

# I 실기연습문제01 커피

※ 저작권법(제25조2항)에 따라 학교 수업을 위한 저작물(사진, 글, 그림, 영상 등)은 저작물을 공유 또는 게시하는 행위는 저작권 위반에 해당될 수 있습니다.

## I 문제

### 정보처리 산업기사 외부평가 실기시험(2018.10.14.)

#### 가. 과제 개요

- 1) 본 과제는 매장별 커피 판매현황을 관리하기 위한 프로그램이다. 본 과제에서는 판매등록, 판매현황, 매장별 판매액, 상품별 판매액 확인 및 등록이 가능하도록 프로그램을 제작하시오.
- 2) 프로그램 개발을 위해 개발 환경을 확인하고, 필요한 설정을 수행하시오.

#### 나. 업무 요건

- 1) 데이터베이스에 테이블을 생성하여야 한다. 테이블의 데이터는 샘플 데이터를 참조하여 테스트를 실시하여야 한다.

#### 다. 프로젝트 준비

- 1) 접속에 사용할 오라클 계정은 'system'이고 암호는 '1234'이다.
- 2) 이클립스(eclipse)의 작업영역(workspace)은 'cWcoffeeManage'를 사용한다.
- 3) 프로젝트를 생성하기 전에 java, jsp, html, css, text 파일의 기본 인코딩을 'UTF-8'로 지정한다.  
(이클립스 Window - Preference 메뉴)
- 4) 이클립스에서 톱캣을 연동하여 실행하기 위한 설정을 수행해야 한다.
- 5) 오라클 관리를 위해 8080 포트를 사용하고 있기 때문에, 톱캣 서버는 8090 포트를 사용하도록 권장한다.
- 6) 프로젝트 유형은 'Dynamic Web Project'를 생성하고, 프로젝트 이름은 'Coffee\_비번호'를 사용한다.  
(비번호를 수험자가 부여받은 번호를 사용한다.)
- 7) 시험 후 이클립스 작업영역(workspace), 즉 'cWcoffeeManage' 디렉토리를 반드시 'sw\_비번.zip'으로 압축해서 제출해야 한다.

[참고] 아래 소스는 Oracle DataBase Connection 함수이다.

```
package DBPKG;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public static Connection getConnection( ) throws Exception{
    Class.forName("oracle.jdbc.OracleDriver");
    Connection con = DriverManager.getConnection
        ("jdbc:oracle:thin:@//localhost:1521/x", "system", "1234");
    return con;
}
```

라. 수행작업

1) 메인화면은 다음과 같은 디자인을 참고하여 작성하시오.

매장별 커피 판매관리 ver 1.0					<header>
판매등록	판매현황	매장별판매액	상품별판매액	홈으로	<nav>
<p style="text-align: center;"><b>매장별 커피 판매관리 프로그램</b></p> <p>매장별 커피 판매를 관리하기 위한 프로그램이다.</p> <p>1. 상품테이블, 매장테이블, 판매테이블을 추가한다.</p> <p>2. 판매등록, 판매현황, 매장별판매액, 상품별판매액 페이지를 작성한다.</p> <p>3. 올바르게 구현되었는지 확인한다.</p>					<section>
Copyright © 2018 All right reserved Semyeong High School					<footer>

2) 데이터 입출력 요건에 맞게 테이블을 생성하시오.

가) 상품 테이블 ( 테이블 명 : tbl\_product\_01 )

순서	컬럼ID	컬럼명	형태	길이	NULL	비고
1	pcode	상품코드	varchar2	10	NOT NULL	Primary Key
2	name	상품명	varchar2	20		
3	cost	금액	number	10		

나) 매장 테이블 ( 테이블 명 ; tbl\_shop\_01 )

순서	컬럼ID	컬럼명	형태	길이	NULL	비고
1	scode	매장코드	varchar2	10	NOT NULL	Primary key
2	sname	매장이름	varchar2	20		

다) 판매 테이블 ( 테이블 명 : tbl\_salelist\_01 )

순서	컬럼ID	컬럼명	형태	길이	NULL	비고
1	saleno	판매번호	number	10	NOT NULL	Primary key
2	pcode	상품코드	varchar2	10	NOT NULL	
3	saledate	판매일	date			
4	scode	매장코드	varchar2	10	NOT NULL	
5	amount	수량	number	10		

3) 샘플 데이터를 SQL 문장을 사용하여 각각 생성된 테이블에 추가 입력하시오.

가) 상품정보 데이터

상품코드	이름	금액
AA01	아메리카노	3000
AA02	에스프레소	3500
AA03	카페라떼	4000
AA04	카라멜마끼	4500
AA05	카푸치노	5000
AA06	초코롤케익	6000
AA07	녹차롤케익	6500
AA08	망고쥬스	7000
AA09	핫초코	2500

나) 매장 데이터

매장코드	매장이름
S001	강남점
S002	강서점
S003	강동점
S004	강북점
S005	동대문점
S006	인천점

다) 판매 테이블

판매번호	판매코드	판매일	매장코드	수량
100001	AA01	20180902	S001	50
100002	AA03	20180902	S002	40
100003	AA04	20180902	S002	20
100004	AA04	20180902	S001	30
100005	AA05	20180902	S004	40
100006	AA03	20180902	S004	30
100007	AA01	20180902	S003	40
100008	AA04	20180902	S004	10
100009	AA01	20180902	S003	20
100010	A005	20180902	S003	30
100011	AA01	20180902	S001	40
100012	AA03	20180902	S002	50
100013	AA04	20180902	S002	50
100014	AA05	20180902	S004	20
100015	AA01	20180902	S003	30

4) 화면별 업무 요구사항 및 화면 구성 요건에 맞게 화면을 구현하시오.

[참고사항]

- 화면의 구성요소는 필수 사항이다.
- 화면의 색깔, 폰트 등 스타일 구성요소는 선택사항이다.

가) 시작화면(index.jsp)

- ① 시작화면은 헤더, 메뉴, 세션, 푸터로 구성된다. 메뉴는 '판매등록', '판매현황', '매장별판매액', '상품별 판매액', '홈으로'로 구성된다.
- ② 푸터(footer)는 저작권 관련 정보로 구성된다.

매장별 커피 판매관리 ver 1.0				
판매등록	판매현황	매장별판매액	상품별판매액	홈으로
<b>매장별 커피 판매관리 프로그램</b>  매장별 커피 판매를 관리하기 위한 프로그램이다.  1. 상품테이블, 매장테이블, 판매테이블을 추가한다. 2. 판매등록, 판매현황, 매장별판매액, 상품별판매액 페이지를 작성한다. 3. 올바르게 구현되었는지 확인한다.				
Copyright © 2018 All right reserved Semyeong High School				

나) 판매등록 화면

매장별 커피 판매관리 ver 1.0																
판매등록	판매현황	매장별총판매액	상품별판매액	홈으로												
<b>판매등록</b> <table border="1" style="margin: auto;"> <tr> <td>판매번호</td> <td><input type="text"/></td> </tr> <tr> <td>상품코드</td> <td><input type="text"/></td> </tr> <tr> <td>판매날짜</td> <td><input type="text"/></td> </tr> <tr> <td>매장코드</td> <td><input type="text"/></td> </tr> <tr> <td>판매수량</td> <td><input type="text"/></td> </tr> <tr> <td colspan="2" style="text-align: center;"> <input type="button" value="등록"/> <input type="button" value="다시쓰기"/> </td> </tr> </table>					판매번호	<input type="text"/>	상품코드	<input type="text"/>	판매날짜	<input type="text"/>	매장코드	<input type="text"/>	판매수량	<input type="text"/>	<input type="button" value="등록"/> <input type="button" value="다시쓰기"/>	
판매번호	<input type="text"/>															
상품코드	<input type="text"/>															
판매날짜	<input type="text"/>															
매장코드	<input type="text"/>															
판매수량	<input type="text"/>															
<input type="button" value="등록"/> <input type="button" value="다시쓰기"/>																
Copyright © 2018 All right reserved Semyeong High School																

다) 판매현황 화면

매장별 커피 판매관리 ver 1.0

판매등록   판매현황   매장별총판매액   상품별판매액   홈으로

판매현황

판매번호	상품코드	판매날짜	매장코드	상품명	판매수량	총판매액
100001	AA01	2018-09-02	S001	아메리카노	50	150,000
100002	AA03	2018-09-02	S002	카페라떼	40	160,000
100003	AA04	2018-09-02	S002	카라멜마끼	20	90,000
100004	AA04	2018-09-02	S001	카라멜마끼	30	135,000
100005	AA05	2018-09-02	S004	카푸치노	40	200,000
100006	AA03	2018-09-02	S004	카페라떼	30	120,000
100007	AA01	2018-09-02	S003	아메리카노	40	120,000
100008	AA04	2018-09-02	S004	카라멜마끼	10	45,000
100009	AA01	2018-09-02	S003	아메리카노	20	60,000
100010	AA05	2018-09-02	S003	카푸치노	30	150,000
100011	AA01	2018-09-02	S001	아메리카노	40	120,000
100012	AA03	2018-09-02	S002	카페라떼	50	200,000
100013	AA04	2018-09-02	S002	카라멜마끼	50	225,000
100014	AA05	2018-09-02	S004	카푸치노	20	100,000
100015	AA01	2018-09-02	S003	아메리카노	30	90,000

Copyright © 2018 All right reserved Semyeong High School

라) 매장별 판매액 화면

매장별 커피 판매관리 ver 1.0		
판매등록	판매현황	매장별판매액 상품별판매액 홈으로
매장별 판매액		
매장코드	매장명	매장별 판매액
S001	강남점	405,000
S002	강서점	675,000
S003	강동점	420,000
S004	강북점	465,000
Copyright © 2018 All right reserved Semyeong High School		

마) 상품별판매액 화면

매장별 커피 판매관리 ver 1.0

판매등록   판매현황   매장별판매액   상품별판매액   홈으로

상품별 판매액

상품코드	상품명	상품별판매액
AA01	아메리카노	540,000
AA03	카페라떼	480,000
AA04	카라멜마끼	495,000
AA05	카푸치노	450,000

Copyright © 2018 All right reserved Semyeong High School

## 해설

정보처리 산업기사의 문제는 다른 여러 프로그래밍 문제가 그러하듯 정답이 여러 개 일 수 있다. 화면상에서 요구 조건이 잘 처리되어진 채 나온다면 정답으로 간주된다.

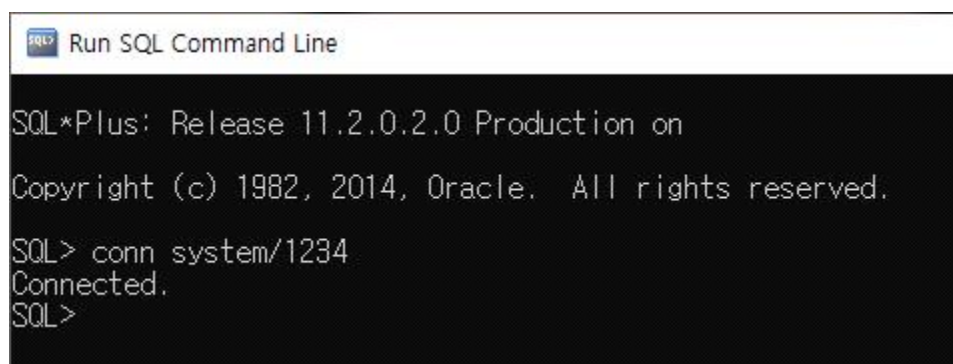
## 프로젝트 준비

1) 접속에 사용할 **오라클 계정은 'system'이고 암호는 '1234'이다.**

검색을 통해서 Run SQL Command Line을 검색하여 실행한다. 우리가 사용한 오라클 데이터베이스의 테이블을 관리하기 위해서 다음의 프로그램을 꾸준히 사용할 예정이다.



Run SQL Command Line을 실행한 이후에는 문제에서 요구하는 계정의 아이디와 암호로 접근을 하게 된다. 이때 암호는 오라클을 설치할 때 지정해준 암호이다.



```
SQL > conn system/1234
```

로그인을 하는 명령어

‘conn 아이디 / 비밀번호’ 순으로 작성

오라클에 접속하면 가장 먼저 할 것은 auto commit을 하도록 설정을 해주는 작업이다. 자동으로 반영되도록 하지 않을 경우에는 테이블에 들어간 자료들이 반영되지 않아서 웹상에서 확인함에 있어 어려움을 겪는 경우들이 생긴다. 이를 대비하기 위해 autocommit을 해주면 편하다.

```
SQL > set autocommit on ;
```

commit은 수정된 내용을 반영한다는 의미이다.  
이때 자동(auto)으로반영하도록 해주면 편하다.

2) 이클립스(eclipse)의 작업영역(workspace)은 'c:\coffeeManage'를 사용한다.  
C 드라이브에 들어가서 coffeeManage 라는 폴더를 우선 만들어준다. 작업영역은 이클립스를 실행한 이후에 지정하게 되는데 실기시험을 볼 때는 이클립스를 키기 전에 프로젝트 준비 부분의 4,5번 부분을 먼저 해주는 것이 좋다.

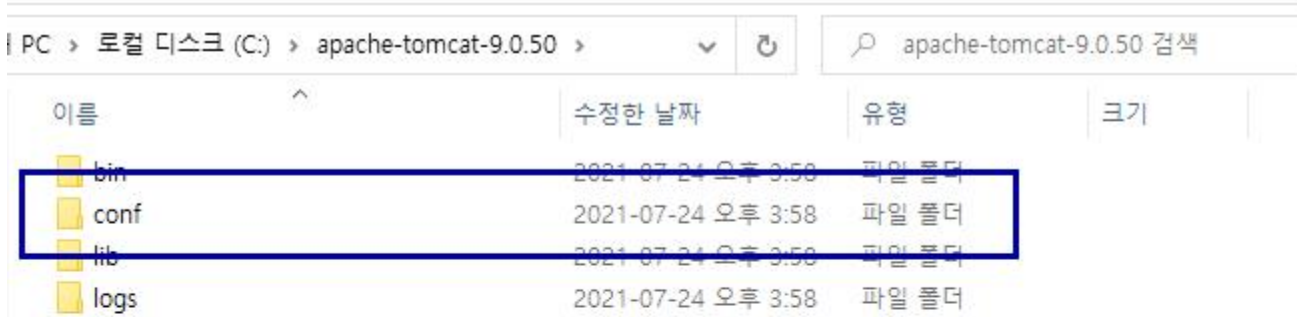
- 4) 이클립스에서 톰캣을 연동하여 실행하기 위한 설정을 수행해야 한다.
- 5) 오라클 관리를 위해 8080 포트를 사용하고 있기 때문에, 톰캣 서버는 8090 포트를 사용하도록 권장한다.

오라클은 8080 포트를 사용하고 톰캣 서버 역시 8080 포트를 사용한다. 즉, 두 개의 프로그램이 하나의 포트를 이용하려다보니 둘 중 하나는 실행이 되지 않는 문제가 발생하고는 한다. 그렇기 때문에 실기시험을 위해서 포트 번호를 바꿀 수 있어야 한다.  
먼저 톰캣 서버가 설치되어 있는 C드라이브의 해당 폴더를 찾아 간다.

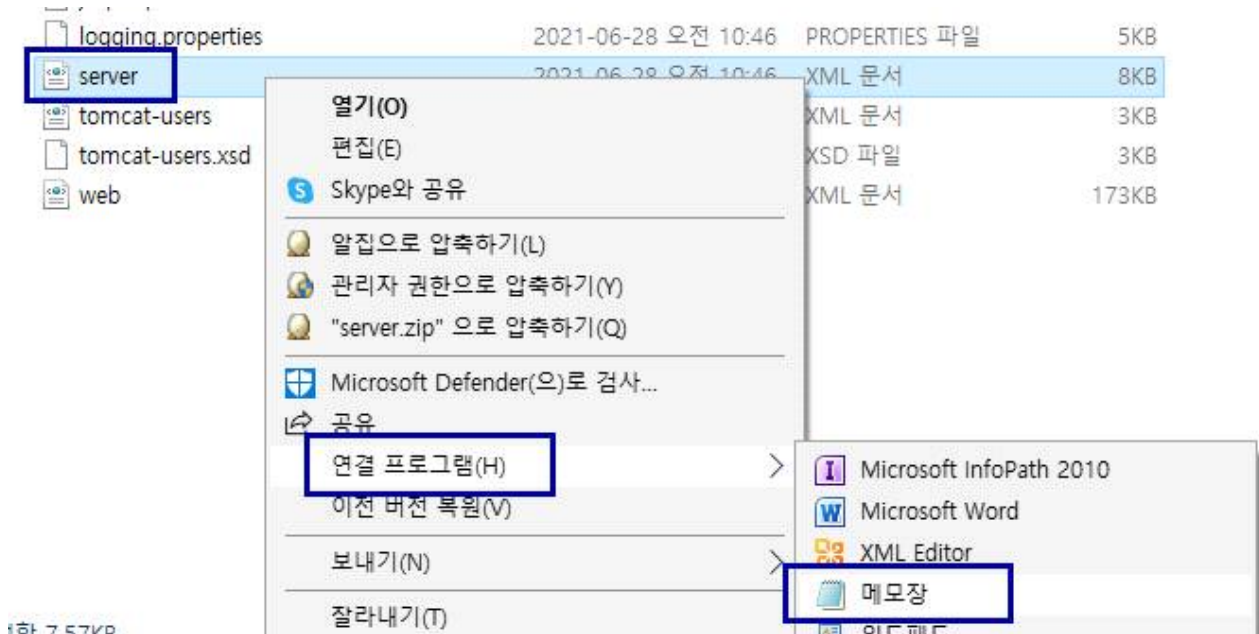
PC > 로컬 디스크 (C:)		로컬 디스크 (C:) 검색	
이름	수정된 날짜	유형	크기
apache-tomcat-9.0.50	2021-07-24 오후 3:58	파일 폴더	
Program Files	2021-07-24 오후 3:55	파일 폴더	
oraclexe	2021-07-23 오후 11:06	파일 폴더	



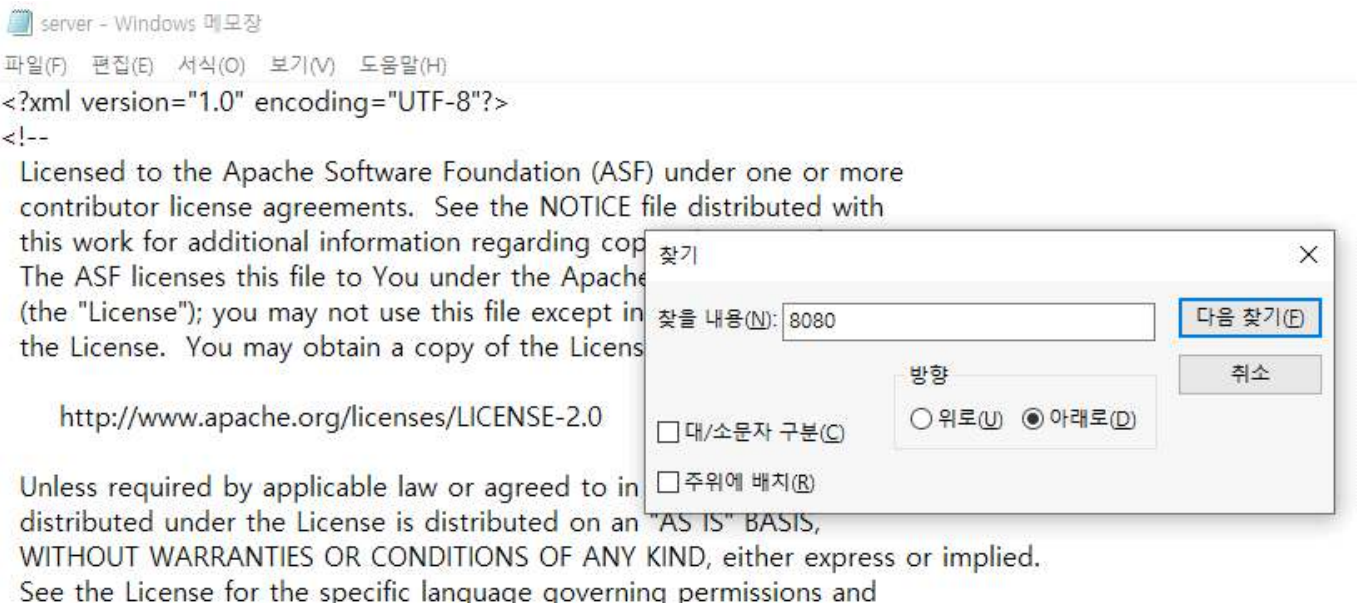
해당 apache-tomcat 폴더 내에 conf 폴더를 찾아 선택한다.



conf 폴더 내에서 server 파일 안에 있는 포트 번호 관련 설정들을 변경해주어야 한다. 이때 내용 변경을 위해 연결프로그램을 '메모장'으로 열면 편하다.



메모장에서 포트번호를 쉽게 변경하는 방법은 찾기를 사용하는 것이다. 찾기 사용을 위해 단축키 **Ctrl+F**를 누른다. 그 이후 8080을 찾으면 총 3개가 나오게 된다.



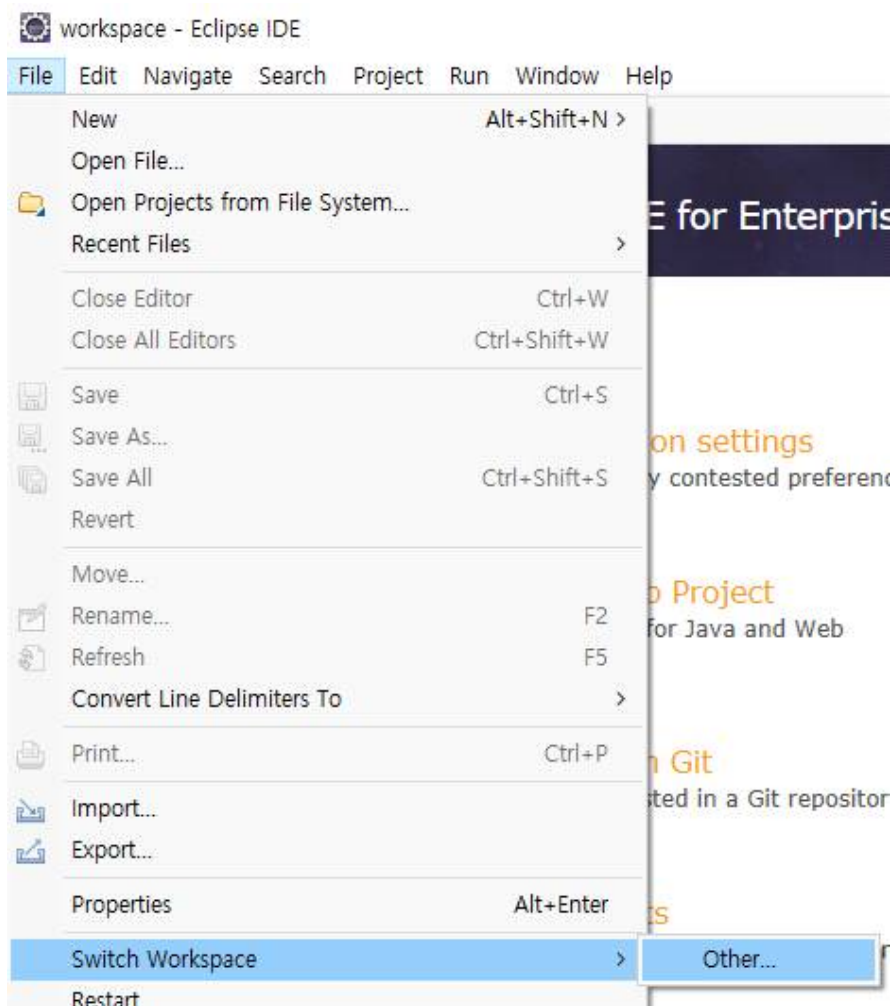


3개 모듈을 8080에서 8090으로 변경하면 톰캣 서버를 무리 없이 이용할 수 있다. 여기까지 완료된 이후 프로젝트 준비에서 2번 항목으로 돌아가면 된다.

```
61
62 <!-- A "Connector" represents an endpoint by which requests are received
63      and responses are returned. Documentation at :
64      Java HTTP Connector: /docs/config/http.html
65      Java AJP  Connector: /docs/config/ajp.html
66      APR (HTTP/AJP) Connector: /docs/apr.html
67      Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
68
69      -->
70      <Connector port="8080" protocol="HTTP/1.1"
71                connectionTimeout="20000"
72                redirectPort="8443" />
73
74      <!-- A "Connector" using the shared thread pool-->
75      <!--
76      <Connector executor="tomcatThreadPool"
77                port="8080" protocol="HTTP/1.1"
78                connectionTimeout="20000"
79                redirectPort="8443" />
80      -->
```

2) 이클립스(eclipse)의 작업영역(workspace)은 'cWcoffeeManage'를 사용한다.

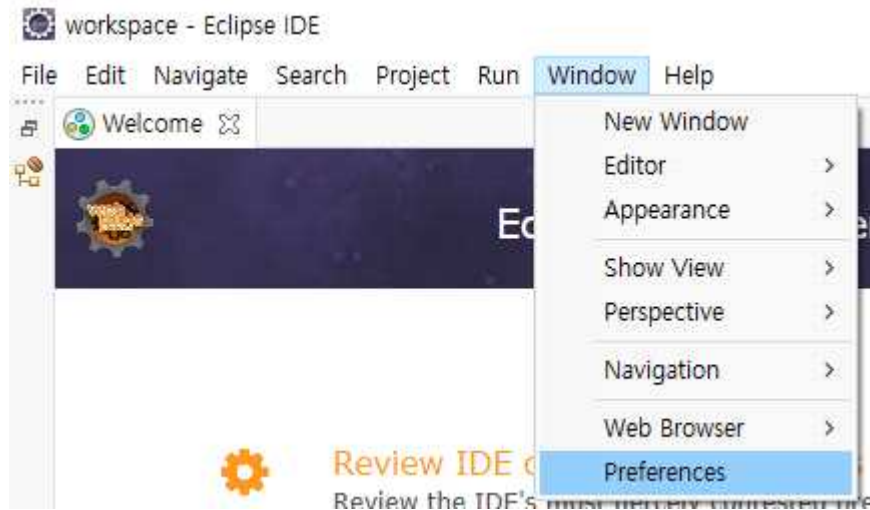
이클립스를 처음 실행할 때 작업영역을 선택할 수 있도록 나온다. 그때 미리 만들어둔 coffeeManage 폴더를 작업영역으로 선택하면 되는데 만약 잘못 지정한 경우에는 다음과 같이 변경이 가능하다.



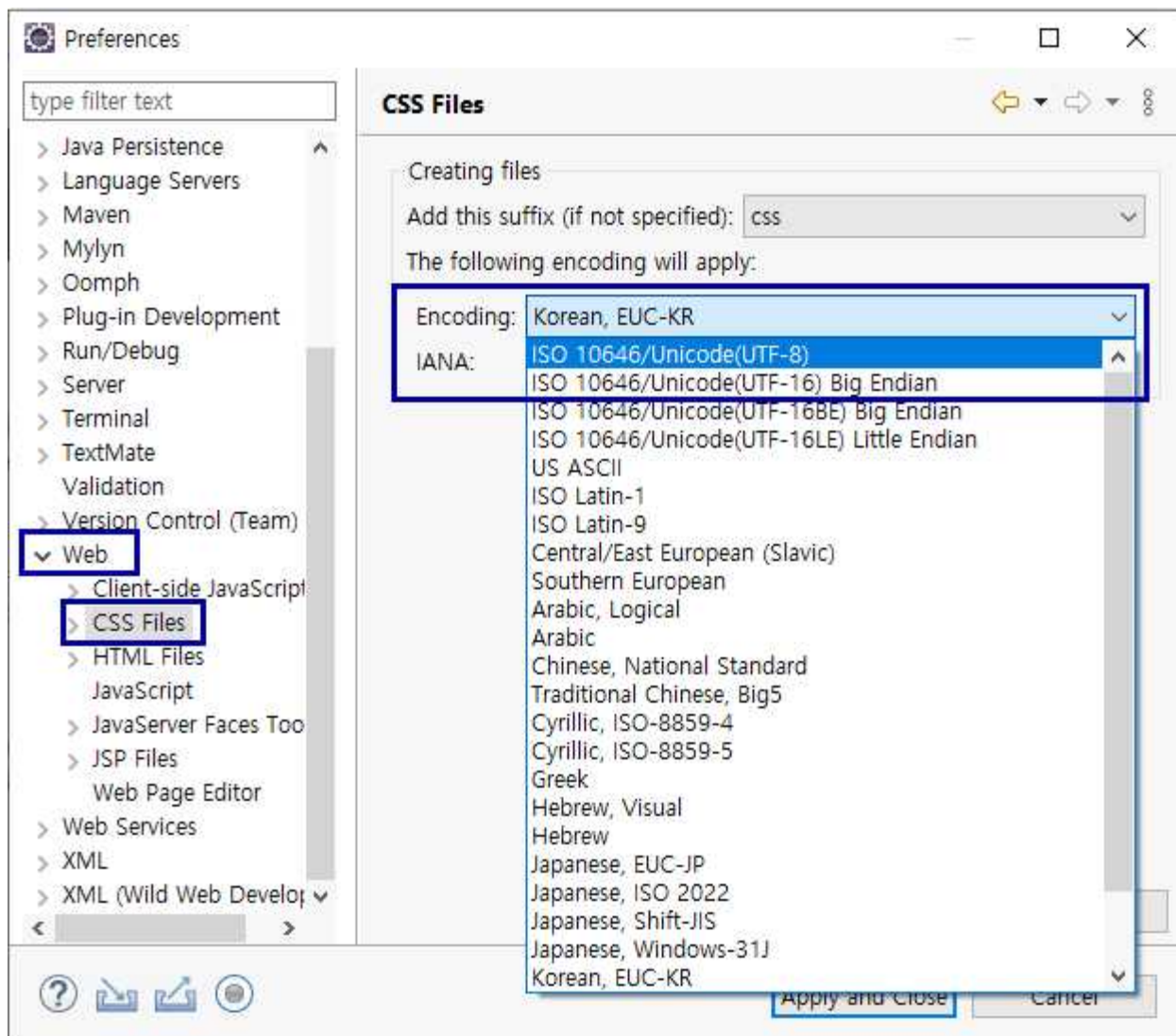
3) 프로젝트를 생성하기 전에 java, jsp, html, css, text 파일의 기본 인코딩을 'UTF-8'로 지정한다.

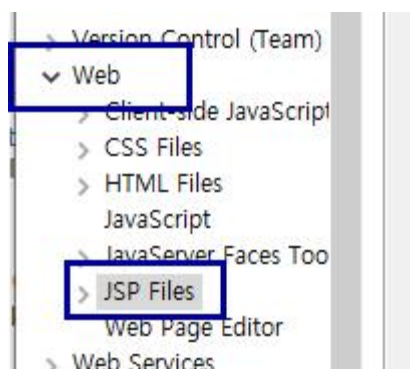
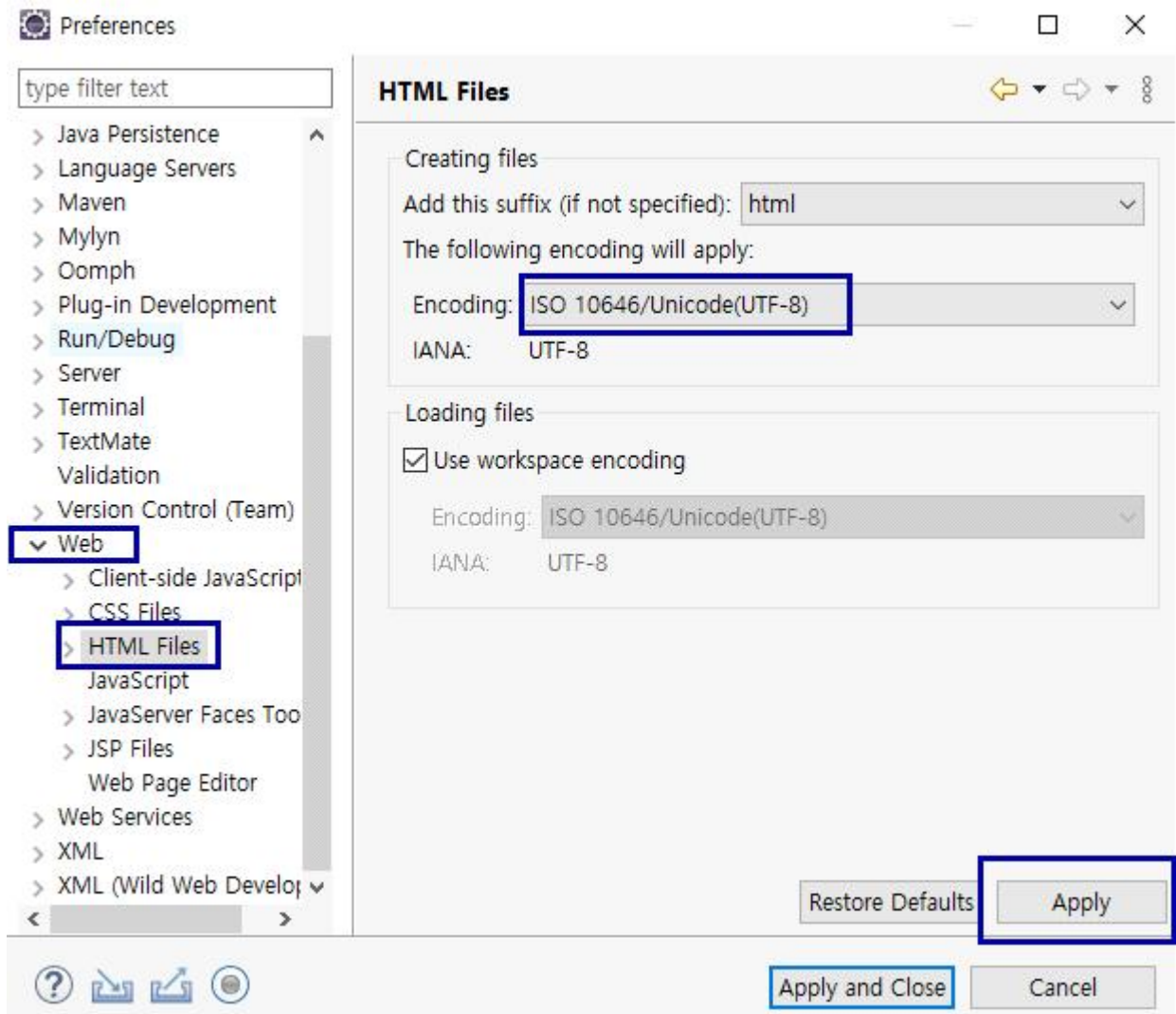
(이클립스 Window - Preference 메뉴)

기본 인코딩 타입을 변경하기 위해서는 Window - Preferences를 선택하여 변경하면 된다.



이때 변경해야 될 것은 Web 에 위치하고 있는 CSS Files, HTML Files, JSP Files이다. 아래 그림과 같이 Encoding을 UTF-8로 선택한 이후에는 Apply 버튼을 눌러준다.

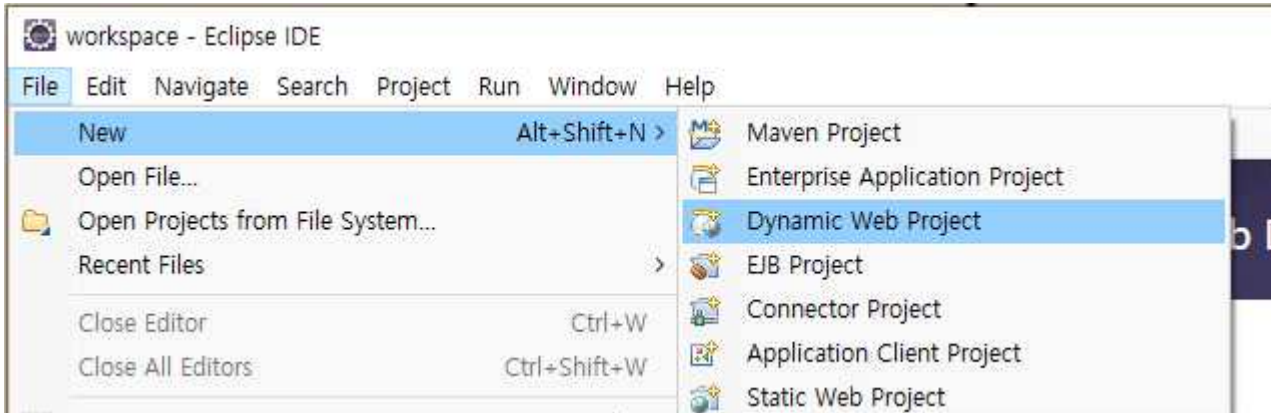




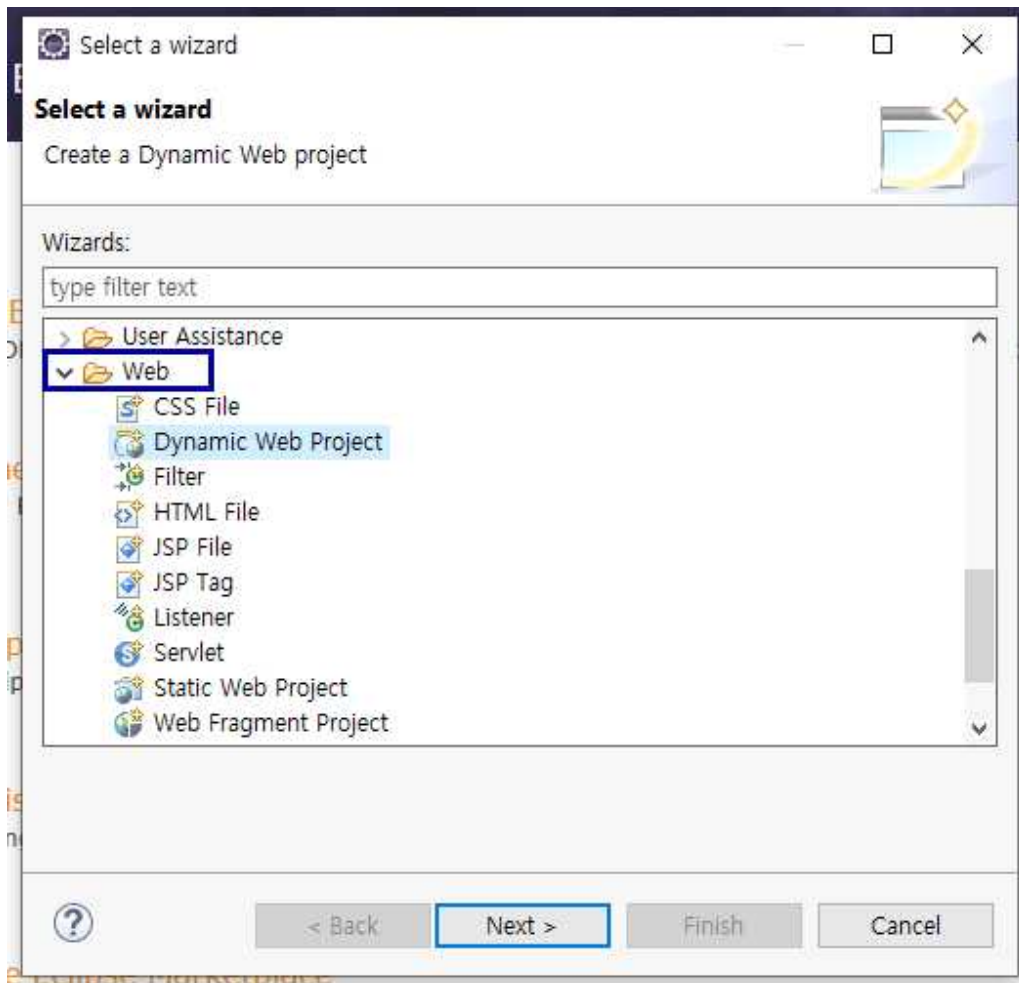
6) 프로젝트 유형은 'Dynamic Web Project'를 생성하고, 프로젝트 이름은 'Coffee\_비번호'를 사용한다.

(비번호를 수험자가 부여받은 번호를 사용한다.)

주어진 Dynamic Web Project를 만들기 위해서는 File - New에서 Dynamic Web Project를 찾으면 된다. 때에 따라서 나타나 있지 않는 경우가 있는데 그런 경우에는 File - New - Other을 선택한다.



Other을 선택하면 모든 형태의 프로젝트가 나오게 된다. 이때 Web을 찾아 그 안의 Dynamic Web Project를 선택하면 동일하게 Dynamic Web Project를 생성할 수 있다.



프로젝트를 생성할 때는 항상 주어진 문제를 확인하여 문제에서 요구하는 프로젝트 명으로 저장한다. 이번 프로젝트에서는 프로젝트명을 'Coffee\_비번호'로 저장하라고 나와 있다. 연습할 때는 비번호 대신 학번을 이용하자.

Project name: Coffee\_2021

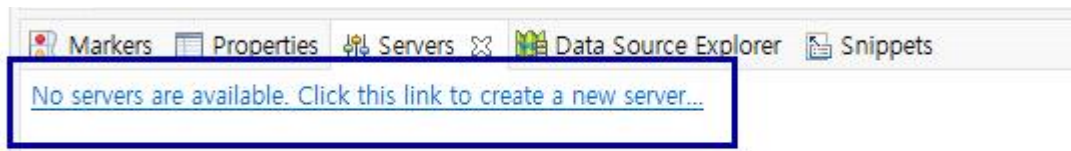


## ■ 메인 화면 구축

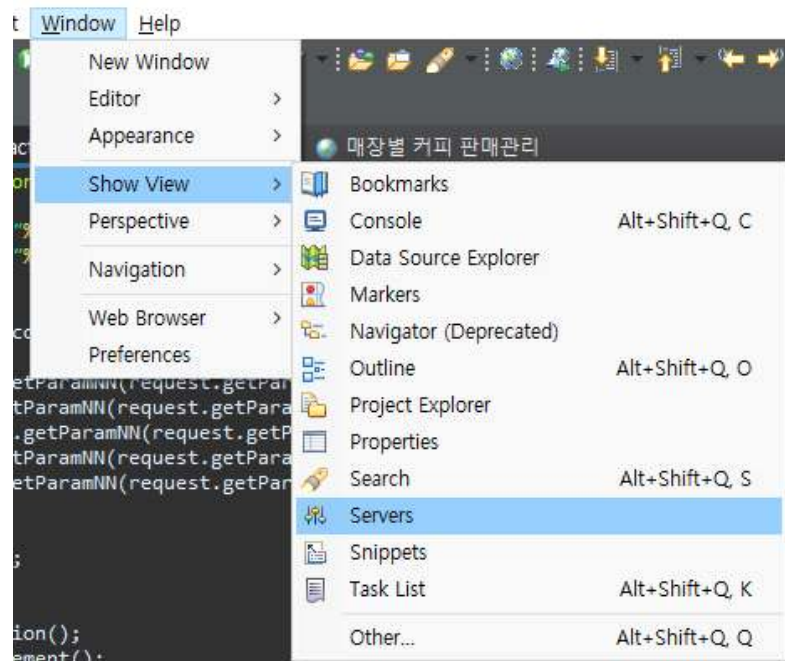
아래와 같이 메인화면이 주워졌을 때 모든 서브 페이지와 동일하게 여러 번 들어가는 부분을 우선적으로 구별해야 된다. 바로 **<header>**, **<nav>** 태그 영역과 **<footer>** 태그 영역이다. 즉 모든 페이지는 section 안의 내용만 다를 뿐 나머지 3개에 대해서는 동일한 모양을 한다. 이를 구축하기 위하여 우선 index.jsp, header.jsp, footer.jsp를 만들어야 한다.

매장별 커피 판매관리 ver 1.0					<header>
판매등록	판매현황	매장별판매액	상품별판매액	홈으로	<nav>
<p style="text-align: center;"><b>매장별 커피 판매관리 프로그램</b></p> <p>매장별 커피 판매를 관리하기 위한 프로그램이다.</p> <ol style="list-style-type: none"> <li>1. 상품테이블, 매장테이블, 판매테이블을 추가한다.</li> <li>2. 판매등록, 판매현황, 매장별판매액, 상품별판매액 페이지를 작성한다.</li> <li>3. 올바르게 구현되었는지 확인한다.</li> </ol>					<section>
Copyright © 2018 All right reserved Semyeong High School					<footer>

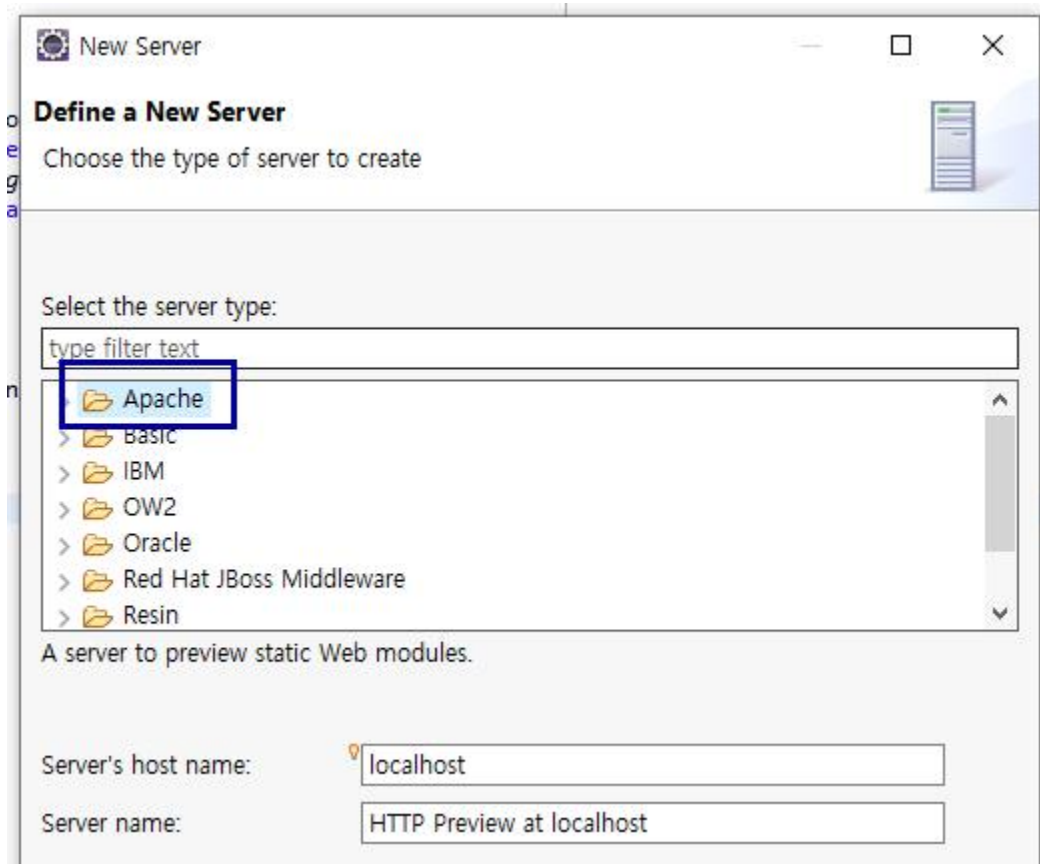
그 전에 jsp 는 백엔드 상에서 동작하기 때문에 **서버 설정**을 먼저 해준다. 서버가 설정되어 있어야만 해당 jsp 페이지들이 정상적으로 동작하는지 확인이 가능하다. 서버 설정을 위해서는 아래에서 Servers 라고 되어 있는 것을 찾아 **Click this link to create a new server**를 클릭한다.



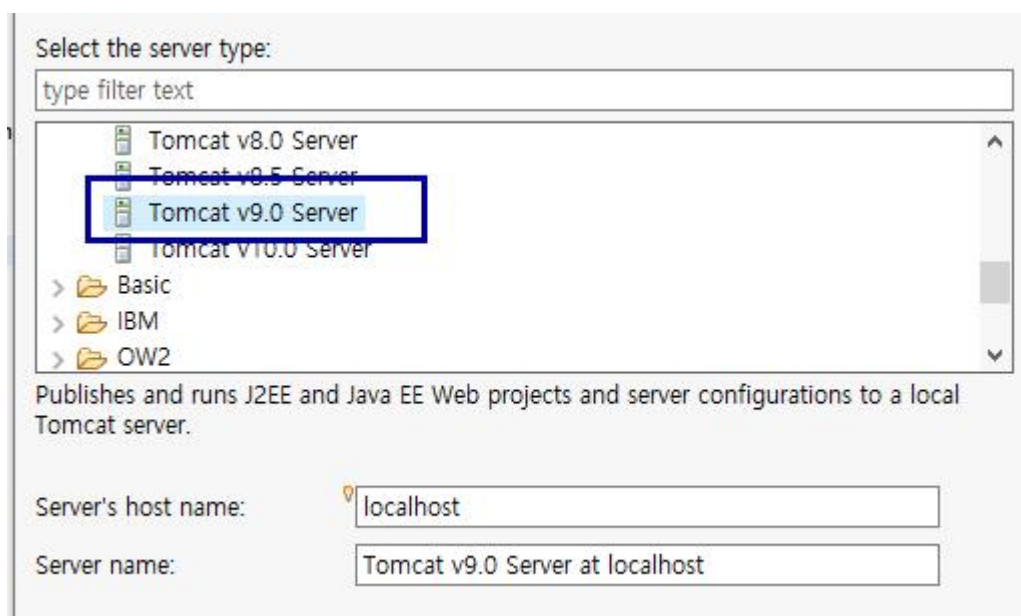
만약 Servers가 보이지 않을 경우에는 Window > Show View > Servers를 찾으면 된다.



다음은 서버를 생성하는 방법이다. 서버를 생성하기 위해서는 어떤 서버를 이용할 것인지 먼저 체크를 해야 되는데 우리는 Tomcat 9 서버를 설치하였다. Tomcat의 경우 아파치(Apache)서버이므로 Apache 폴더를 선택해 준다.

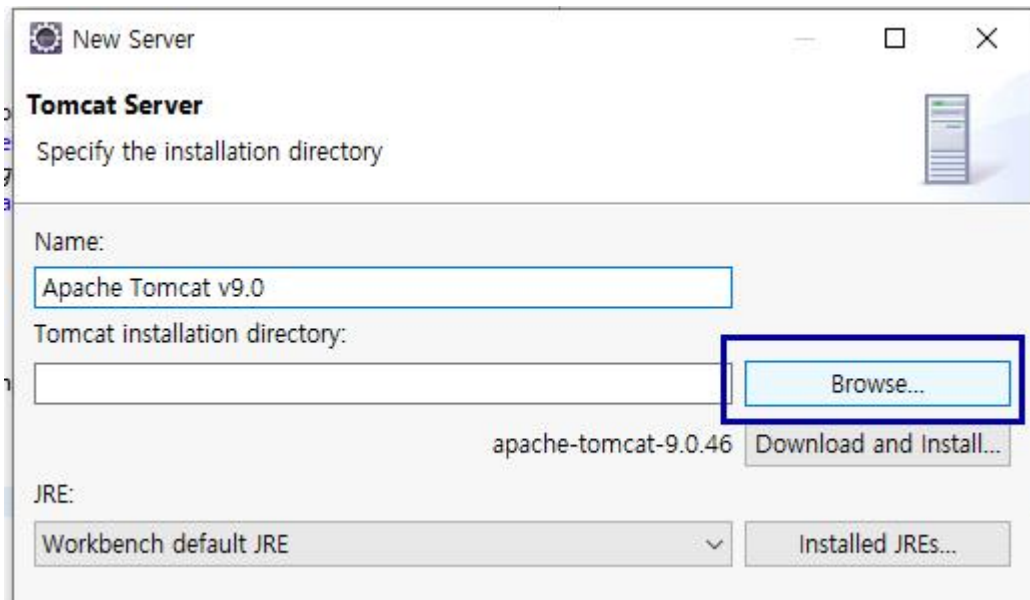


아파치(Apache) 내에서는 맞는 버전을 찾아 선택한 후 다음으로 넘어가자.

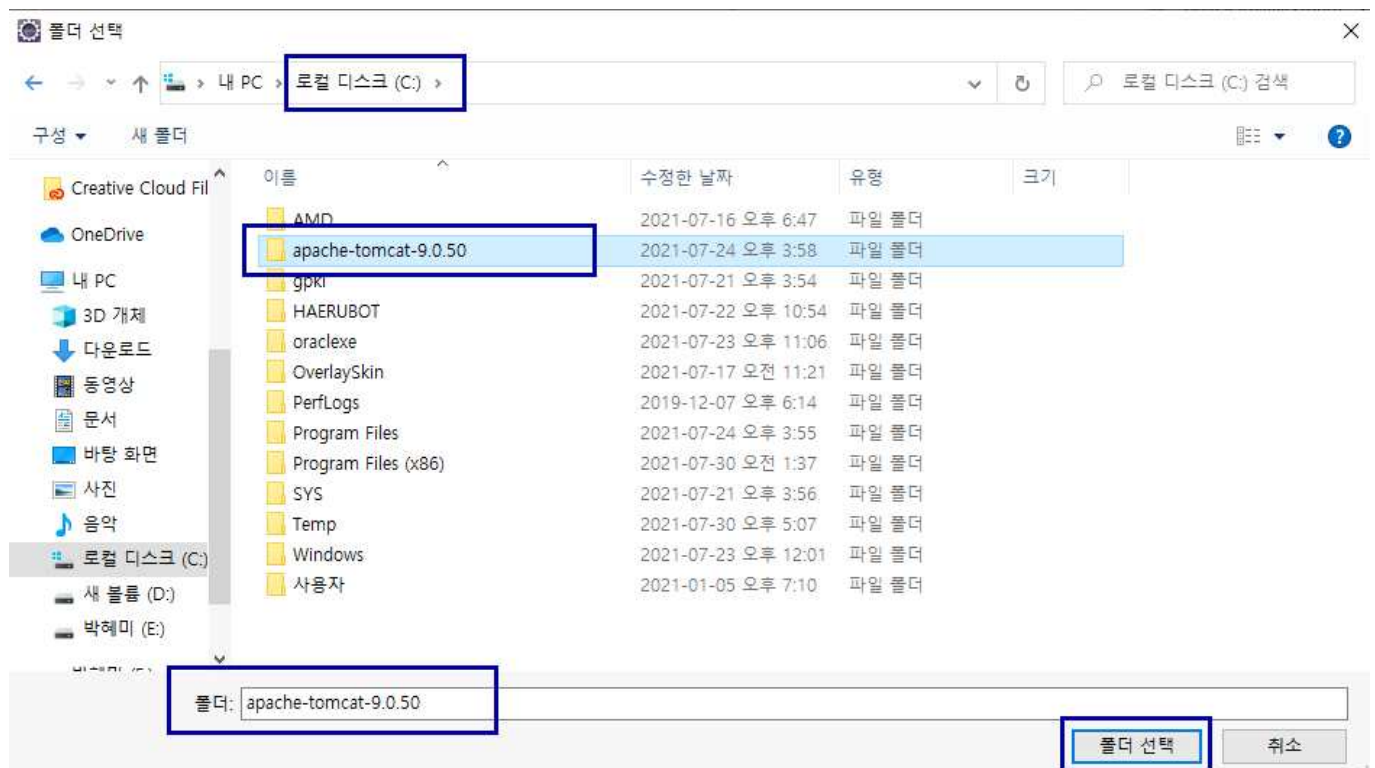




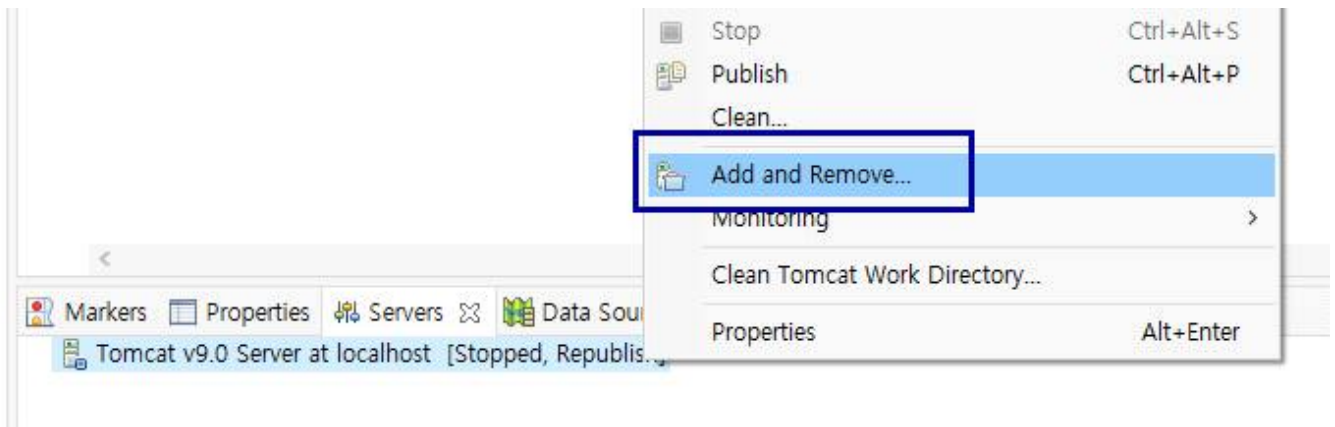
그리고 나면 Tomcat 이 설치되어 있는 위치를 물어볼 것이다. 설치 과정을 기억하고 있다면 톰캣 서버는 C드라이브에 있음을 기억할 수 있을 것이다.



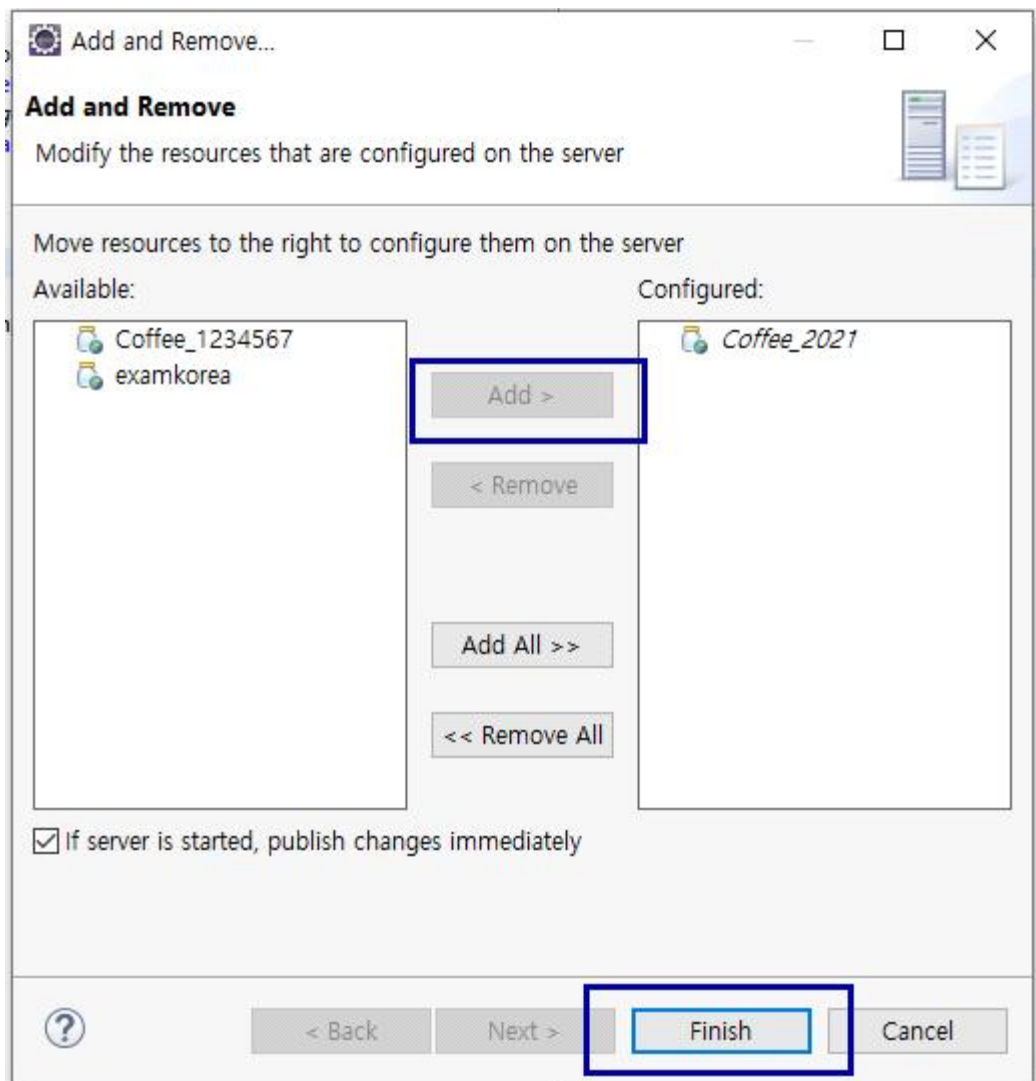
C 드라이브에 apache-tomcat-9 버전을 선택한 이후에 폴더 선택을 누른 이후 finish를 누르면 서버 생성이 완료된다.



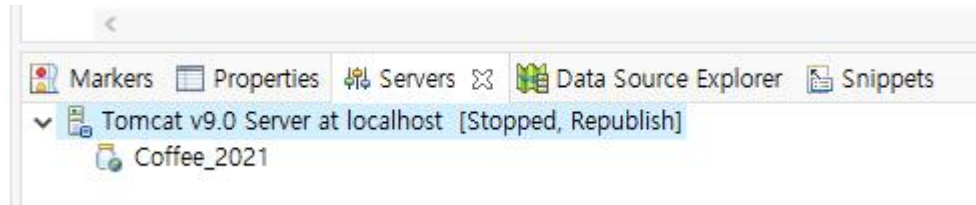
그리고 나면 서버에서 Tomcat을 확인할 수 있게 되고 오른쪽 클릭 이후에 Add and Remove를 통해서 여러 프로젝트들을 추가하거나 삭제할 수 있다.



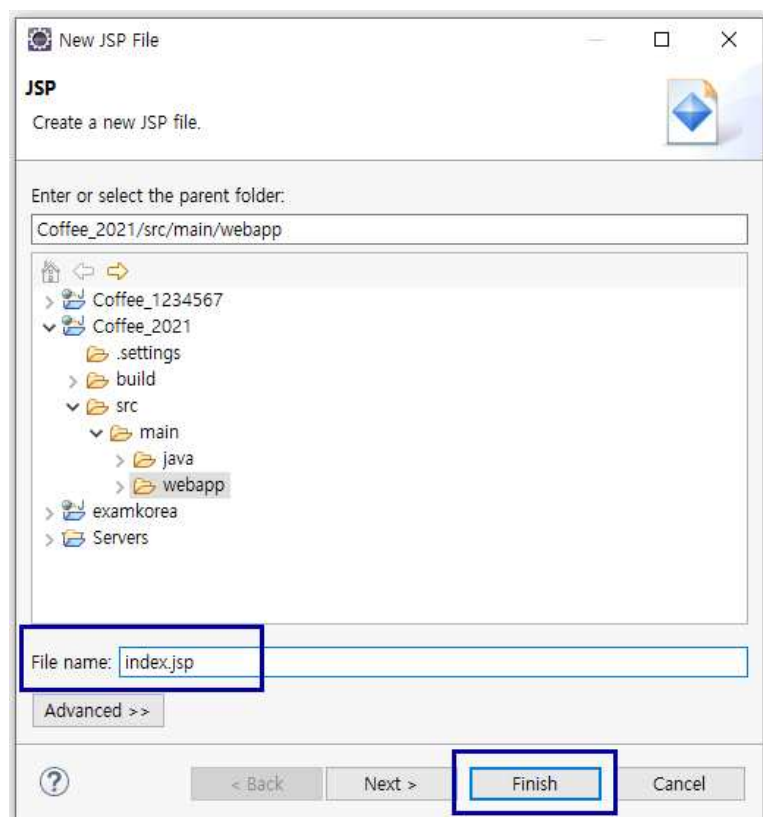
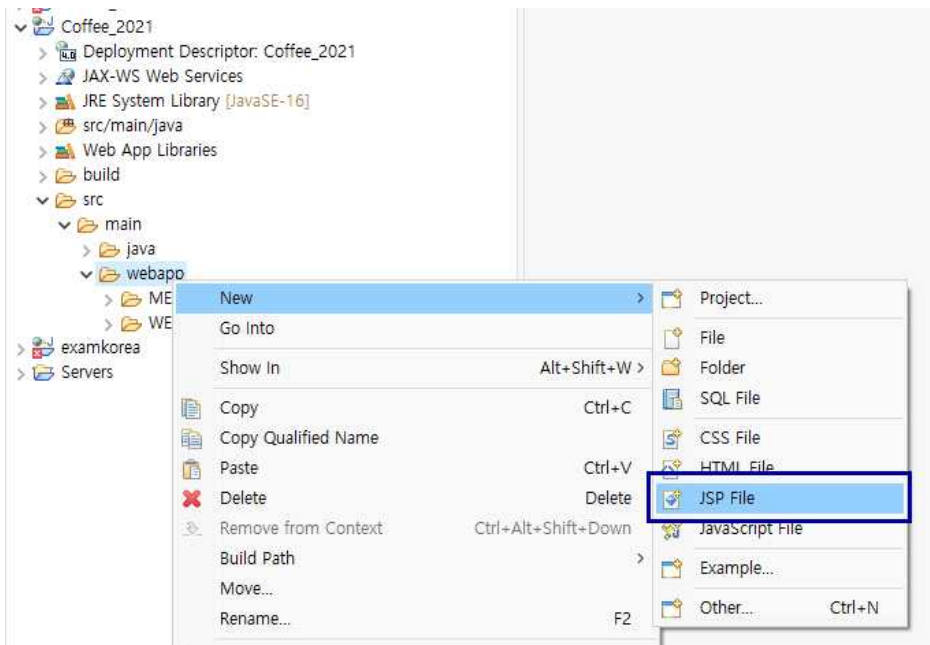
여러 개의 프로젝트를 사용하는 것도 가능하기 때문에 공부함에 있어서 여러 프로젝트를 만들고 하나의 서버로 관리하는 것이 가능하다.



다음과 같이 프로젝트 명이 서버 아래에 뜬다면 서버 설정까지도 끝난 것이다.



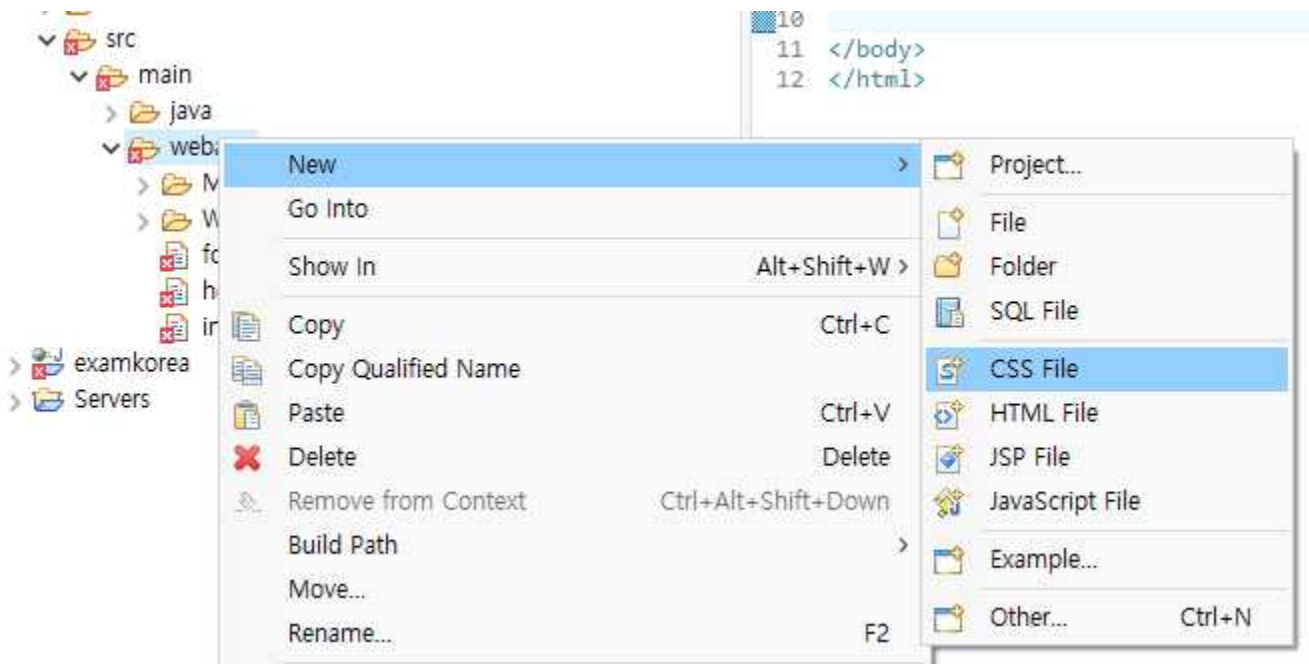
서버 설정이 끝났으면 본격적으로 index.jsp를 비롯한 페이지들을 구축하고 테스트할 수 있다. jsp 페이지는 src > main > webapp 에 만들면 된다. 서버단에서 운영되는 페이지이다보니 위치가 중요하므로 생성 위치를 잘 기억하도록 하자.



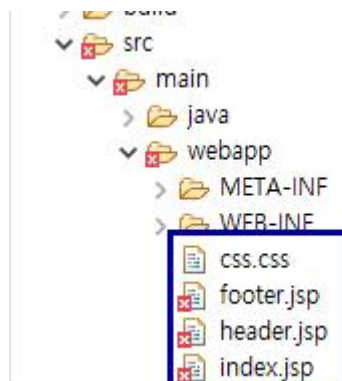
앞서서 설정을 다 했다면 만들어진 jsp 페이지는 자동으로 UTF-8 형식으로 인코딩이 지정된 채 생성될 것이다. 만약 그렇지 않다면 UTF-8 지정부터 다시 해보도록 하자.

```
*index.jsp
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6   <meta charset="UTF-8">
7   <title>insert title here</title>
8 </head>
9 <body>
10
11 </body>
12 </html>
```

필요한 JSP, CSS 파일을 다음과 같이 만들자.



모든 문제에는 다음의 4가지는 필수적으로 필요하므로 만들고 암기하도록 하자.



다시 index.jsp 로 돌아가면 모든 페이지에 공통으로 나타나는 <header>, <nav> 그리고 <foooter> 태그의 경우 하나의 다른 jsp 파일로 만들어 줄 예정이다.

```

1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6   <meta charset="UTF-8">
7   <title>매장별 커피 판매관리</title>
8   <link rel="stylesheet" href="css.css">
9 </head>
10 <body>
11   <header></header>
12   <nav></nav>
13
14   <section></section>
15
16   <footer></footer>
17 </body>
18 </html>
19

```

## header.jsp

header.jsp 에서는 <header> 태그와 <nav> 태그를 다루게 된다.

매장별 커피 판매관리 ver 1.0					<header>
판매등록	판매현황	매장별판매액	상품별판매액	홈으로	<nav>
<p>매장별 커피 판매관리 프로그램</p> <p>매장별 커피 판매를 관리하기 위한 프로그램이다.</p> <ol style="list-style-type: none"> <li>1. 상품테이블, 매장테이블, 판매테이블을 추가한다.</li> <li>2. 판매등록, 판매현황, 매장별판매액, 상품별판매액 페이지를 작성한다.</li> <li>3. 올바르게 구현되었는지 확인한다.</li> </ol>					<section>
Copyright © 2018 All right reserved Semyeong High School					<footer>

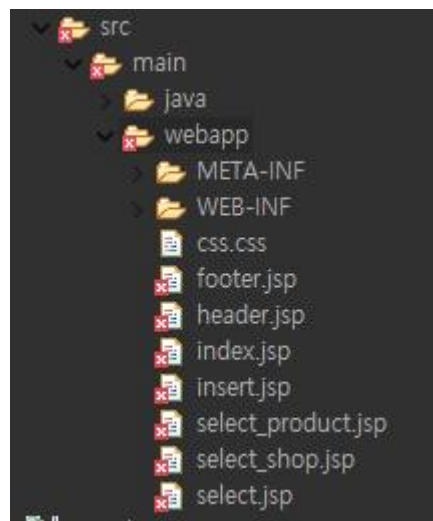
각각의 태그를 사용하라고 주어졌기 때문에 태그 안에다가 필요한 내용을 적으며 되며 여기서는 <h2>에는 제목인 매장별 커피 판매관리를 <nav> 태그 안에는 네비게이션을 구축하여 넣었다. 이때 네비게이션에서는 각각 연결될 jsp 주소를 적게 되는데 서브페이지의 역할에 따라 이름을 적어주면 좋다.

매장별 커피 판매관리 프로그램에서는 다음과 같이 이름을 주었기 때문에 다음의 insert.jsp, select.jsp, select\_shop.jsp, select\_product.jsp를 추가적으로 만들어줘야 한다.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

<header>
<h2> 매장별 커피 판매관리 ver 1.0 </h2>
</header>

<!-- 네비게이션 -->
<nav>
    <ul>
        <li><a class="nav1" href="insert.jsp">판매등록</a></li>
        <li><a class="nav2" href="select.jsp">판매현황</a></li>
        <li><a class="nav3" href="select_shop.jsp">매장별판매액</a></li>
        <li><a class="nav4" href="select_product.jsp">상품별판매액</a></li>
        <li><a class="nav5" href="index.jsp">홈으로</a></li>
    </ul>
</nav>
```





footer.jsp 에서는 <footer> 태그만 있으면 된다.

매장별 커피 판매관리 ver 1.0					<header>
판매등록	판매현황	매장별판매액	상품별판매액	홈으로	<nav>
<p>매장별 커피 판매관리 프로그램</p> <p>매장별 커피 판매를 관리하기 위한 프로그램이다.</p> <ol style="list-style-type: none"> <li>1. 상품테이블, 매장테이블, 판매테이블을 추가한다.</li> <li>2. 판매등록, 판매현황, 매장별판매액, 상품별판매액 페이지를 작성한다.</li> <li>3. 올바르게 구현되었는지 확인한다.</li> </ol>					<section>
Copyright © 2018 All right reserved Semyeong High School					<footer>

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

<footer>
Copyright &copy; 2018 Semyeong Computer High School
</footer>
```

## DBPKG

패키지의 경우 이번 커피 문제처럼 DBPKG 형태로 주어질 수도 있으며 그렇지 않는 경우도 있다. 오라클 데이터베이스를 연결하는 소스 코드를 제공해주나 이를 가공해서 사용해야 된다. 아래의 코드에서 위의 import 3가지는 하나로 합쳐서 작성할 수 있으며 유의해야 할 것은 데이터베이스를 연결할 때 아이디와 비밀번호가 올바르게 되어 있는지 확인하는 것이다.

```
package DBPKG;
```

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.SQLException;
```



```
import java.sql.*;
```

```
public static Connection getConnection( ) throws Exception{
```

```
    Class.forName("oracle.jdbc.OracleDriver");
```

```
    Connection con = DriverManager.getConnection
```

```
        ("jdbc:oracle:thin:@//localhost:1521/xe","system","1234");
```

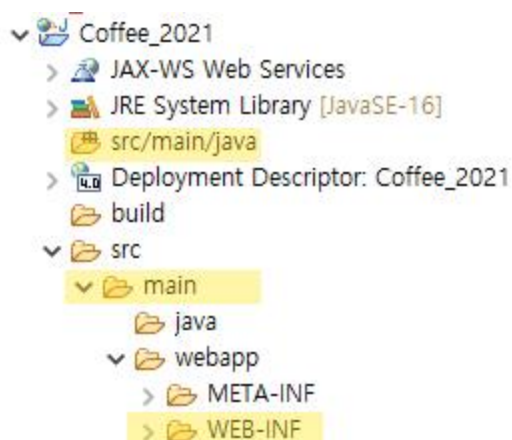
```
    return con;
```

“아이디”, “비밀번호”

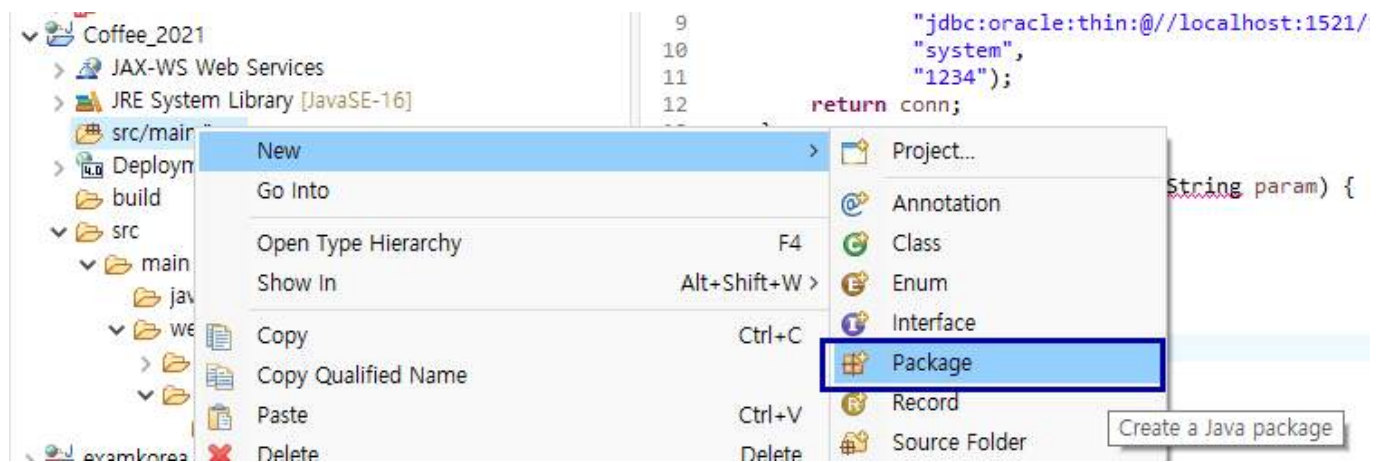
```
}
```

내용 확인이 끝났으면 DBPKG를 자바 패키지 파일로 생성해보자. 이때 이클립스의 버전에 따라 Dynamic Web Project의 폴더 구성이 다르게 보일 수 있다.

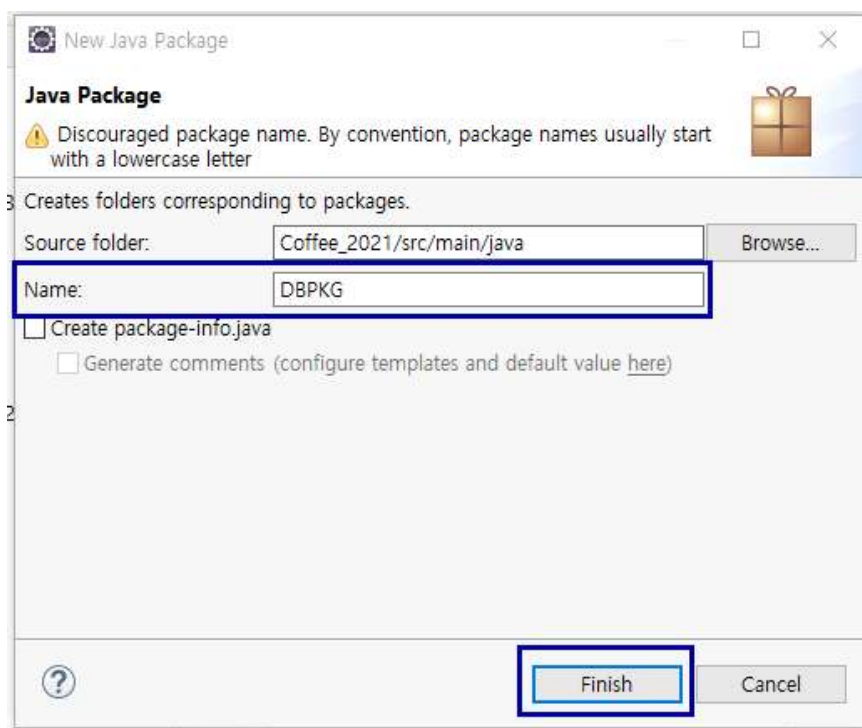
src/main/java 폴더에 자바패키지를 생성하게 되며, index.jsp를 비롯한 여러 웹페이지는 src 폴더 안에 위치하고 있는 main > webapp 폴더에 마지막으로 데이터베이스 연결을 위해 필요한 ojdbc 파일은 main 폴더 안의 webapp > WEB-INF > lib 폴더 안에 옮겨 넣게 된다.



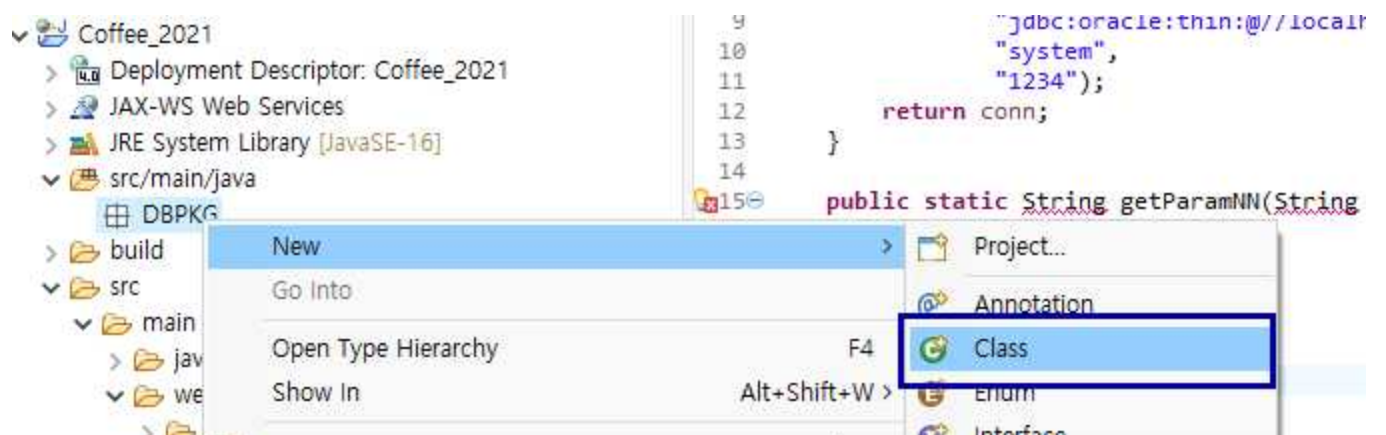
자바패키지 생성을 위해서 src/main/java 폴더에 우클릭을 해서 **New > Package**를 선택한다.



이름은 DBPKG로 지정하여 패키지를 생성한다.



생성된 패키지에 우클릭을 해서 **New > Class**를 선택하여 패키지 안에 클래스를 생성하도록 한다.



DBPKG 안의 클래스의 이름은 지정되어 있지 않지만 수업에는 Util을 사용할 예정이다. 문제에서 제공되는 부분이 아니므로 항상 Util 로 지정하여 생성하도록 하자. (JAVA는 대소문자를 구별한다.)

New Java Class

**Java Class**

⚠ This package name is discouraged. By convention, package names usually start with a lowercase letter

Source folder: Coffee\_2021/src/main/java Browse...

Package: DBPKG Browse...

☐ Enclosing type: Browse...

Name: Util

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)  
☐ Constructors from superclass  
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))  
☐ Generate comments

? Finish Cancel

그러면 DBPKG 라는 패키지 안에 Util.java 라는 클래스가 생성이 되고 추후 jsp에서 import만 하고 나면 해당 내용들을 여러 번이고 사용할 수 있는 public 라이브러리가 된다 .

```
1 package DBPKG;
2
3 public class Util {
4
5 }
6
7
```

DBPKG 안의 내용은 다음과 같이 구성한다. 위에 있는 getConnection( ) 함수는 제공되는 내용을 바탕으로 재 구성하여 만들게 되는 함수이다. 아래의 getParamNN의 경우 주어지지 않는 함수이나 추후 데이터 처리를 할 때 자주 필요하기 때문에 암기를 꼭 하도록 하자.

```
package DBPKG;

import java.sql.*;

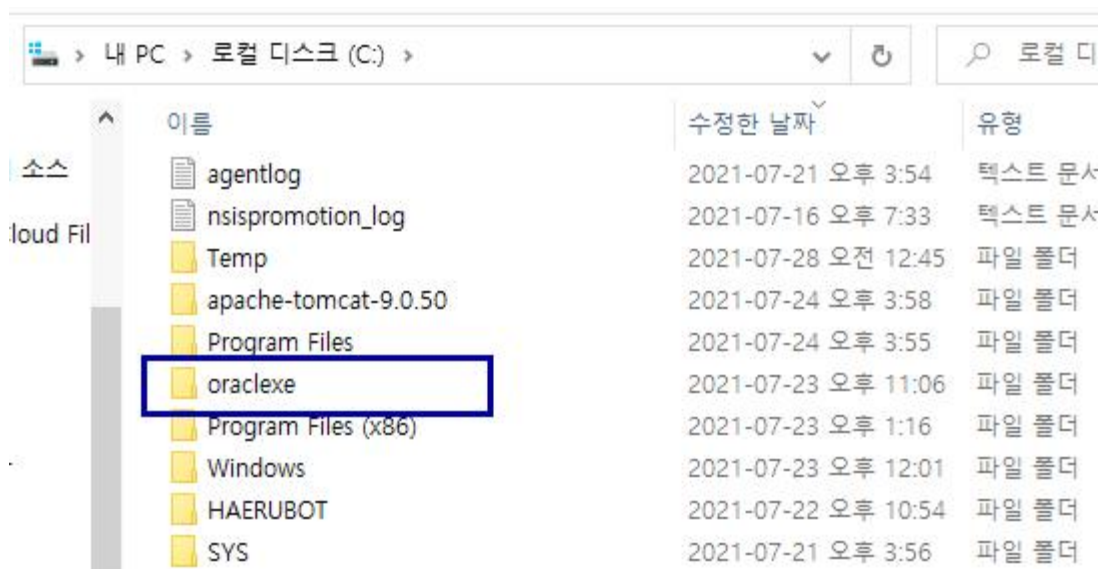
public class Util {
    public static Connection getConnection() throws Exception {
        Class.forName("oracle.jdbc.OracleDriver");
        Connection conn = DriverManager.getConnection(
            "jdbc:oracle:thin:@//localhost:1521/x",
            "system",
            "1234");
        return conn;
    }

    public static String getParamNN(String param) {
        if(param == null)
            return "";
        return param;
    }
}
```

데이터베이스를 연결하는 함수  
문제에서 제공  
(가공해서 사용)

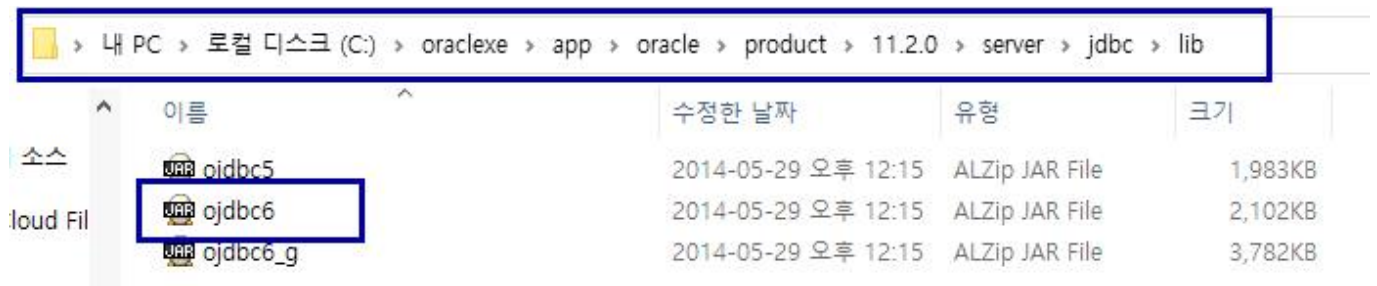
NULL 값이 데이터로 넘어 왔을 때  
공백을 입력해줌으로서  
오류를 방지하기 위한 함수(암기★)

지금 만든 DBPKG 안의 내용은 데이터베이스를 연결하여 사용하기 위한 함수이다. 다만 이클립스에서 오라클 데이터베이스를 사용하기 위해서는 위의 클래스에 나와 있는 jdbc 라는 것이 필요하다. 이 jdbc는 오라클에서 제공하기 때문에 오라클 설치 위치로 가서 해당 파일을 찾아야 한다. 오라클은 기본적으로 **C드라이브로 이동하여 oraclexe** 라는 폴더를 찾는 것으로 시작된다.

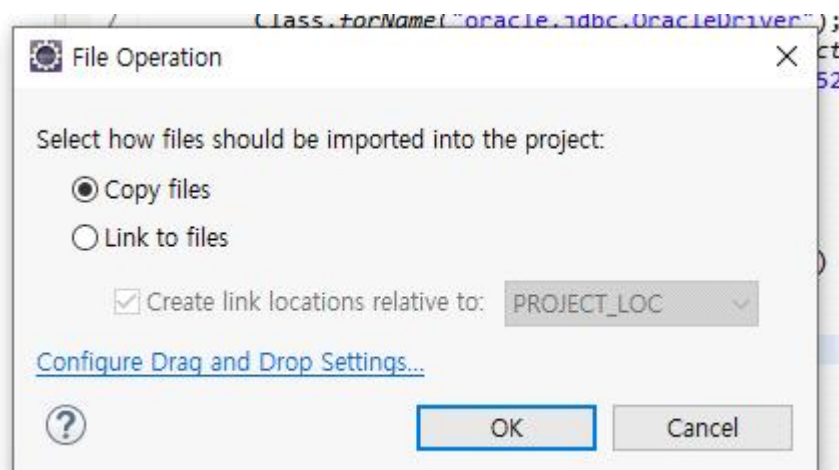
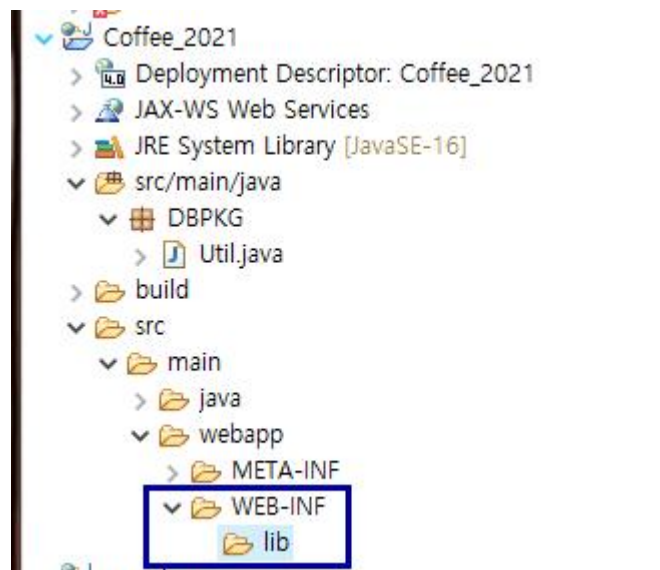




다음의 위치는 꼭 암기를 해야 된다. 시험에서 주어지지 않기 때문에 직접 찾아서 옮기는 작업을 해야 되므로 위치를 확인하고 외우도록 하자. 만약 기억이 나지 않을 경우에는 윈도우의 자체적인 검색 기능을 활용하여도 좋다. 여기서 우리가 사용할 것은 **ojdbc6 JAR 파일**이다. 압출을 풀 필요 없이 사용하므로 찾았다면 다시 이클립스로 돌아오면 된다.

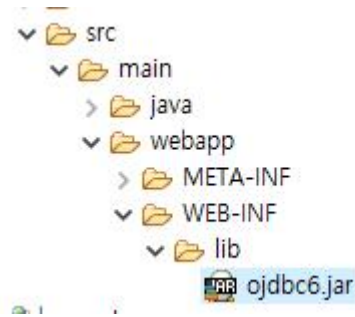


ojdbc가 들어가야 하는 위치는 WEB-INF 에 있는 lib 라는 폴더 내이다. 간단하게 **마우스로 드래그 앤 드랍**을 통해서 이동시킬 수 있다. 드래그 앤 드랍을 했음에도 이클립스 상에서 보이지 않는다면 **F5를 눌러 새로고침**을 하도록 하자.



[마우스로 드래그 앤 드랍을 했을 때 나타나는 화면]





여기까지 완료 되었다면 데이터베이스를 연결하여 사용할 준비가 다 된 것이다. 이제는 데이터베이스를 사용하기 위해 데이터베이스 내에 테이블을 생성하고 데이터를 넣는 자료를 하자.

## ■ 데이터베이스

데이터베이스는 미리 **메모장**에 작성하도록 하자. 메모장에 한 줄 작성한 이후에 오라클에 복사 + 붙여넣기를 통해 오류 여부를 확인하고 오류가 없다면 계속해서 사용하도록 하자. 오라클의 데이터베이스는 MySQL 등과는 달리 데이터베이스를 생성하지 않고 바로 테이블을 생성해서 사용하면 된다.

이때 CREATE TABLE 이라는 테이블 생성 명령어를 사용하여 다음과 같이 테이블을 생성하게 된다. 데이터타입 역시 MySQL과 다른 부분이 존재하나 문제에 주어지는 형태를 확인하여 테이블을 생성해주면 된다. 이때 길이에 유의해서 테이블을 생성해야 되며 **SQL은 대소문자를 구별하지 않기** 때문에 소문자로 작성해도 무방하다.

```
CREATE TABLE 테이블 이름(
    컬럼이름 데이터타입,
    컬럼이름 데이터타입
);
```

가) 상품 테이블 ( 테이블 명 : tbl\_product\_01 )

순서	컬럼ID	컬럼명	형태	길이	NULL	비고
1	pcode	상품코드	varchar2	10	NOT NULL	Primary Key
2	name	상품명	varchar2	20		
3	cost	금액	number	10		

**PRIMARY KEY**의 경우 유일무이해야 하며 NULL 일 수 없기 때문에 NOT NULL 제약조건을 추가적으로 줄 필요가 없다. 컬럼ID는 꼭 주어진 대로 작성하도록 하자.

```
CREATE TABLE tbl_product_01(
    pcode VARCHAR2(10) PRIMARY KEY,
    name VARCHAR2(20),
    cost NUMBER(10)
);
```

데이터를 입력할 때에도 메모장을 활용하는 것이 좋다. 모든 데이터를 넣는 값을 작성하기 전에 한줄만 작성해서 복사 + 붙여넣기를 통해서 문법 오류 없이 제대로 동작하는지 확인한 다음에 문제가 없다면 안의 값만 바꾼 INSERT 명령문을 여러 개 사용하도록 하자. 그 이후 여러 줄을 한 번에 복사해서 Run SQL Command Line에 붙여넣기 해도 곧바로 실행 되서 편리하게 데이터 입력이 가능하다. 이렇게 메모장을 이용하게 되면 추후 잘못되어 새로 테이블을 만들거나 해야 될 때도 편하게 이용할 수 있으므로 꼭 메모장에 먼저 작성하도록 하자.

```
INSERT INTO tbl_product_01 VALUES('AA01', '아메리카노', 3000);
```

모든 컬럼에 대해 값을 넣기 때문에 따로 컬럼 지정을 할 필요는 없으나  
생성했던 순서에 맞도록 값을 줘야한다.

여기서 pcode, name은 문자열이기 때문에 ‘ ’(따옴표)를 사용한다.

상품코드	이름	금액
AA01	아메리카노	3000
AA02	에스프레소	3500
AA03	카페라떼	4000
AA04	카라멜마끼	4500
AA05	카푸치노	5000
AA06	초코롤케익	6000
AA07	녹차롤케익	6500
AA08	망고쥬스	7000
AA09	핫초코	2500

아래와 같이 한 줄을 먼저 실행한 다음에 문제가 없으면 메모장에 내용을 추가하여 여러 데이터를 넣을 수 있도록 한다.

```
SQL> INSERT INTO tbl_product_01 VALUES('AA01', '아메리카노', 3000);
1 row created.
```

아래와 같이 메모장에 우선적으로 적은 이후에 모든 행을 복사해서 붙여넣기를 하면 한 번에 여러 줄의 쿼리문이 실행이 된다.

```
CREATE TABLE tbl_product_01(
    pcode VARCHAR2(10) PRIMARY KEY,
    name VARCHAR2(20),
    cost NUMBER(10)
);
INSERT INTO tbl_product_01 VALUES('AA01', '아메리카노', 3000);
INSERT INTO tbl_product_01 VALUES('AA02', '에스프레소', 3500);
INSERT INTO tbl_product_01 VALUES('AA03', '카페라떼', 4000);
INSERT INTO tbl_product_01 VALUES('AA04', '카라멜마끼', 4500);
INSERT INTO tbl_product_01 VALUES('AA05', '카푸치노', 5000);
INSERT INTO tbl_product_01 VALUES('AA06', '초코롤케익', 6000);
INSERT INTO tbl_product_01 VALUES('AA07', '녹차롤케익', 6500);
INSERT INTO tbl_product_01 VALUES('AA08', '망고쥬스', 7000);
INSERT INTO tbl_product_01 VALUES('AA09', '핫초코', 2500);
```

나) 매장 테이블 ( 테이블 명 ; tbl\_shop\_01 )

순서	컬럼ID	컬럼명	형태	길이	NULL	비고
1	scode	매장코드	varchar2	10	NOT NULL	Primary key
2	sname	매장이름	varchar2	20		

매장코드	매장이름
S001	강남점
S002	강서점
S003	강동점
S004	강북점
S005	동대문점
S006	인천점

```
CREATE TABLE tbl_shop_01(
    scode VARCHAR2(10) PRIMARY KEY,
    sname VARCHAR2(20)
);
INSERT INTO tbl_shop_01 VALUES('S001', '강남점');
INSERT INTO tbl_shop_01 VALUES('S002', '강서점');
INSERT INTO tbl_shop_01 VALUES('S003', '강동점');
INSERT INTO tbl_shop_01 VALUES('S004', '강북점');
INSERT INTO tbl_shop_01 VALUES('S005', '동대문점');
INSERT INTO tbl_shop_01 VALUES('S006', '인천점');
```

다) 판매 테이블 ( 테이블 명 : tbl\_salelist\_01 )

순서	컬럼ID	컬럼명	형태	길이	NULL	비고
1	saleno	판매번호	number	10	NOT NULL	Primary key
2	pcode	상품코드	varchar2	10	NOT NULL	
3	saledate	판매일	date			
4	scode	매장코드	varchar2	10	NOT NULL	
5	amount	수량	number	10		

판매번호	판매코드	판매일	매장코드	수량
100001	AA01	20180902	S001	50
100002	AA03	20180902	S002	40
100003	AA04	20180902	S002	20
100004	AA04	20180902	S001	30
100005	AA05	20180902	S004	40
100006	AA03	20180902	S004	30
100007	AA01	20180902	S003	40
100008	AA04	20180902	S004	10
100009	AA01	20180902	S003	20
100010	A005	20180902	S003	30
100011	AA01	20180902	S001	40
100012	AA03	20180902	S002	50
100013	AA04	20180902	S002	50
100014	AA05	20180902	S004	20
100015	AA01	20180902	S003	30

```

CREATE TABLE tbl_salelist_01(
  saleno NUMBER(10) PRIMARY KEY,
  pcode VARCHAR2(10) NOT NULL,
  saledate DATE,
  scode VARCHAR2(10) NOT NULL,
  amount NUMBER(10)
);
INSERT INTO tbl_salelist_01 VALUES(100001, 'AA01', '20180902', 'S001', 50);
INSERT INTO tbl_salelist_01 VALUES(100002, 'AA03', '20180902', 'S002', 40);
INSERT INTO tbl_salelist_01 VALUES(100003, 'AA04', '20180902', 'S002', 20);
INSERT INTO tbl_salelist_01 VALUES(100004, 'AA04', '20180902', 'S001', 30);
INSERT INTO tbl_salelist_01 VALUES(100005, 'AA05', '20180902', 'S004', 40);
INSERT INTO tbl_salelist_01 VALUES(100006, 'AA03', '20180902', 'S004', 30);
INSERT INTO tbl_salelist_01 VALUES(100007, 'AA01', '20180902', 'S003', 40);
INSERT INTO tbl_salelist_01 VALUES(100008, 'AA04', '20180902', 'S004', 10);
INSERT INTO tbl_salelist_01 VALUES(100009, 'AA01', '20180902', 'S003', 20);
INSERT INTO tbl_salelist_01 VALUES(100010, 'AA05', '20180902', 'S003', 30);
INSERT INTO tbl_salelist_01 VALUES(100011, 'AA01', '20180902', 'S001', 40);
INSERT INTO tbl_salelist_01 VALUES(100012, 'AA03', '20180902', 'S002', 50);
INSERT INTO tbl_salelist_01 VALUES(100013, 'AA04', '20180902', 'S002', 50);
INSERT INTO tbl_salelist_01 VALUES(100014, 'AA05', '20180902', 'S004', 20);
INSERT INTO tbl_salelist_01 VALUES(100015, 'AA01', '20180902', 'S003', 30);

```

## index.jsp

index.jsp 에서는 header.jsp 와 footer.jsp를 추가해줘야 한다. jsp에서 다른 jsp 페이지를 추가하는 것은 다음과 같은 문법을 사용한다. jsp를 불러오는 방법은 여러 가지 있지만 과정평가 실기시험 준비를 위해서는 아래를 기억해두도록 하자.

```
<jsp:include page="~~~.jsp"></jsp:include>
```

그 이후에는 이전에 만들어 두었던 패키지를 import 해보자. 결과적으로 index.jsp는 다음과 같은 모양이 된다.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@page import="java.sql.*"%>
<%@page import="DBPKG.Util"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>매장별 커피 판매관리</title>
<link rel="stylesheet" href="css.css" />
</head>
<body>

    <jsp:include page="header.jsp"></jsp:include>

    <section></section>

    <jsp:include page="footer.jsp"></jsp:include>

</body>
</html>
```

이때 위의 소스코드를 전체 선택하여 복사한다. 해당 부분은 **모든 서버페이지에 동일하게 들어가는 내용이므로** 지금껏 만들어두었던 insert.jsp , select.jsp , select\_shop.jsp , select\_product.jsp 에 붙여넣기 해준다. 여기까지 완료가 되었다면 이제 index.jsp 안의 내용을 만들어주도록 하자.

매장별 커피 판매관리 ver 1.0					<header>
판매등록	판매현황	매장별판매액	상품별판매액	홈으로	<nav>
<div>매장별 커피 판매관리 프로그램</div> <p>매장별 커피 판매를 관리하기 위한 프로그램이다.</p> <ol style="list-style-type: none"><li>1. 상품테이블, 매장테이블, 판매테이블을 추가한다.</li><li>2. 판매등록, 판매현황, 매장별판매액, 상품별판매액 페이지를 작성한다.</li><li>3. 올바르게 구현되었는지 확인한다.</li></ol>					<section>
Copyright © 2018 All right reserved Semyeong High School					<footer>

다음과 같은 내용을 만들기 위해서는 아래와 같이 소스 코드를 작성하면 된다.

(section 태그 안의 내용만 작성되었음.)

```
<section>
  <h3>매장별 커피 판매관리 프로그램</h3>

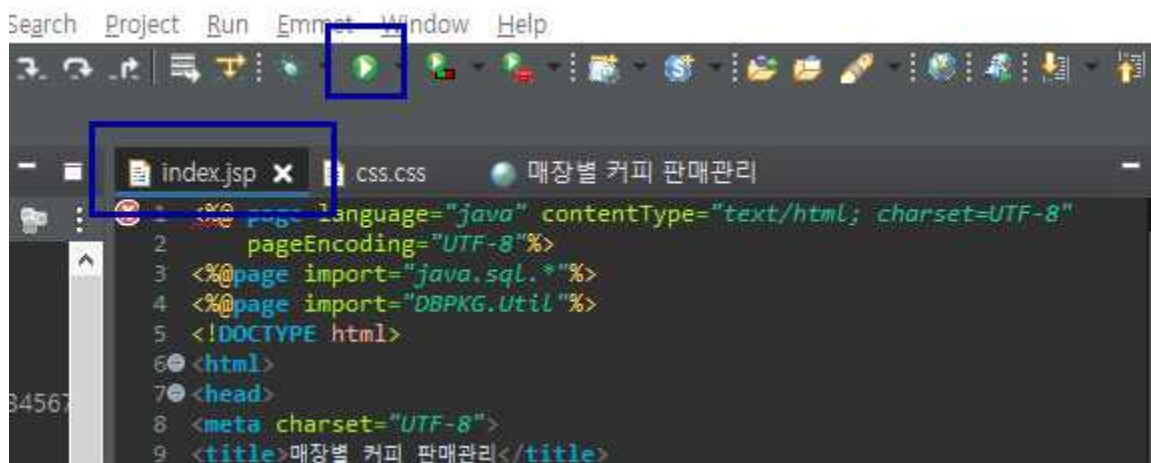
  <pre>
매장별 커피 판매를 관리하기 위한 프로그램이다.

1. 상품테이블, 매자테이블, 판매테이블을 추가한다.
2. 판매등록, 판매현황, 매장별판매액, 상품별판매액 페이지를 작성한다.
3. 올바르게 구현되었는지 확인하다.

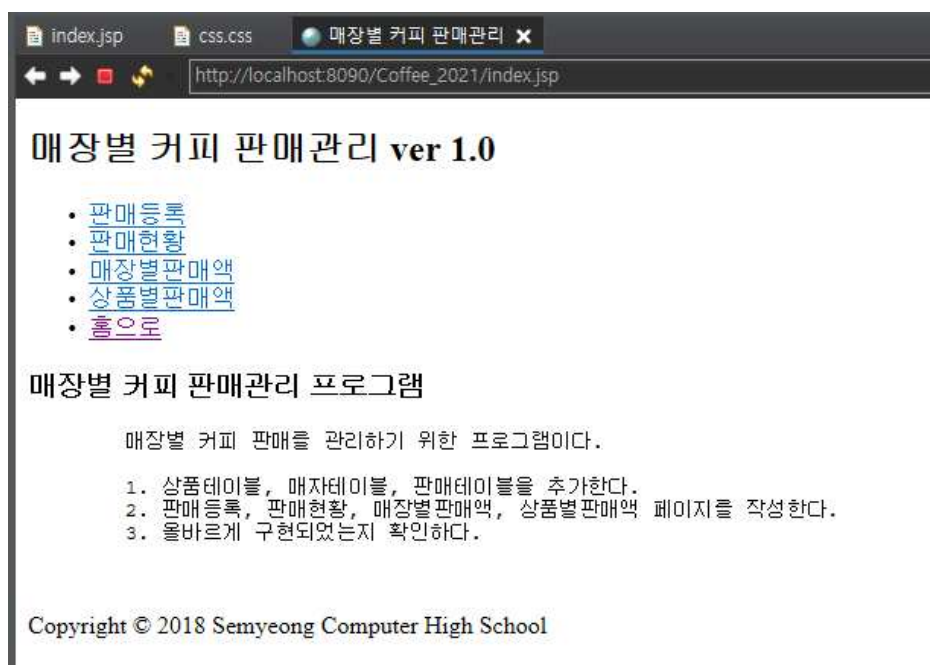
  </pre>

</section>
```

정상적으로 동작하는지 알아보기 위해서는 희망하는 페이지를 선택한 뒤 위에 있는 재생 버튼을 눌러서 실행해 보면 된다.



css를 아직 하지 않았기 때문에 다음과 같이 나온다면 정상적으로 나오는 것이다.





마지막으로 현재 페이지를 나타내는 밑줄을 위해서 다음과 같이 text-decoration : underline을 추가하면 된다. 아래의 소스코드가 완성된 index.jsp이다.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@page import="java.sql.*"%>
<%@page import="DBPKG.Util"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>매장별 커피 판매관리</title>
<link rel="stylesheet" href="css.css" />
<style>.nav5{text-decoration:underline;}</style>
</head>
<body>

    <jsp:include page="header.jsp"></jsp:include>

    <section>
        <h3>매장별 커피 판매관리 프로그램</h3>

        <pre>
매장별 커피 판매를 관리하기 위한 프로그램이다.

1. 상품테이블, 매자테이블, 판매테이블을 추가한다.
2. 판매등록, 판매현황, 매장별판매액, 상품별판매액 페이지를 작성한다.
3. 올바르게 구현되었는지 확인하다.

        </pre>

    </section>

    <jsp:include page="footer.jsp"></jsp:include>

</body>
</html>
```

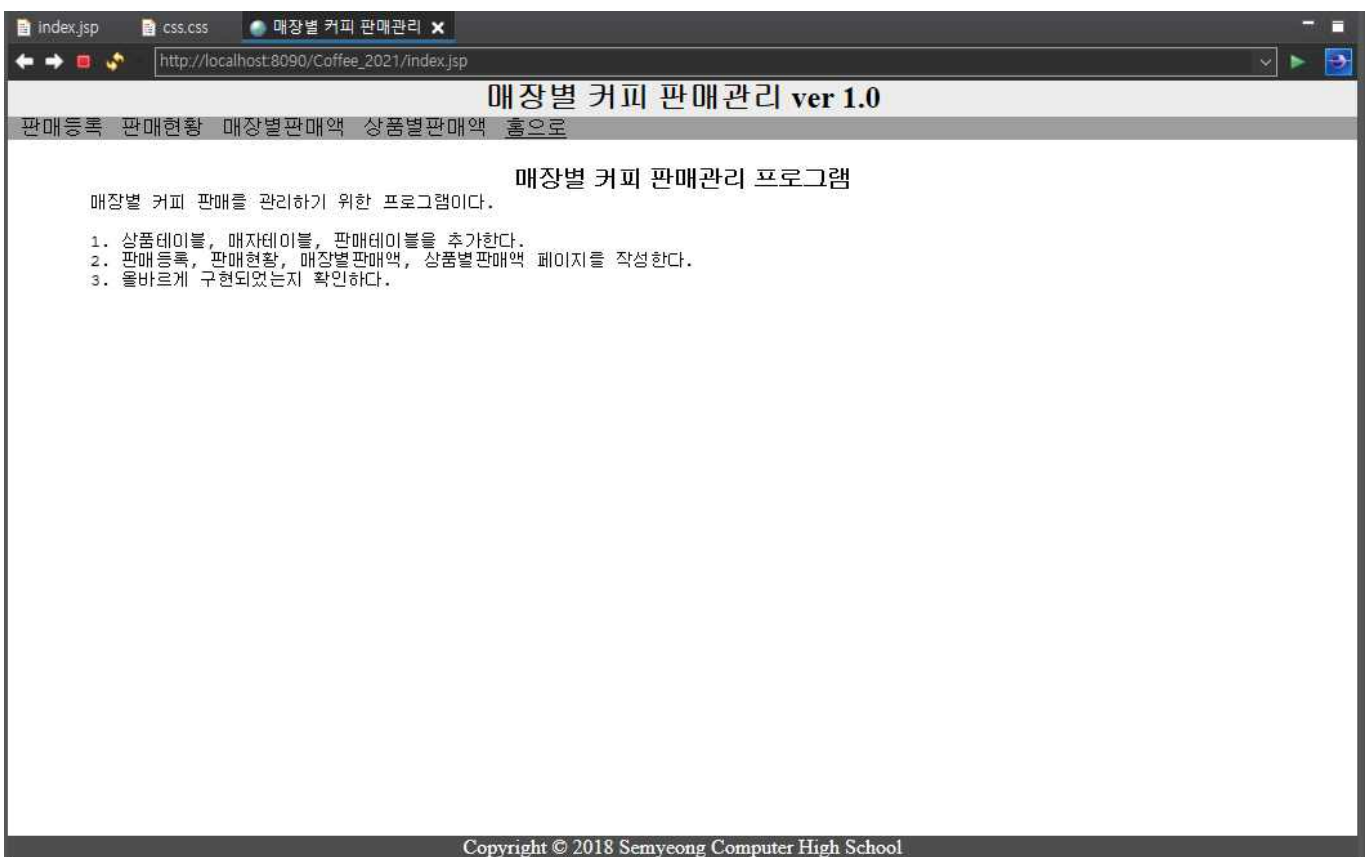
```

@charset "UTF-8";

*{
    padding : 0;
    margin : 0;
    text-decoration : none;
    color : black;
}

header{
    background-color : #e5e5e5;
    text-align : center;
}
nav{ background-color : #999999; }
li{
    display : inline-block;
    margin-left : 10px;
}
footer{
    background-color : #4c4c4c;
    color : white;
    text-align : center;
}
h2, h3{ text-align : center; }
section{
    margin : 20px 0;
    min-height : 500px;
}
form{ text-align : center; }
table{ margin : 0 auto; border : 1 ; }

```



## ■ 판매등록 화면(insert.jsp)

insert 화면을 만들기 위해서는 우선 form과 table 태그를 이용하여 아래의 모양과 같이 만들어줘야 한다.

매장별 커피 판매관리 ver 1.0																
판매등록	판매현황	매장별총판매액	상품별총판매액	홈으로												
<div>판매등록</div> <table><tr><td>판매번호</td><td><input type="text"/></td></tr><tr><td>상품코드</td><td><input type="text"/></td></tr><tr><td>판매날짜</td><td><input type="text"/></td></tr><tr><td>매장코드</td><td><input type="text"/></td></tr><tr><td>판매수량</td><td><input type="text"/></td></tr><tr><td colspan="2"><input type="button" value="등록"/> <input type="button" value="다시쓰기"/></td></tr></table>					판매번호	<input type="text"/>	상품코드	<input type="text"/>	판매날짜	<input type="text"/>	매장코드	<input type="text"/>	판매수량	<input type="text"/>	<input type="button" value="등록"/> <input type="button" value="다시쓰기"/>	
판매번호	<input type="text"/>															
상품코드	<input type="text"/>															
판매날짜	<input type="text"/>															
매장코드	<input type="text"/>															
판매수량	<input type="text"/>															
<input type="button" value="등록"/> <input type="button" value="다시쓰기"/>																
Copyright © 2018 All right reserved Semyeong High School																

우선 <head> 태그 안에 다음과 같이 underline 속성을 추가해준다.

```
<head>
<meta charset="UTF-8">
<title>매장별 커피 판매관리</title>
<link rel="stylesheet" href="css.css" />
<style>.nav1{text-decoration:underline;}</style>
</head>
```

<section> 부분에서 사용되는 내용은 다음과 같다.

아래와 같이 입력하는 페이지에서는 무조건 form 태그가 나와야 하며 form 태그는 데이터를 넘기는데 사용된다. 이때 데이터를 넘기는 방식(method)은 GET과 POST 두 가지가 있는데 과정평가 시험 준비를 할 때는 눈에 보이는 **GET 방식 사용**을 권장한다.

action의 경우 submit 버튼을 눌렀을 때 데이터가 전달되는 페이지를 의미한다. action.jsp를 작성하고 action.jsp를 하나 새로 만들자.

```
<form action="action.jsp" method="get">

<table></table>
    form 태그는 데이터를 넘기는데 사용된다.
</form>    그러므로 table 밖에 form태그를 적어준다.
```

또 하나 주의해야할 점은 input에 name 속성을 주는 것이다. name 은 submit을 통해서 데이터가 넘어갈 때 데이터가 저장되는 변수명이 된다고 생각하면 편하다. 각 input에 name을 줄 때는 임의로 주기 보다는 **데이터베이스 테이블에서의 컬럼명과 동일**하게 주면 추후 프로그래밍하기 편하기 때문에 주어진 문제를 잘 찾아보고 동일하게 값을 주도록 하자.

이번 매장별 커피 판매 관리 프로그램에서는 다음과 같이 section을 구성하면 된다.

```
<section>
<h3>판매등록</h3>

<form action="action.jsp" method="get">
<table>
<tr>
    <td>판매번호</td>
    <td><input type="text" name="saleno"/></td>
</tr>
<tr>
    <td>상품코드</td>
    <td><input type="text" name="pcode"/></td>
</tr>
<tr>
    <td>판매날짜</td>
    <td><input type="text" name="saledate"/></td>
</tr>
<tr>
    <td>매장코드</td>
    <td><input type="text" name="scode"/></td>
</tr>
<tr>
    <td>판매수량</td>
    <td><input type="text" name="amount"/></td>
</tr>
<tr>
    <td colspan="2">
        <input type="submit" value="등 록"/>
        <input type="reset" value="다시쓰기"/>
    </td>
</tr>
</table>
</form>

</section>
```

## ■ 판매등록 화면(action.jsp)

데이터를 넘겨받은 action.jsp에서는 데이터베이스를 연결하여 데이터베이스 테이블에 실제로 데이터를 입력하는 작업을 한다. 실제로 보여지는 페이지는 아니기 때문에 css를 비롯하여 꾸미는 것은 필요없다. 가장 먼저 데이터베이스 연결을 위해 다음과 같이 패키지를 import 해준다.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@page import="java.sql.*"%>
<%@page import="DBPKG.Util"%>
```

그 아래에는 다음과 같이 구성한다.

Util.getParamNN 은 앞서 DBPKG를 구성할 때 작성한 NULL 값 처리를 위한 함수였다. 해당 함수를 이용하여 넘어온 데이터 중에서 NULL 값이 있는 경우 이를 "(공백)"으로 대체하는 역할을 한다.

```
<%
    request.setCharacterEncoding("UTF-8"); 인코딩 설정

    String saleno = Util.getParamNN(request.getParameter("saleno"));
    String pcode = Util.getParamNN(request.getParameter("pcode"));
    String saledate = Util.getParamNN(request.getParameter("saledate"));
    String scode = Util.getParamNN(request.getParameter("scode"));
    String amount = Util.getParamNN(request.getParameter("amount"));

    Connection conn = null;
    Statement stmt = null;

    conn = Util.getConnection();
    stmt = conn.createStatement(); DB 연결
    String sql = "";

    stmt.executeUpdate(sql);

%>
```

여기까지 작성이 완료되었다면 연결된 DB를 바탕으로 SQL 문을 작성하여 실행하는 것이 남았다. stmt.executeUpdate(sql) 함수는 주어진 sql 이란 내용의 쿼리문을 실행시키는 함수이다. 그럼 INSERT를 위한 sql 문을 작성해보도록 하자. 현재 주어진 내용들은 3번째 테이블인 tbl\_salelist\_01 에 넣어줘야 할 데이터이다. 이를 위해서 메모장에 작성해놓았던 INSERT문을 하나 가져오자.

```
INSERT INTO tbl_salelist_01 VALUES(100001, 'AA01', '20180902', 'S001', 50);
```

다음의 SQL문은 이미 검증된 SQL문이다. 이미 실행을 한 번 했었기 때문에 절대 문법적 오류가 없기 때문에 이를 가지고 오면 편하게 수정하여 사용할 수 있다. 여기에 우리가 바꿔줘야 할 것은 VALUES 안에 있는 내용들을 변수들로 바꿔주는 것이다.

```
String sql = "INSERT INTO tbl_salelist_01 VALUES(100001, 'AA01', '20180902', 'S001', 50)";
                                     number형          number형
String sql = "INSERT INTO tbl_salelist_01 VALUES(" + saleno + ", 'AA01', '20180902', 'S001', " + amount + ")";
```



숫자의 경우 따옴표가 필요 없기 때문에 앞뒤로 “ ” 큰 따옴표를 이용하여 글자와 변수 사이를 구별해주고 변수명을 입력해주면 된다. 나머지 VARCHAR형의 경우 글자이기 때문에 ‘ ’ 작은 따옴표를 이용해주어야 하기 때문에 복잡할 수 있다. 이는 다음과 같이 나타낼 수 있다. 큰 따옴표와 작은 따옴표 그리고 반점으로 인하여 sql 변수 안에 쿼리문이 복잡하기 때문에 가장 실수가 많이 나오는 부분 중 하나이다.

```
String sql = "INSERT INTO tbl_salelist_01 VALUES
('"+saleno+"','"+pcode+"','"+saledate+"','"+scode+"','"+amount+"");
```

마지막으로 이번 문제에서 saledate는 date 형식이므로 올바른 형식으로 값을 넣어줘야 한다. 이를 위해 TO\_DATE 함수(암기★)를 사용한다. 결과적으로 완성된 SQL 쿼리문은 다음과 같다.

```
String sql = "INSERT INTO tbl_salelist_01 VALUES('"+saleno+"','"+pcode+"','"+
"TO_DATE('"+saledate+"','YYYY-MM-DD'),'"+scode+"','"+amount+"");
```

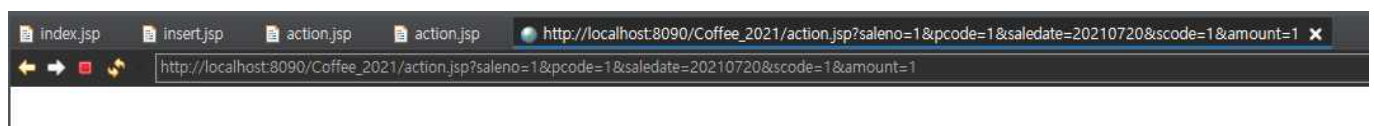
마지막으로 정상적으로 동작하는지 실험을 해보자.

다음과 같이 임의의 값을 넣어준다. 이번 문제에서는 유효성 검사가 없기 때문에 혹 숫자로만 받아야 하는 판매번호나 판매수량에 글자가 들어갈 경우에도 action.jsp로 넘어가게 되고 SQL 문 실행 단계에서 오류가 날 수 있으므로 꼭 가능한 값을 넘겨주도록 하자.

**판매등록**

판매번호	<input type="text" value="1"/>
상품코드	<input type="text" value="1"/>
판매날짜	<input type="text" value="20210720"/>
매장코드	<input type="text" value="1"/>
판매수량	<input type="text" value="1"/>

form 의 method를 GET 방식으로 하였기 때문에 URL을 통해서 실제로 값이 넘어가는 것을 확인할 수 있다. 다음과 같이 넘어간 값을 확인할 수 없다면 insert.jsp 로 돌아가 name이 제대로 설정되었는지 확인하는 과정이 필요하다.



값을 입력하였으나 조회하는 화면이 구현되어 있지 않으므로 정확한 확인을 위해서 데이터베이스에 접근하여 SELECT문을 이용하여 해당 데이터베이스 테이블을 조회해보았다. 다음과 같이 하나의 튜플이 추가된 것이 확인 가능하다면 틀린 곳 없이 프로그래밍을 잘한 것이다 .

```
SQL> select * from tbl_salelist_01;
```

SALENO	PCODE	SALEDATE	SCOE	AMOUNT
100001	AA01	18/09/02	S001	50
100002	AA03	18/09/02	S002	40
100003	AA04	18/09/02	S002	20
100004	AA04	18/09/02	S001	30
100005	AA05	18/09/02	S004	40
100006	AA03	18/09/02	S004	30
100007	AA01	18/09/02	S003	40
100008	AA04	18/09/02	S004	10
100009	AA01	18/09/02	S003	20
100010	AA05	18/09/02	S003	30
100011	AA01	18/09/02	S001	40
SALENO	PCODE	SALEDATE	SCOE	AMOUNT
100012	AA03	18/09/02	S002	50
100013	AA04	18/09/02	S002	50
100014	AA05	18/09/02	S004	20
100015	AA01	18/09/02	S003	30
1	1	21/07/20	1	1

16 rows selected.

마지막으로 이번 기출문제에서는 주어지지 않았으나 대부분의 기출 문제에서는 입력을 완료한 이후에 특정 페이지로 돌아가도록 만들라는 조건을 주고는 한다. 페이지 이동에 필요한 함수는 다음과 같다.

```
response.sendRedirect("페이지주소");
```

해당 함수를 먼저 작성할 경우에는 GET을 통해서 정확한 데이터가 전달되었는지 확인이 어려우므로 입력이 잘 들어가는 것을 확인한 이후에 작성해주면 된다. 이번 프로젝트에서는 따로 주어지지 않았기 때문에 insert.jsp(입력창)으로 돌아가도록 아래와 같이 페이지를 구성하였다.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@page import="java.sql.*"%>
<%@page import="DBPKG.Util"%>

<%
    request.setCharacterEncoding("UTF-8");

    String saleno = Util.getParamNN(request.getParameter("saleno"));
    String pcode = Util.getParamNN(request.getParameter("pcode"));
    String saledate = Util.getParamNN(request.getParameter("saledate"));
    String scode = Util.getParamNN(request.getParameter("scode"));
    String amount = Util.getParamNN(request.getParameter("amount"));

    Connection conn = null;
    Statement stmt = null;

    conn = Util.getConnection();
    stmt = conn.createStatement();
    String sql = "INSERT INTO tbl_salelist_01 VALUES('"+ saleno + "', '"+ pcode + "', " +
        "TO_DATE('"+ saledate + "', 'YYYY-MM-DD'), '"+ scode + "', "+ amount + ")";

    stmt.executeUpdate(sql);

    response.sendRedirect("insert.jsp");
%>
```

## ■ 판매현황 화면(select.jsp)

다음과 같이 판매현황을 나타내기 위해서는 어떤 데이터베이스 테이블에서 내용들을 가지고 왔는지 확인하는 과정이 필요하다. 판매현황의 경우 상품 테이블(tbl\_product\_01)과 판매 테이블(tbl\_salelist\_01) 테이블 두 개가 동시에 조회되어야 함을 파악했다면 SQL 쿼리문을 고민하는 것이 가장 먼저 해야 될 일이다.

매장별 커피 판매관리 ver 1.0

판매등록   판매현황   매장별총판매액   상품별총판매액   홈으로

판매현황

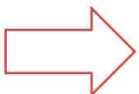
판매번호	상품코드	판매날짜	매장코드	상품명	판매수량	총판매액
100001	AA01	2018-09-02	S001	아메리카노	50	150,000
100002	AA03	2018-09-02	S002	카페라떼	40	160,000
100003	AA04	2018-09-02	S002	카라멜마끼	20	90,000
100004	AA04	2018-09-02	S001	카라멜마끼	30	135,000
100005	AA05	2018-09-02	S004	카푸치노	40	200,000
100006	AA03	2018-09-02	S004	카페라떼	30	120,000
100007	AA01	2018-09-02	S003	아메리카노	40	120,000
100008	AA04	2018-09-02	S004	카라멜마끼	10	45,000
100009	AA01	2018-09-02	S003	아메리카노	20	60,000
100010	AA05	2018-09-02	S003	카푸치노	30	150,000
100011	AA01	2018-09-02	S001	아메리카노	40	120,000
100012	AA03	2018-09-02	S002	카페라떼	50	200,000
100013	AA04	2018-09-02	S002	카라멜마끼	50	225,000
100014	AA05	2018-09-02	S004	카푸치노	20	100,000
100015	AA01	2018-09-02	S003	아메리카노	30	90,000

Copyright © 2018 All right reserved Semyeong High School

여러 개의 테이블을 함께 조회하거나 데이터를 가공해야 할 때에는 INNER JOIN을 이용하면 편리하다. INNER 조인은 다음과 같이 이용할 수 있다. ON 다음에 외래키를 이용하여 연결해주면 된다. 테이블이 세 개일 경우에도 동일하게 가능하다.

SELECT \* FROM tbl\_name INNER JOIN tbl\_age ON id=no;

tbl_name		tbl_age					
id	name	no	age				
1	chris	1	16				
2	sam	2	10				
3	amy	4	15				



id	name	no	age
1	chris	1	16
2	sam	2	10

이번 판매현황에서는 두 개의 테이블을 이용하지만 추후 다른 페이지에서 3개의 테이블을 이용할 수도 있기 때문에 우선은 세 개의 테이블을 연결하는 INNER JOIN 문을 작성해놓으면 좋다.

```
SELECT *
FROM tbl_product_01 tp
INNER JOIN tbl_salelist_01 tsl ON tp.pcode = tsl.pcode
INNER JOIN tbl_shop_01 ts ON tsl.scode = ts.scode ;
```

특히 3개 이상의 테이블을 연결할 때에는 순서가 중요하다. 서로 서로 연결되는 부분이 어디인지 잘 찾아서 올바르게 연결을 해주면 된다. 위에서 tp, tsl, ts의 경우는 별칭으로 tbl\_product\_01을 다 쓰기에는 길어서 해당 SQL 문 안에서 줄여 쓰겠다는 의미이다.

다시 판매현황으로 돌아가서 다시 메모장을 이용하여 SQL문을 작성하도록 하자. 우리가 조회해야 할 내용의 위의 \* 표시에 작성해주면 된다. 메모장에 작성한 이후에 run SQL Command Line 에 입력을 하여 문법적으로 올바른지 확인을 하면 된다.

```
SELECT tsl.salenno, tp.pcode,
TO_CHAR(tsl.saledate, 'YYYY-MM-DD') saledate,
ts.scode, tp.name, tsl.amount,
TO_CHAR(tsl.amount * tp.cost, '999,999,999,999') cost
FROM tbl_product_01 tp
INNER JOIN tbl_salelist_01 tsl ON tp.pcode = tsl.pcode
INNER JOIN tbl_shop_01 ts ON tsl.scode = ts.scode ;
```

```
SQL> SELECT tsl.salenno, tp.pcode,
2  TO_CHAR(tsl.saledate, 'YYYY-MM-DD') saledate,
3  ts.scode, tp.name, tsl.amount,
4  TO_CHAR(tsl.amount * tp.cost, '999,999,999,999') cost
5  FROM tbl_product_01 tp
6  INNER JOIN tbl_salelist_01 tsl ON tp.pcode = tsl.pcode
7  INNER JOIN tbl_shop_01 ts ON tsl.scode = ts.scode ;
```

SALENO	PCODE	SALEDATE	SCODE
NAME		AMOUNT	
COST			
100002	AA03	2018-09-02	S002
카페라떼		40	
160,000			
100003	AA04	2018-09-02	S002
카라멜마끼		20	
90,000			
SALENO	PCODE	SALEDATE	SCODE

그럼 다시 판매현황 코드로 돌아와서 네비게이션의 밑줄 먼저 해결해준다.

```
<head>
<meta charset="UTF-8">
<title>매장별 커피 판매관리</title>
<link rel="stylesheet" href="css.css" />
<style>.nav2{text-decoration:underline;}</style>
</head>
```

다음은 section 영역을 우선 다음과 같이 만들어준다.

```
<section>
<h3>판매현황</h3>

<table>
<tr>
<td>판매번호</td>
<td>상품코드</td>
<td>판매날짜</td>
<td>매장코드</td>
<td>상품명</td>
<td>판매수량</td>
<td>총판매액</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</table>

</section>
```

데이터베이스를 다음과 같이 연결해준다.

```
<%
    request.setCharacterEncoding("UTF-8");

    Connection conn = null;
    Statement stmt = null;
    conn = Util.getConnection();
    stmt = conn.createStatement();
    String sql = "";

    ResultSet rs = stmt.executeQuery(sql);
%>
```



위치는 section 위로 작성하면 된다. 그 이후 sql 안에 위의 쿼리문을 넣어주면 다음과 같이 된다. 실행이 되는 쿼리문임을 이미 확인했기 때문에 한줄로 작성되어 있어도 문제가 없다.

```

1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@page import="java.sql.*"%>
4 <%@page import="DBPKG.Util"%>
5
6 <%
7   request.setCharacterEncoding("UTF-8");
8
9   Connection conn = null;
10  Statement stmt = null;
11  conn = Util.getConnection();
12  stmt = conn.createStatement();
13  String sql = "SELECT tsl.salenno, tp.pcode, TO_CHAR(tsl.saledate, 'YYYY-MM-DD') saledate, ts.scode, tp.name, tsl.amount, TO_CHAR(tsl.amount * tp.cost, '999,999,999,999') co
14
15  ResultSet rs = stmt.executeQuery(sql);
16 %>

```

다음은 내용이다. SQL 쿼리를 통해서 여러줄의 튜플이 가져와지면 이를 출력해야 된다. 출력을 위해서는 반복문을 이용하게 된다. tr 의 내용들이 반복문 안에 들어갈 수 있도록 다음과 같이 작성해준다.

```

<section>
<h3>판매현황</h3>

<table>
<tr>
  <td>판매번호</td>
  <td>상품코드</td>
  <td>판매날짜</td>
  <td>매장코드</td>
  <td>상품명</td>
  <td>판매수량</td>
  <td>총판매액</td>
</tr>

<tr>
  <td></td>
  <td></td>
  <td></td>
  <td></td>
  <td></td>
  <td></td>
  <td></td>
</tr>

</table>

</section>

```

반복되어야 하는 내용  
위아래로 while 문이 감싸도록  
해준다

```

<%   while(rs.next()){ %>
  <tr>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
  </tr>
<% } %>

```

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@page import="java.sql.*"%>
<%@page import="DBPKG.Util"%>
<%

    request.setCharacterEncoding("UTF-8");

    Connection conn = null;
    Statement stmt = null;
    conn = Util.getConnection();
    stmt = conn.createStatement();
    String sql = "SELECT tsl.salenno, tp.pcode, TO_CHAR(tsl.saledate, 'YYYY-MM-DD')
saledate, ts.scode, tp.name, tsl.amount, TO_CHAR(tsl.amount * tp.cost, '999,999,999,999') cost
FROM tbl_product_01 tp INNER JOIN tbl_salelist_01 tsl ON tp.pcode = tsl.pcode INNER JOIN
tbl_shop_01 ts ON tsl.scode = ts.scode";

    ResultSet rs = stmt.executeQuery(sql);
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>매장별 커피 판매관리</title>
<link rel="stylesheet" href="css.css" />
<style>.nav2{text-decoration:underline;}</style>
</head>
<body>
    <jsp:include page="header.jsp"></jsp:include>
    <section>
        <h3>판매현황</h3>

        <table border="1">
            <tr>
                <td>판매번호</td>
                <td>상품코드</td>
                <td>판매날짜</td>
                <td>매장코드</td>
                <td>상품명</td>
                <td>판매수량</td>
                <td>총판매액</td>
            </tr>

            <%
                while(rs.next()){ %>
                <tr>
                    <td><%=rs.getInt("saleno") %></td>
                    <td><%=rs.getString("pcode") %></td>
                    <td><%=rs.getString("saledate") %></td>
                    <td><%=rs.getString("scode") %></td>
                    <td><%=rs.getString("name") %></td>
                    <td><%=rs.getInt("amount") %></td>
                    <td><%=rs.getString("cost") %></td>
                </tr>
            <% } %>

            </table>
        </section>
        <jsp:include page="footer.jsp"></jsp:include>
    </body>
</html>

```

## ■ 매장별판매액 화면(select\_shop.jsp)

매장별판매액 화면과 상품별판매액 화면은 판매현황을 조금만 수정해서 만들 수 있다.

매장별 커피 판매관리 ver 1.0

판매등록   판매현황   매장별판매액   상품별판매액   홈으로

매장별 판매액

매장코드	매장명	매장별 판매액
S001	강남점	405,000
S002	강서점	675,000
S003	강동점	420,000
S004	강북점	465,000

Copyright © 2018 All right reserved Semyeong High School

이미 앞서서 INNER JOIN 문을 만들어 놓았으며 해당 INNER JOIN은 세 개의 데이터베이스 테이블을 모두 사용하는 것이기 때문에 어떤 테이블에서 데이터를 필요로 하는지 고려하지 않고 사용해도 된다

```
SELECT *  
FROM tbl_product_01 tp  
INNER JOIN tbl_salelist_01 tsl ON tp.pcode = tsl.pcode  
INNER JOIN tbl_shop_01 ts ON tsl.scode = ts.scode ;
```

다만 역기서 \* 안에 들어갈 내용만 달리해주면 되는 것이다.

매장 코드는 ts.scode, 매장명은 ts.sname을 이용하면 되며 매장별 판매액 계산을 위하여 GROUP BY를 이용해야 된다. 이때 매장코드 순으로 정렬되어 있으므로 매장코드의 오름차순으로 정렬하면 된다. 이때도 마찬가지로 작성한 SQL 쿼리문을 먼저 데이터베이스에 직접 입력을 해 본 다음 문제가 없으면 그대로 사용하면 된다.

```
SELECT ts.scode, ts.sname,  
TO_CHAR(SUM(tp.cost * tsl.amount), '999,999,999,999') cost  
FROM tbl_product_01 tp  
INNER JOIN tbl_salelist_01 tsl ON tp.pcode = tsl.pcode  
INNER JOIN tbl_shop_01 ts ON tsl.scode = ts.scode  
GROUP BY ts.scode, ts.sname  
ORDER BY 1;
```

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@page import="java.sql.*"%>
<%@page import="DBPKG.Util"%>
<%

    request.setCharacterEncoding("UTF-8");

    Connection conn = null;
    Statement stmt = null;

    conn = Util.getConnection();
    stmt = conn.createStatement();
    String sql = "SELECT ts.scode, ts.sname, TO_CHAR(SUM(tp.cost * tsl.amount),
'999,999,999,999') cost FROM tbl_product_01 tp INNER JOIN tbl_salelist_01 tsl ON tp.pcode =
tsl.pcode INNER JOIN tbl_shop_01 ts ON tsl.scode = ts.scode GROUP BY ts.scode, ts.sname
ORDER BY 1";

    ResultSet rs = stmt.executeQuery(sql);
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>매장별 커피 판매관리</title>
<link rel="stylesheet" href="css.css" />
<style>.nav3{text-decoration:underline;}</style>
</head>
<body>

    <jsp:include page="header.jsp"></jsp:include>

    <section>
    <h3>매장별 판매액</h3>

    <table border="1">
    <tr>
        <td>매장코드</td>
        <td>매장명</td>
        <td>매장별 판매액</td>
    </tr>
    <%
        while(rs.next()){ %>
        <tr>
            <td><%=rs.getString("scode") %></td>
            <td><%=rs.getString("sname") %></td>
            <td><%=rs.getString("cost") %></td>
        </tr>
    <% } %>
        </table>

    </section>

    <jsp:include page="footer.jsp"></jsp:include>

</body>
</html>

```

## ■ 상품별판매액 화면(select\_product.jsp)

이번에도 역시 동일하게 하면 된다. 네비게이션에서 밑줄을 처리해주고 SQL문을 만든 다음에 출력하는 것을 만들면 된다. 과정평가 실기시험을 준비하는 과정에서 SQL 문을 작성하는 부분만 다르고 나머지는 거의 동일하게 진행된다.

매장별 커피 판매관리 ver 1.0

판매등록   판매현황   매장별판매액   상품별판매액   홈으로

상품별 판매액

상품코드	상품명	상품별판매액
AA01	아메리카노	540,000
AA03	카페라떼	480,000
AA04	카라멜마끼	495,000
AA05	카푸치노	450,000

Copyright © 2018 All right reserved Semyeong High School

```
SELECT tp.pcode, tp.name,  
TO_CHAR(SUM(tp.cost * tsl.amount), '999,999,999,999') cost  
FROM tbl_product_01 tp  
INNER JOIN tbl_salelist_01 tsl ON tp.pcode = tsl.pcode  
INNER JOIN tbl_shop_01 ts ON tsl.scode = ts.scode  
GROUP BY tp.pcode, tp.name  
ORDER BY 1;
```

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@page import="java.sql.*"%>
<%@page import="DBPKG.Util"%>
<%
    request.setCharacterEncoding("UTF-8");

    Connection conn = null;
    Statement stmt = null;
    conn = Util.getConnection();
    stmt = conn.createStatement();
    String sql = "SELECT tp.pcode, tp.name, TO_CHAR(SUM(tp.cost * tsl.amount),
'999,999,999,999') cost FROM tbl_product_01 tp INNER JOIN tbl_salelist_01 tsl ON tp.pcode =
tsl.pcode INNER JOIN tbl_shop_01 ts ON tsl.scode = ts.scode GROUP BY tp.pcode, tp.name
ORDER BY 1";

    ResultSet rs = stmt.executeQuery(sql);
%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>매장별 커피 판매관리</title>
<link rel="stylesheet" href="css.css" />
<style>.nav4{text-decoration:underline;}</style>
</head>
<body>

    <jsp:include page="header.jsp"></jsp:include>

    <section>
    <h3>상품별 판매액</h3>

    <table border="1">
    <tr>
        <td>상품코드</td>
        <td>상품명</td>
        <td>상품별판매액</td>
    </tr>

    <%
        while(rs.next()){ %>
        <tr>
            <td><%=rs.getString("pcode") %></td>
            <td><%=rs.getString("name") %></td>
            <td><%=rs.getString("cost") %></td>
        </tr>
    <% } %>

    </table>

    </section>

    <jsp:include page="footer.jsp"></jsp:include>

</body>
</html>

```