delete from orderlist where table_no = ? AND menu_name = ? AND orderTime=?; ");  
        mySt = conn.prepareStatement(sql.toString());  
  
        for (int i = 0; i < deleteMenus.size(); i++) {  
            mySt.setInt(1, tableNumber);  
            mySt.setString(2, deleteMenus.get(i));  
            mySt.setString(3, deleteTimes.get(i));  
            mySt.addBatch();  
            mySt.clearParameters();  
        }  
        mySt.executeBatch();  
    } catch (SQLException e) {  
        System.out.println("deleteOrders 함수 실행중 예외발생");  
        e.printStackTrace();  
    } finally {  
        close(conn, mySt, null);  
    }  
}
```

## Part 2

## 프로젝트의 전체적인 흐름



카운터 페이지의 경우 주문 내역을 가져올 때 테이블에 동일한 메뉴의 주문이 있는 경우 합쳐서 가져오도록 Util에 함수를 하나 더 만들어 실행해서 가져왔습니다.

```
int[][] tableLocation = dataUtil.TableData();
request.setAttribute("Location", tableLocation);
ArrayList<Order> Order = dataUtil.OrderData();
ArrayList<Order> OrderMerge = dataUtil.OrderDataMerge(Order);
request.setAttribute("Order", OrderMerge);
```

```
public ArrayList<Order> OrderDataMerge(ArrayList<Order> order) {
    // TODO Auto-generated method stub

    ArrayList<Order> orderDatas = new ArrayList<>();

    for(Order a : order) {

        int resultIndex = orderDatas.indexOf(a);
        if(resultIndex>=0) {

            orderDatas.get(resultIndex).setOrderQuantity(orderDatas.get(resultIndex)
                .getOrderQuantity()+a.getOrderQuantity());

        }else {

            orderDatas.add(a);

        }

    }

    return orderDatas;
}
```

```
@Override
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    Order other = (Order) obj;
    return Objects.equals(menuName, other.menuName) && tableNumber == other.tableNumber;
}
```

## Part 2

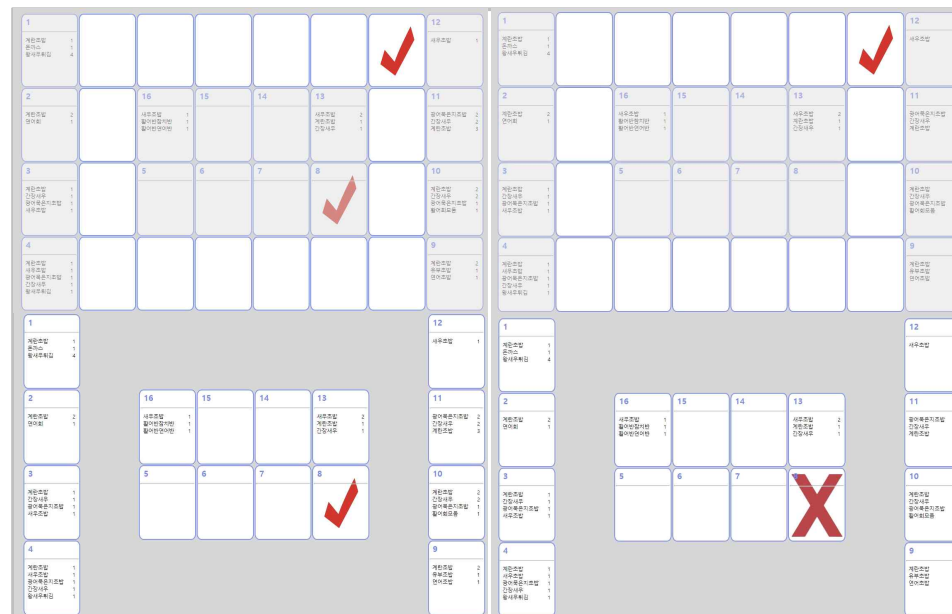
# 프로젝트의 전체적인 흐름

### < 테이블 변경의 유효성 검사 및 입력 취소시 >

```
let $DataTd = document.querySelectorAll(".DataTd");
let tableNumbers = new Array();
for(let i = 0 ; i < $DataTd.length ; i++){
    tableNumbers.push(Number($DataTd[i].childNodes[1].textContent.replace('\n', '').trim()));
}
```

```
$('.DataTd').click(function(event){
    let tableNum
    while(true){
        tableNum = Number(prompt('바꾸실 테이블 번호를 입력하세요'));

        if(!Number.isInteger(tableNum)){
            alert('정수가 아닙니다. 다시입력해주세요');
            continue;
        }
        if(tableNumbers.includes(tableNum)) {
            alert('이미 존재하는 테이블 번호입니다 다시입력해주세요');
            continue;
        }
        if(tableNum>32){
            alert('테이블 번호는 32를 넘을 수 없습니다. 다시입력해주세요');
            continue;
        }
        break;
    }
    if(tableNum == 0){
        IngFunction = '';
        $('.DataTd').off('click');
        $('.DataTd').off('mouseenter');
        $('.DataTd').off('mouseleave');
        $(this).css('background-image', '');
        return;
    }
})
```

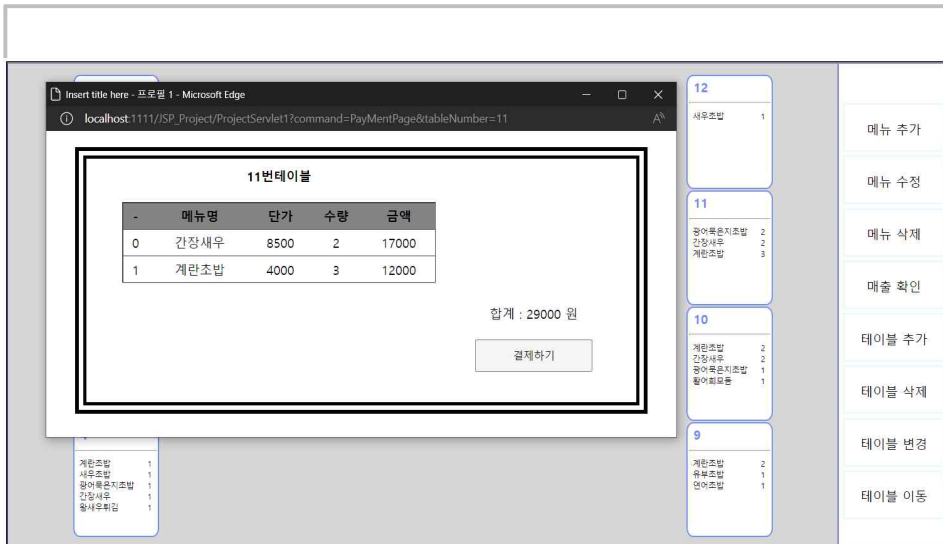


테이블 추가 삭제 변경 이동의 경우 자바스크립트를 이용해서 위와 같이 표시할 수 있도록 작성하고 서블릿으로 넘어갈 때 테이블의 행과 열정보와 추가적인 정보들을 넘겨주었습니다.

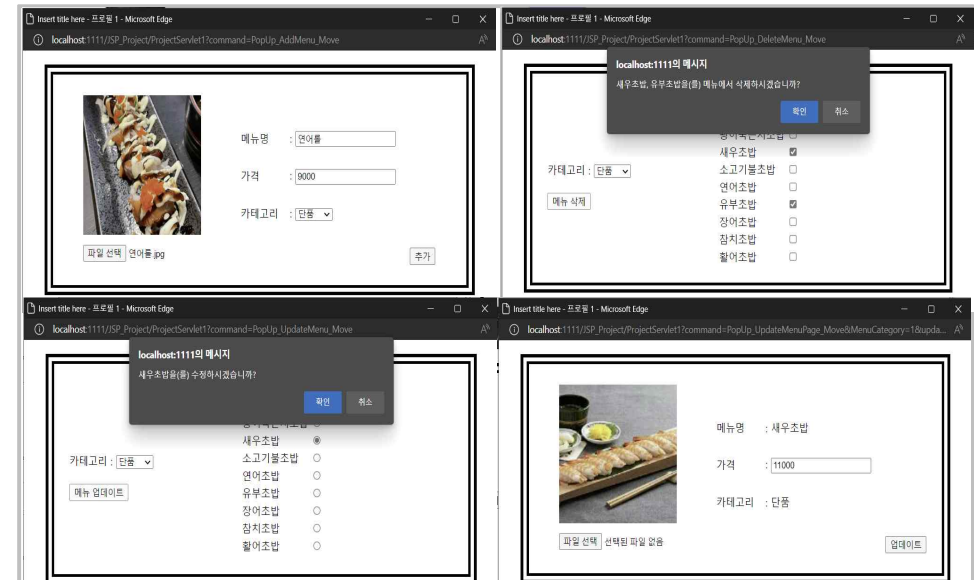


## Part 2

# 프로젝트의 전체적인 흐름



각 테이블 Div를 클릭시에는 확인전인 주문과 테이블에서 결제완료된 주문을 제외한 나머지 주문만을 보여주고 결제하기를 누를시에 테이블에 존재하는 주문들의 상태를 업데이트 해줍니다.



메뉴 추가 수정 삭제또한 팝업창으로 열리며 입력하거나 선택한 데이터들에 대하여 insert Update Delete 가 이루어집니다.

## Part 2

# 프로젝트의 전체적인 흐름

Insert title here - 프로필 1 - Microsoft Edge

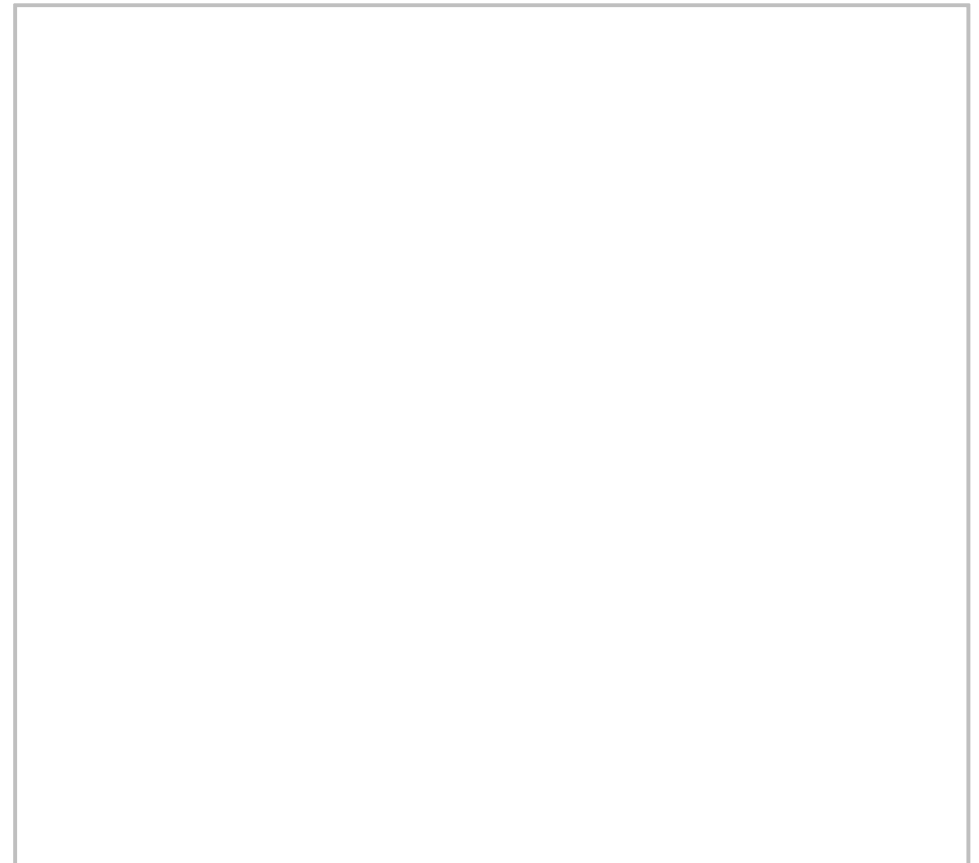
localhost:1111/JSP\_Project/ProjectServlet?command=PopUp\_BetweenDateCashUp&startDate=2022-12-13&end...

2022-12-13 ~ 2022-12-14 기간 선택

합계 : 1644900 원

-	메뉴명	단가	수량	금액
1	활어회모듬L	55000	5	275000
2	연어회	33000	7	231000
3	왕새우튀김	20000	8	160000
4	참치초밥	18000	6	108000
5	소고기불초밥	15000	7	105000
6	장어초밥	18000	5	90000
7	장어반스테이크반	17500	5	87500
8	활어반연어반	17500	5	87500
9	모듬고로케	12000	7	84000

매출 확인은 테이블 번호와 상관없이 해당 날짜의 주문 내역들을 모두 가져와주는데 이때 추가적으로 동일한 메뉴를 하나로 합쳐주고 내림차순으로 정렬해서 가져와 줍니다.





# Part 3

## 프로젝트 코드 설명



## Part 3

# 프로젝트 코드 설명 - 객체 비교

```
static <T extends Comparable<? super T>>  
void  
  
static <T> void  
  
sort(List<T> list)  
Sorts the specified list into ascending order, according to the natural  
ordering of its elements.  
  
sort(List<T> list, Comparator<? super T> c)  
Sorts the specified list according to the order induced by the specified  
comparator.  
  
@Override  
public int compareTo(CashUpData cashUpData) {  
    // TODO Auto-generated method stub  
    if (this.menu_price*this.menu_quantity < cashUpData.menu_price*cashUpData.menu_quantity) {  
        return -1;  
    } else if (this.menu_price*this.menu_quantity == cashUpData.menu_price*cashUpData.menu_quantity) {  
        return 0;  
    } else {  
        return 1;  
    }  
}
```

Collections 클래스의 sort를 사용하기 위해서는 Comparable 인터페이스를 구현한 클래스 이거나 Comparator 클래스의 객체가 필요하게 되는데

이번 프로젝트에서는 Comparable 인터페이스의 추상 메서드인 compareTo를 오버라이딩 해서 사용했습니다.

```
Collections.sort(cashUpDatasDataMerge,Collections.reverseOrder());
```

```
@Override  
public int compareTo(CashUpData cashUpData) {  
    // TODO Auto-generated method stub  
    if (this.menu_price*this.menu_quantity < cashUpData.menu_price*cashUpData.menu_quantity) {  
        return -1;  
    } else if (this.menu_price*this.menu_quantity == cashUpData.menu_price*cashUpData.menu_quantity) {  
        return 0;  
    } else {  
        return 1;  
    }  
}
```

내림차순으로 정리하기 위해서 compareTo의 return 값을 반대로 작성하는 방법도 있었지만 sort 함수에 Collections.reverseOrder()를 매개로 같이 넣어서 내림차순으로 정렬하여 사용했습니다.

```
public void order(int tableNumber, String[] menuNames, String[] menuQuantity) {  
    // TODO Auto-generated method stub  
  
    Connection conn = null;  
    PreparedStatement mySt = null;  
    try {  
        conn = dataSource.getConnection();  
        StringBuffer sql = new StringBuffer(" INSERT INTO orderlist(table_no,menu_name,quantity) VALUES(?,?,?)");  
        mySt = conn.prepareStatement(sql.toString());  
  
        for (int i = 0; i < menuNames.length; i++) {  
            mySt.setInt(1, tableNumber);  
            mySt.setString(2, menuNames[i]);  
            mySt.setString(3, menuQuantity[i]);  
            mySt.addBatch();  
            mySt.clearParameters();  
        }  
        mySt.executeBatch();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    } finally {  
        close(conn, mySt, null);  
    }  
}
```

함수안에서 여러번의 sql문이 실행되는 경우 반복문과 preparedStatement의 batch 기능을 이용해서 함수를 만들었습니다.

하나의 sql문을 실행시켜주는 함수를 반복문을 통해 실행시켜 주게되면 Connection을 끊고 연결해야 되기 때문에 효율적이지 못하다.

객체에 다시 할당해서 사용할 경우 close 시 마지막 preparedStatement만 닫히게 된다.

객체는 한번만 할당하고 .clearParameters()를 통해 파라미터를 초기화 하고 다시 할당해서 사용하게 되면 여러번 사용 가능하다.

하지만 PreparedStatement 의 batch 기능을 이용하게 되면 매번 쿼리 실행 시 소비되는 리소스, 쿼리 실행 후 결과를 받는 자바 쪽 단계 없이 한번에 여러건을 실행하므로 속도가 향상되는 장점이 있다. 하지만 배치에 저장되는데 한계가 있으므로 일정 주기마다 실행해주는게 좋다.

# Part 4

## 프로젝트에 대한 리뷰



- ① 에러 처리하는 부분의 부족
- ② 정리되지 않은 서블릿 DAO DTO
- ③ 부족한 자바스크립트에 대한 이해
- ④ 다양한 기술의 부재로 인해 구현하지 못한 기능들