

// 정보처리 산업기사 외부평가 대비

// 보여줄 페이지를 (header+nav),(section),(footer) 의 3등분으로 나누고, 내용들을 추가하여 css 로 꾸며 봤어요.

// index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ include file="header2.jsp"%>
<h2>쇼핑몰 회원관리 프로그램</h2>
<div>
    쇼핑몰 회원정보와 회원매출정보 데이터베이스를 구축하고
    회원관리 프로그램을 작성하는 프로그램이다.<br>
    프로그램 작성 순서<br>
    1. 회원정보 테이블을 생성한다. <br>
    2. 매출정보 테이블을 생성한다. <br>
    3. 회원정보, 매출정보 테이블에 제시된
    문제지의 참조데이터를 추가 생성한다. <br>
    4. 회원정보 입력 화면 프로그램을 작성한다. <br>
    5. 회원정보 조회 프로그램을 작성한다. <br>
    6. 회원매출정보 조회 프로그램을 작성한다.
</div>
<%@ include file="footer2.jsp"%>
```

// footer2.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
</div>
</section>
<footer class="footer">
    <div>HRDKOREA Copyright&copy;2016 All rights reserved.
        Human
        Resuources Development Service of Korea.</div>
</footer>
</body>
</html>
```

// header2.jsp

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

```

<html>
<head>
<meta charset="UTF-8">
<link rel="stylesheet" href="./css/common.css">
<title>쇼핑몰 회원 관리 Ver 1.0</title>
</head>
<body>
  <header class="header">
    <div>쇼핑몰 회원 관리 Ver 1.0</div>
  </header>
  <nav class="nav">
    <ul>
      <li><a href="#">회원등록</a></li>
      <li><a href="#">회원목록 조회/수정</a></li>
      <li><a href="#">회원매출조회</a></li>
      <li><a href="#">홈으로.</a></li>
    </ul>
  </nav>
  <section class="section">
    <div>

```

```
// css/common.css
```

```

html, body {
  margin: 0;
  padding: 0;
  height: 100%;
}

.header {
  width: 100%;
  height: 10%;
  background: blue;
  text-align: center;
  font-size: xx-large;
  display: table; /* 표처럼 보이게 변경. */
}

.header div {
  color: white;
  height: 100%; /* 전체 중에서 할당된 10% 모두(100%) 사용. */
  display: table-cell; /* div 를 테이블의 1칸 처럼 표시. */
  vertical-align: middle; /* 수직의 중앙 배치 */
}

.nav {
  width: 100%;
  height: 5%;
  background: aqua;
}

.section {

```

```

        width: 100%;
        height: 75%;
        background: olive;
    }

.section div h2 {
    margin-top: 0px;
    text-align: center;
    padding-top: 20px;
}

.footer {
    width: 100%;
    height: 10%;
    background: blue;
}

ul { /* unordered List 순서 없는 목록 */
    list-style-type: none; /* 목록 앞에 모양 지우기 */
    margin: 0;
    padding: 0;
    overflow: hidden; /* 넘치는 내용이 있다면 표시 안함. */
}

li {
    float: left; /* 둥둥떠서 왼쪽으로 붙이기 */
}

li a {
    display: block; /* 인라인 레벨을 블록레벨로 디스플레이 변경. */
    color: white;
    text-align: center;
    padding: 10px 16px;
    text-decoration: none; /* 링크 태그의 꾸미기 없음. */
    vertical-align: middle; /* 블록레벨에서 상하 가운데 위치. */
}

```

// 실습용 테이블 구성.

sql plus 에서 컬럼 폭 조정

```
set linesize 1000
```

```
col (컬럼명) format a(원하는 너비)
```

예)

system 계정에서 member_tb1_02 컬럼폭 수정.

```
conn system/1234
select * from member_tbl_02;
set linesize 1000
-- 창의 크기를 좌우로 충분히 늘립니다.
-- 아직 address 컬럼이 너무 길어서 별로예요.
col address format a40
col custname format a10
select * from member_tbl_02;
-- ctrl+c (복사) ctrl+v (붙여넣기) 잘 되네요.
```

비슷한 내용 반복 사용시,

메모장에 작성하고 sql command line 에서 붙여넣기 하면 편해요.

```
// 문제에서 요구한 데로 system / 1234 이용을 위해서
```

```
// sys 계정 로그인후, system 계정의 비밀번호 변경.
```

```
alter user system identified by "1234";
```

```
-- system 계정의 비밀번호를 1234 로 변경.
```

```
// member_tbl_02 생성.
```

```
create table member_tbl_02(
```

```
  custno number(6) primary key,
```

```
  custname varchar2(20),
```

```
  phone varchar2(13),
```

```
  address varchar2(60),
```

```
  joindate date,
```

```
  grade char(1),
```

```
  city char(2)
```

```
);
```

```
// 시퀀스 생성 (시퀀스 오류시 수정보다는 삭제후 재 생성 추천)
```

```
create sequence custno_seq
```

```
start with 100001;
```

```
// 샘플 데이터 추가.
```

```
insert into member_tbl_02 values
```

```
(custno_seq.nextval, '김행복','010-1111-2222','서울 동대문구 휘경1동','20151202','A','01');
```

```
insert into member_tbl_02 values
```

```
(custno_seq.nextval, '이축복','010-1111-3333','서울 동대문구 휘경2동','20151206','B','01');
```

```
insert into member_tbl_02 values
```

```
(custno_seq.nextval, '장민음','010-1111-4444','울릉군 울릉읍 독도1리','20151001','B','30');
```

```
insert into member_tbl_02 values
```

```
(custno_seq.nextval, '최사랑','010-1111-5555','울릉군 울릉읍 독도2리','20151113','A','30');
```

```
insert into member_tbl_02 values
```

```
(custno_seq.nextval, '진평화','010-1111-6666','제주도 제주시 외나무골','20151225','B','60');
```

```
insert into member_tbl_02 values
```

```
(custno_seq.nextval, '차공단','010-1111-7777','제주도 제주시 감나무골','20151211','C','60');
```

```
// 혹시 입력값이 틀리면 alter 이용해서 수정후, 커밋.
```

// money_tbl_02, custno 컬럼은 외래키 참조 설정 안함. (1명의 고객이 여러 번 구매)

```
create table money_tbl_02(  
  
custno number(6),  
  
salenol number(8),  
  
pcost number(8),  
  
amount number(4),  
  
price number(8),  
  
pcode varchar2(4),  
  
sdate date,  
  
primary key(custno,salenol)  
  
);
```

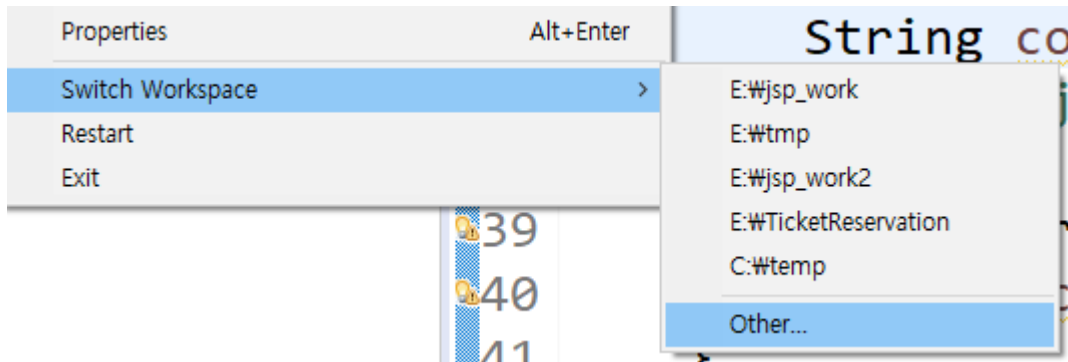
```
insert into money_tbl_02 values(100001, 20160001, 500, 5, 2500, 'A001', '20160101');  
insert into money_tbl_02 values(100001, 20160002, 1000, 4, 4000, 'A002', '20160101');  
insert into money_tbl_02 values(100001, 20160003, 500, 3, 1500, 'A008', '20160101');  
insert into money_tbl_02 values(100002, 20160004, 2000, 1, 2000, 'A004', '20160102');  
insert into money_tbl_02 values(100002, 20160005, 500, 1, 500, 'A001', '20160103');  
insert into money_tbl_02 values(100003, 20160006, 1500, 2, 3000, 'A003', '20160103');  
insert into money_tbl_02 values(100004, 20160007, 500, 2, 1000, 'A001', '20160104');  
insert into money_tbl_02 values(100004, 20160008, 300, 1, 300, 'A005', '20160104');  
insert into money_tbl_02 values(100004, 20160009, 600, 1, 600, 'A006', '20160104');  
insert into money_tbl_02 values(100004, 20160010, 3000, 1, 3000, 'A007', '20160106'); commit;
```

// hrdkorea 생성 C:\hrdkorea

// 이클립스 설정.

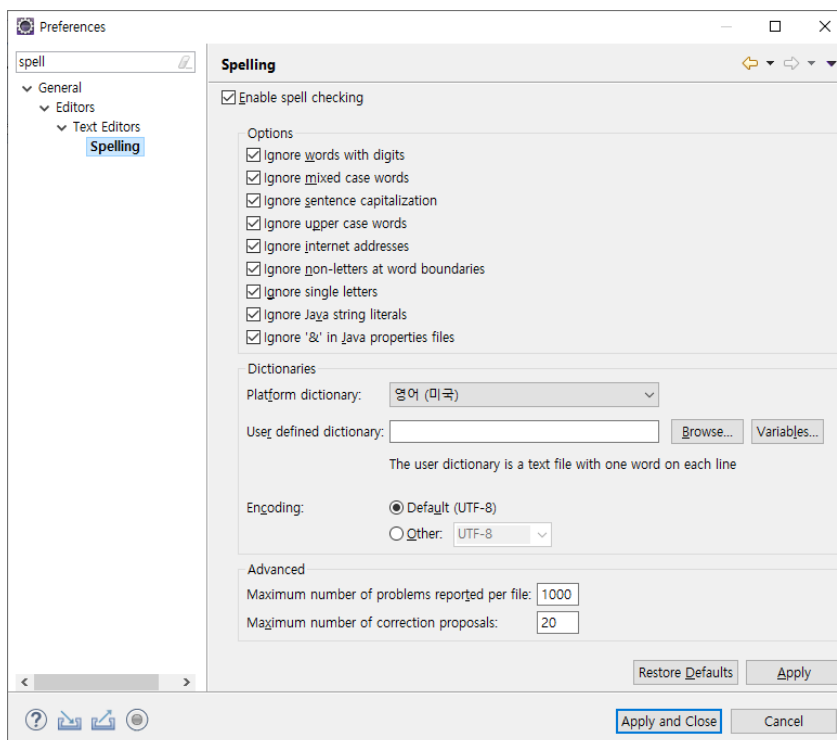
이클립스 실행후, 워크스페이스 변경.

메뉴 > 파일 > switch workspace > other > C:\hrdkorea

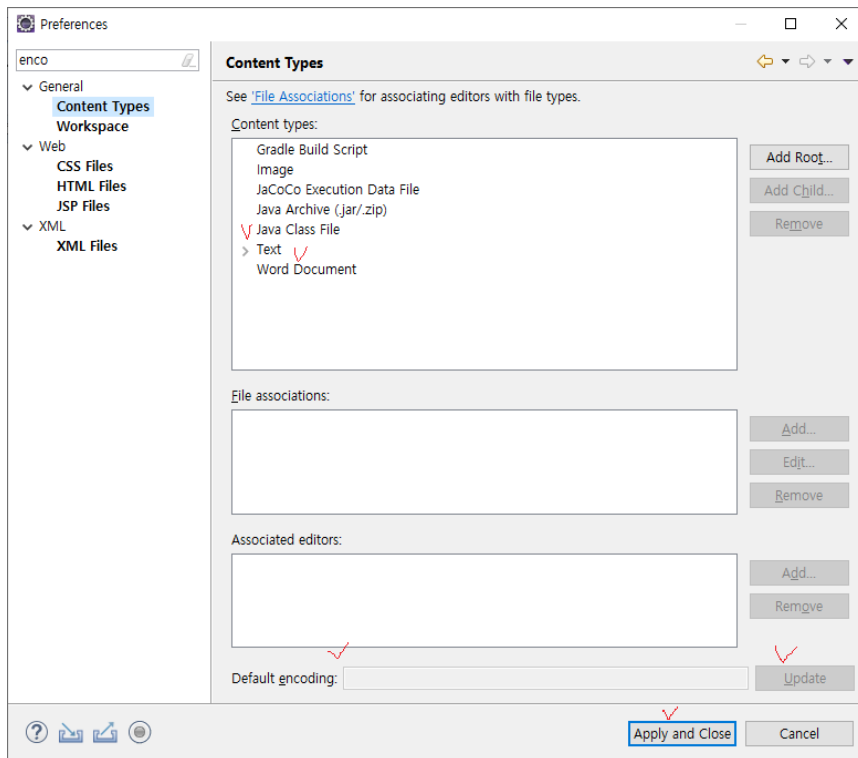


// 엔코딩은 이미 시험장에 설정되어 있음

// encoding : menu > window > preferences >

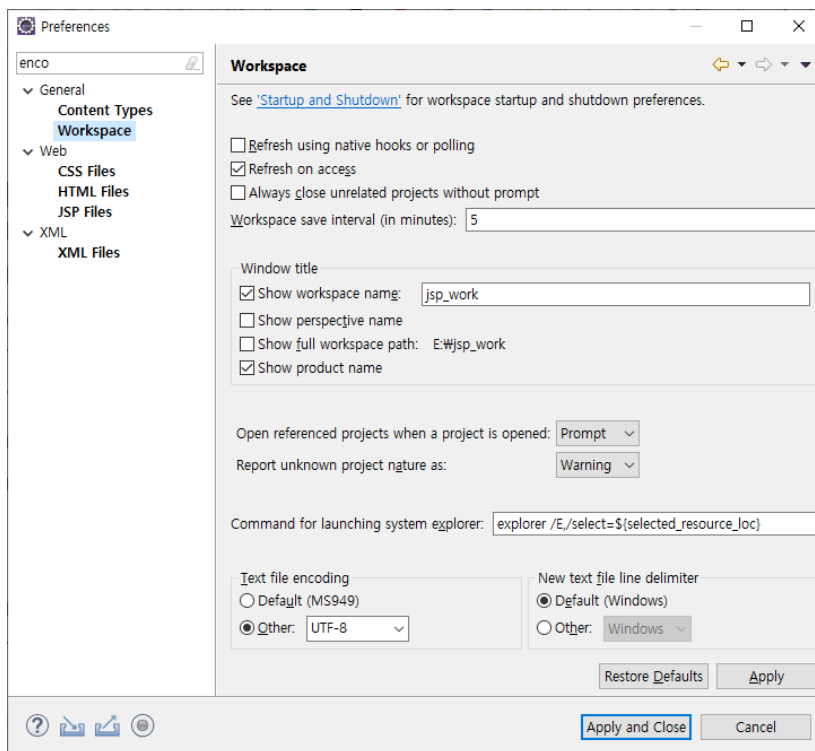


Spell 검색후, utf-8 로 변경.

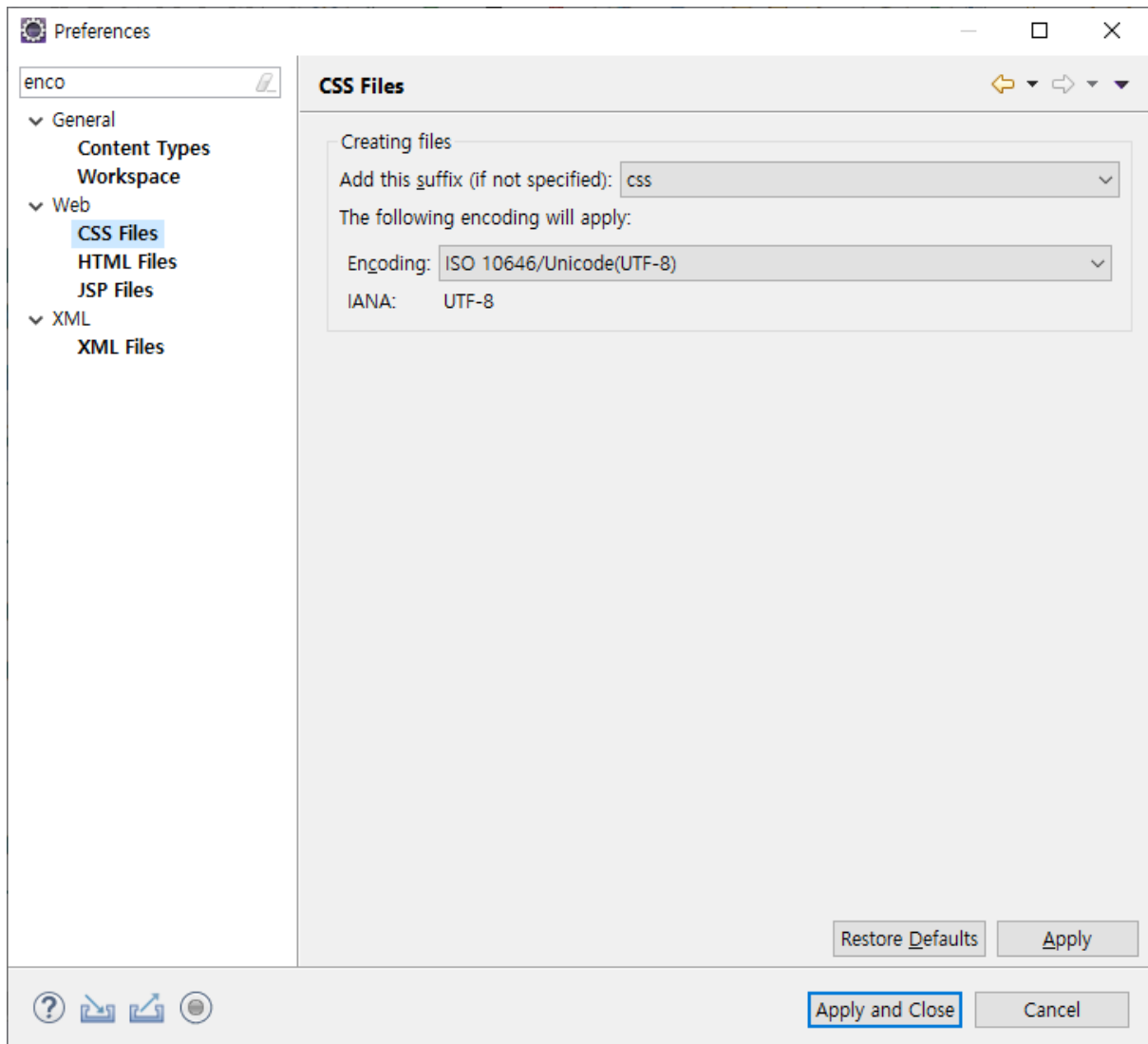


Enco 검색후, content types > java class File > Default encoding : utf-8 > update

Enco 검색후, content types > text > Default encoding : utf-8 > update



Workspace > text file encoding : utf-8 > apply



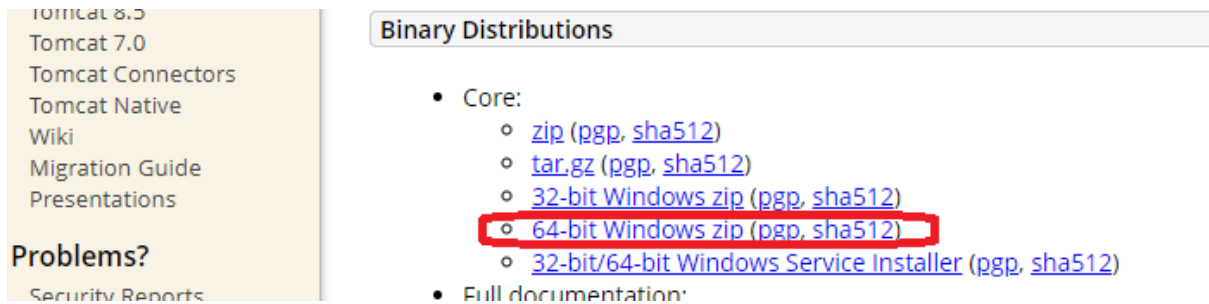
Css Files > encoding : utf-8 > apply

Html, jsp, xml 도 동일 변경후, apply

// 새 프로젝트 생성 : Dynamic web project >> 프로젝트명 (HRD_비번)

// 톱캣은 installer 버전이 아니라, zip 버전 설치. (이미 시험장에 설치되어 있음)

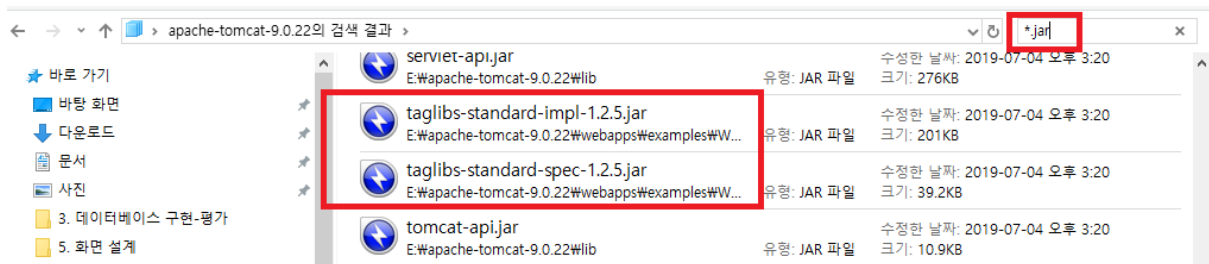
// 인스톨러 버전은 제어판 프로그램 추가 / 제거에서 삭제.



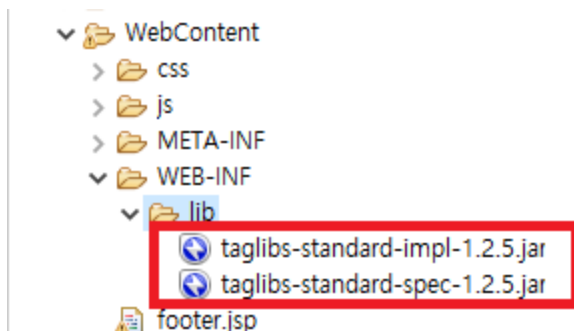
// 톰캣 설치후 이클립스에 서버 연동.

// el 과 jstl 사용을 위한 라이브러리 추가.

// 톰캣 zip 버전 압축 해제한 폴더에서, 우측 상단 검색 *jar



위 이미지 2개 파일 선택 복사후,



작업중인 프로젝트 WebContent > WEB-INF > lib 에 붙여넣기.

// ojdbc6.jar

C:\Woraclexe\app\Woracle\product\11.2.0\server\jdbc\lib

ojdbc6.jar 복사해서

E:\apache-tomcat-9.0.22\lib 붙여넣기. (톰캣 설치 폴더\lib)

// 테스트 서블릿 생성후, 서블릿 오류 발생시, 프로젝트 우클릭 build path > libraries > add library
> server runtime > tomcat 지정.

// 프로젝트 > java resources > src 우클릭

New Java Class

Java Class

✖ Type already exists.

Source folder: HRD_36/src Browse...

Package: vo Browse...

☐ Enclosing type: Browse...

Name: MemberBean

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

패키지 : vo, 파일명 : MemberBean 생성.

package vo;

```

public class MemberBean {
    private int custno;
    private String custname;
    private String phone;
    private String address;
    private String joindate;
    private String grade;
    private String city;

    public int getCustno() {
        return custno;
    }
    public void setCustno(int custno) {
        this.custno = custno;
    }
    public String getCustname() {
        return custname;
    }
    public void setCustname(String custname) {
        this.custname = custname;
    }
    public String getPhone() {
        return phone;
    }
    public void setPhone(String phone) {
        this.phone = phone;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
    public String getJoindate() {
        return joindate;
    }
    public void setJoindate(String joindate) {
        this.joindate = joindate;
    }
    public String getGrade() {
        return grade;
    }
    public void setGrade(String grade) {
        this.grade = grade;
    }
    public String getCity() {
        return city;
    }
    public void setCity(String city) {
        this.city = city;
    }
}


```

// vo 패키지에서 우클릭 ActionForward 클래스 생성.


```
package vo;
```

```
public class ActionForward {  
    // 상태에 따라서 포워딩 시킬지 리다이렉트 시킬지 판단.  
    private boolean isRedirect = false;  
    private String path = null;  
  
    public boolean isRedirect() {  
        return isRedirect;  
    }  
  
    public void setRedirect(boolean isRedirect) {  
        this.isRedirect = isRedirect;  
    }  
  
    public String getPath() {  
        return path;  
    }  
  
    public void setPath(String path) {  
        this.path = path;  
    }  
}
```

// src 우클릭 class

 New Java Class

Java Class

 Type already exists.

Source folder:

Package:

☐ Enclosing type:

Name:

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

Superclass:

Interfaces:

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



패키지 : db, 파일명 : JdbcUtil

```
package db;
```

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

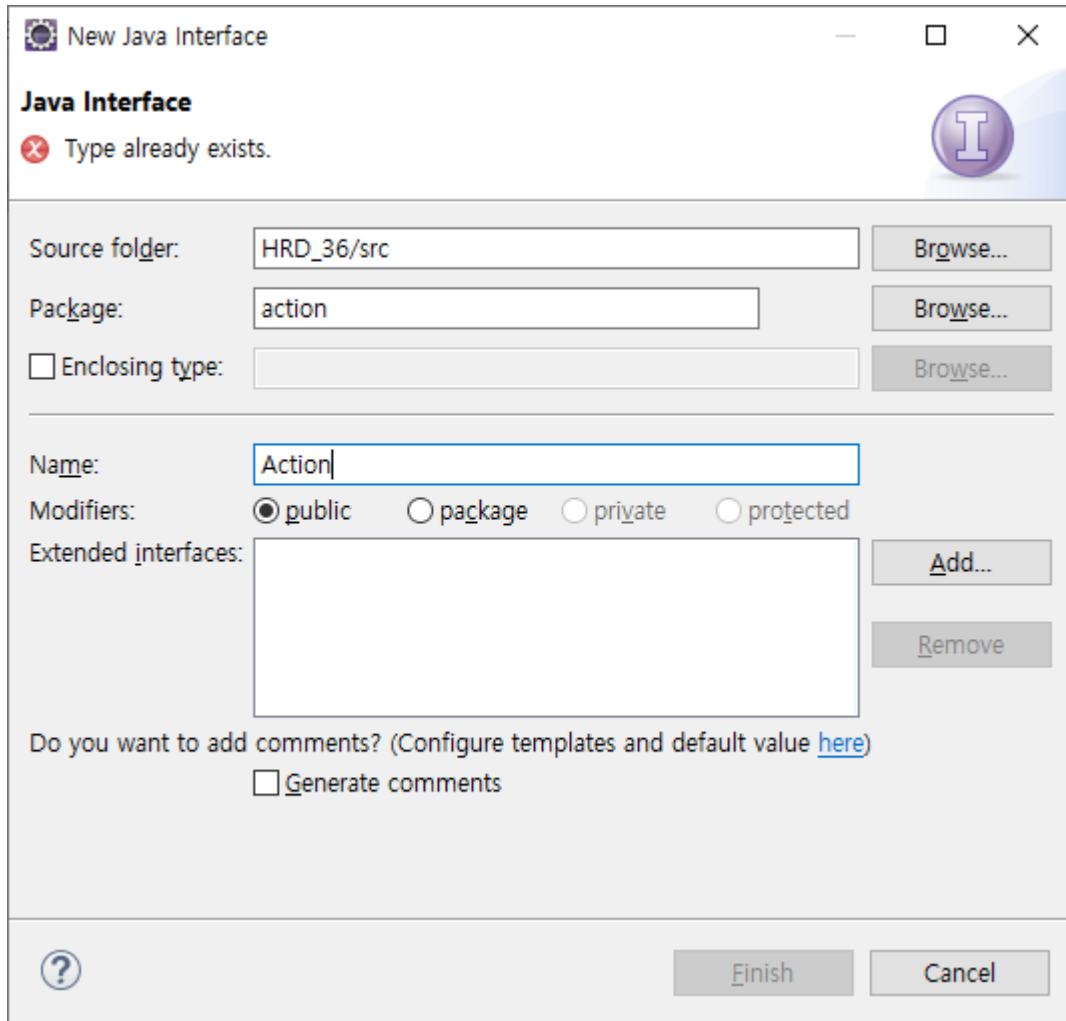
```
public class JdbcUtil {
    public static Connection getConnection()
        throws Exception{
        Class.forName("oracle.jdbc.OracleDriver");
        Connection con
            =DriverManager.getConnection(
                "jdbc:oracle:thin:@//localhost:1521/xe",
```

```

        "system", "1234");
    return con;
}
}

```

// src 우클릭, interface



패키지명 : action, 인터페이스명 : Action

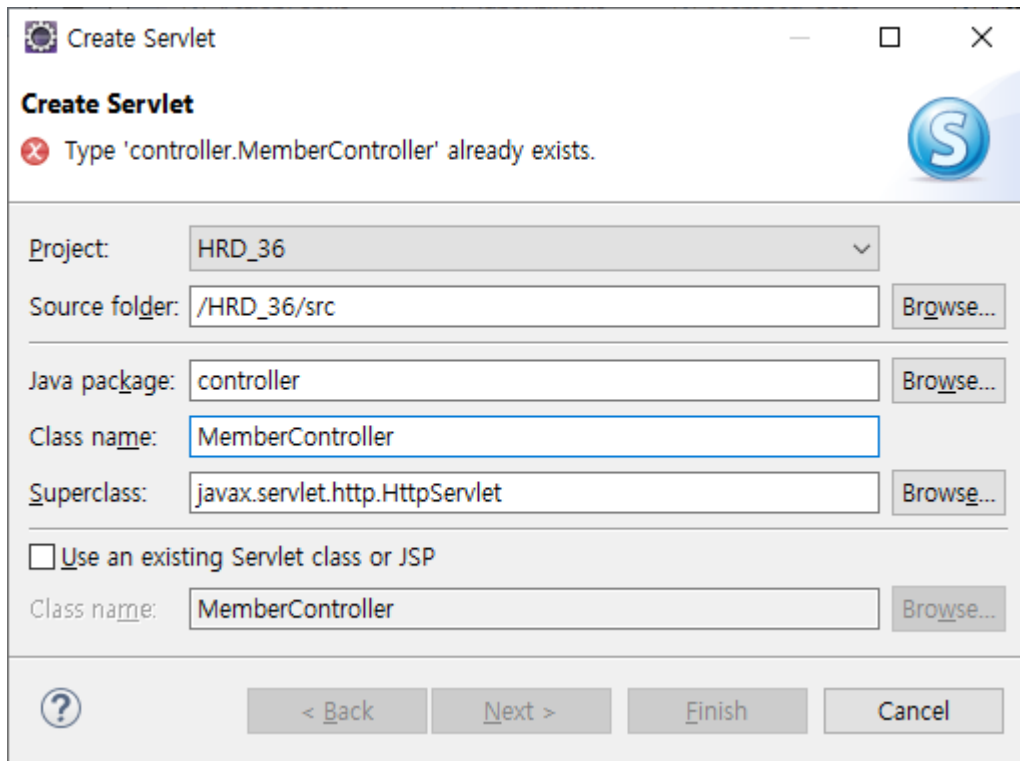
```
package action;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

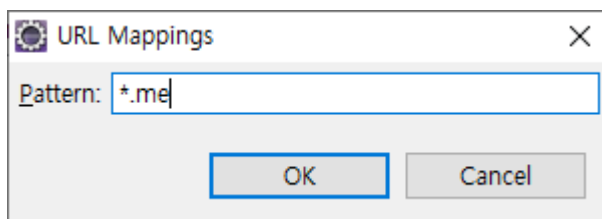
```
import vo.ActionForward;
```

```
public interface Action {
    public ActionForward execute(HttpServletRequest req, HttpServletResponse
resp) throws Exception;
}
```

// src 우클릭, 서블릿.



The 'Create Servlet' dialog box in an IDE. It has a title bar with a gear icon and the text 'Create Servlet'. Below the title bar, there's a red error icon and the message 'Type 'controller.MemberController' already exists.' To the right of this message is a blue circular icon with a white 'S'. The dialog contains several input fields: 'Project:' with a dropdown menu showing 'HRD_36'; 'Source folder:' with a text field containing '/HRD_36/src' and a 'Browse...' button; 'Java package:' with a text field containing 'controller' and a 'Browse...' button; 'Class name:' with a text field containing 'MemberController'; 'Superclass:' with a text field containing 'javax.servlet.http.HttpServlet' and a 'Browse...' button. Below these fields is a checkbox labeled 'Use an existing Servlet class or JSP'. Under this checkbox is another 'Class name:' text field containing 'MemberController' and a 'Browse...' button. At the bottom of the dialog are four buttons: a help icon (?), '< Back', 'Next >', and 'Finish', followed by a 'Cancel' button.



The 'URL Mappings' dialog box. It has a title bar with a gear icon and the text 'URL Mappings'. Below the title bar is a 'Pattern:' text field containing '*.me'. At the bottom are two buttons: 'OK' and 'Cancel'.

// doGet, doPost 선택하고 마침, 아래 코드로 수정.

패키지명 : controller, 클래스명 : MemberController

```
package controller;
```

```
import java.io.IOException;
```

```
import javax.servlet.ServletException;
```

```
import javax.servlet.annotation.WebServlet;
```

```
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
import action.Action;
```

```
import vo.ActionForward;
```

```
@WebServlet("*.me")
```



```

public class MemberController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        doProcess(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        doProcess(request, response);
    }

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

        String RequestURI = request.getRequestURI();
        // localhost:8090/HRD_36/home.jsp
        String contextPath = request.getContextPath();
        // localhost:8090/HRD_36/
        String command = RequestURI.substring(contextPath.length());
        // home.jsp

        ActionForward forward = null;
        Action action = null;
    }
}

```

// home.jsp 를 index.jsp 로 이름 변경.

// db.JdbcUtil.java 코드 추가.

```
package db;
```

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

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

```
import java.sql.ResultSet;
```

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

```
import java.sql.Statement;
```

```

public class JdbcUtil {

    public static Connection getConnection() throws Exception {

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

        Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "1234");

        return con;

    }

    public static void close(Connection con) {

        try {

            con.close();

        } catch (SQLException e) {

            e.printStackTrace();

        }

    }

    public static void close(Statement stmt) {

        try {

            stmt.close();

        } catch (SQLException e) {

            e.printStackTrace();

        }

    }

    public static void close(ResultSet rs) {

        try {

```

```
        rs.close();

    } catch (SQLException e) {

        e.printStackTrace();

    }

}

public static void commit(Connection con) {

    try {

        con.commit();

    } catch (SQLException e) {

        e.printStackTrace();

    }

}

public static void rollback(Connection con) {

    try {

        con.rollback();

    } catch (SQLException e) {

        e.printStackTrace();

    }

}

}
```

// controller.MemberController.java 코드 추가.

```
package controller;
```

```
import java.io.IOException;
```

```
import javax.servlet.RequestDispatcher;
```

```
import javax.servlet.ServletException;
```

```
import javax.servlet.annotation.WebServlet;
```

```
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
import action.Action;
```

```
import action.MemberListAction;
```

```
import vo.ActionForward;
```

```
@WebServlet("*.me")
```

```
public class MemberController extends HttpServlet {
```

```
    private static final long serialVersionUID = 1L;
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
        throws ServletException, IOException {
```

```
        doProcess(request, response);
```

```
    }
```

```
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
```

```
        throws ServletException, IOException {
```

```
        doProcess(request, response);
```

```
    }
```

```
protected void doProcess(HttpServletRequest request, HttpServletResponse response)

    throws ServletException, IOException {

    request.setCharacterEncoding("UTF-8");


    String RequestURI = request.getRequestURI();

    // localhost:8090/HRD_36/home.jsp

    String contextPath = request.getContextPath();

    // localhost:8090/HRD_36/

    String command = RequestURI.substring(contextPath.length());

    // home.jsp


    ActionForward forward = null;

    // 페이지 이동 방식 결정. 리다이렉트인지 포워딩 인지.

    Action action = null;


    if(command.equals("/memberListAction.me")) {

        action=new MemberListAction();

        try {

            forward=action.execute(request, response);

        }catch(Exception e) {

            e.printStackTrace();

        }

    }


    if(forward!=null) {
```

```

        if(forward.isRedirect()) {

            response.sendRedirect(forward.getPath());

            // 리퀘스트 객체를 공유하지 않고 페이지 이동.

        }else {

            RequestDispatcher dispatcher

            =request.getRequestDispatcher(

                forward.getPath());

            dispatcher.forward(request, response);

            // 리퀘스트 객체를 공유하여 페이지 이동.

        }

    }

}

}

```

// action.MemberListAction.java 에 코드 추가.

```
package action;
```

```
import java.util.ArrayList;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
import svc.MemberListService;
```

```
import vo.ActionForward;
```

```
import vo.MemberBean;
```

```

public class MemberListAction implements Action {

    @Override

    public ActionForward execute(HttpServletRequest req

        , HttpServletResponse resp) throws Exception {

        ActionForward forward = null;

        forward=new ActionForward();

        MemberListService memberListService

        = new MemberListService();

        ArrayList<MemberBean> memberList

        = memberListService.getMemberList();

        // 호출한 메소드에서 다시 다오를 호출해서 쿼리를 수행하고,

        // 결과를 어레이리스트로 받아 옴.

        req.setAttribute("memberList", memberList);

        forward.setPath("./list.jsp");

        return forward;

    }

}

```

// svc.MemberListService.java 에 코드 추가.

```
package svc;
```

```
import java.sql.Connection;

import java.util.ArrayList;


import dao.MemberDAO;

import db.JdbcUtil;

import vo.MemberBean;


public class MemberListService {

    public ArrayList<MemberBean> getMemberList() {

        Connection con=null;

        try {

            con=JdbcUtil.getConnection();

//            오라클과 연결 만들기.

        } catch (Exception e) {

            e.printStackTrace();

        }


        MemberDAO memberDAO = MemberDAO.getInstance();

        // 싱글톤, 캘린더가 대표 예제.

        // 객체를 계속 재생성 하는 것이 아니라,

        // 객체를 1개만 생성하고, 참조해서 사용하는 것.

        // 쿼리 작업을 위한 준비.

        memberDAO.setConnection(con);

        // 오라클과의 커넥션을 다오에 전달.

        ArrayList<MemberBean> memberList
```



```

        = memberDAO.selectMemberList();

        // selectMemberList() 수행한 결과를 어레이리스트로 받기.

        JdbcUtil.close(con);

        // 커넥션 객체 닫기.


        return memberList;

    }

}

```

// dao.MemberDAO.java 에 코드 추가.

```
package dao;
```

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

```
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
```

```
import java.util.ArrayList;
```

```
import db.JdbcUtil;
```

```
import vo.MemberBean;
```

```
public class MemberDAO {
```

```
    public static MemberDAO instance;
```

```
    // 다른 클래스에서도 사용할 수 있도록 static 선언.
```

```
    Connection con;// 오라클 연결용
```

```
    PreparedStatement pstmt;// 쿼리 전달용
```

```
    ResultSet rs;// 쿼리 결과 조작용.
```

```
public static MemberDAO getInstance() {  
  
    if (instance == null) {  
  
        instance = new MemberDAO();  
  
    } // 객체가 null이면 생성하고, 아니라면 참조만 리턴.  
  
    return instance;  
  
}
```

```
public void setConnection(Connection con) {  
  
    this.con = con;  
  
    // 서비스에서 커넥션을 참조했고, 그것을 이용하여.  
  
    // 멤버다오에 객체 초기화.  
  
}
```

```
public ArrayList<MemberBean> selectMemberList() {  
  
    String sql = "select * from member_tbl_02 order by custno asc";  
  
    // 회원테이블에서 모든 정보 가져오기. 회원번호의 오름차순으로 정렬.  
  
    ArrayList<MemberBean> memberList = new ArrayList<>();  
  
    // 회원 여러명의 정보 저장용.  
  
    MemberBean mb = null; // 회원 1명의 값을 임시 저장할 공간.  
  
  
    try {  
  
        pstmt = con.prepareStatement(sql);  
  
        rs = pstmt.executeQuery();  
  
  
        if (rs.next()) {
```

```

        do {

            mb = new MemberBean();

            mb.setCustno(rs.getInt("custno"));

            mb.setCustname(rs.getString("custname"));

            mb.setPhone(rs.getString("phone"));

            mb.setAddress(rs.getString("address"));

            mb.setGrade(rs.getString("grade"));

            mb.setCity(rs.getString("city"));

            mb.setJoindate(rs.getString("joindate"));


            memberList.add(mb);

        } while (rs.next());

    }

} catch (Exception ex) {

    ex.printStackTrace();

}finally {

    JdbcUtil.close(rs);

    JdbcUtil.close(pstmt);

}

return memberList;

}

}

```

// WebContent > list.jsp 생성.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ include file="header.jsp"%>
<h2>회원목록 조회/수정</h2>
<table border="1">
    <colgroup>
        <col width="10%">
        <col width="10%">
        <col width="20%">
        <col width="20%">
        <col width="20%">
        <col width="10%">
        <col width="10%">
    </colgroup>
    <!-- 컬그룹을 이용하여 테이블의 컬럼별 너비를 지정. -->
    <thead>
        <tr>
            <th>회원번호</th><!-- 컬럼의 타이틀 -->
            <th>회원성명</th>
            <th>전화번호</th>
            <th>주소</th>
            <th>가입일자</th>
            <th>고객등급</th>
            <th>거주지역</th>
        </tr>
    </thead>
    <tbody>
        <c:forEach var="member" items="${memberList }">
            <tr style="text-align: center;">
                <td>${member.custno }</td>
                <td>${member.custname }</td>
                <td>${member.phone }</td>
                <td>${member.address }</td>
                <td>${member.joindate }</td>
                <td>${member.grade }</td>
                <td>${member.city }</td>
            </tr>
        </c:forEach>
    </tbody>
</table>

<%@ include file="footer.jsp"%>
```

(목록 꾸미기부터)

// static 특징,

```
public class Counter {
```

```

        private static int count = 5;

        public static int getCount() {
            return count;
        }
    }

    public class Down {
        private int cnt = 88;

        public int getCnt() {
            return cnt;
        }
    }

    public class Main {

        public static void main(String[] args) {

            Down d = new Down();
            // d.getCnt();
            System.out.println("d=" + d.getCnt());
            System.out.println("count="+ Counter.getCount());
            // static 으로 선언된 메소드는 객체 생성 없이,
            // 클래스명으로 접근 가능.

        }

    }

```

```

import static des.Counter.*;
public class Main {

    public static void main(String[] args) {

        Down d = new Down();
        // d.getCnt();
        System.out.println("d=" + d.getCnt());
        System.out.println("count="+ getCount());
        // static 으로 선언된 메소드는 객체 생성 없이,
        // 클래스명으로 접근 가능.
        // import static을 선언하면 클래스명도 생략 가능.

    }

}

```

// 목록에서 날짜 표시와 회원구분 표시 변경하기.

```
String sql = "select custno, custname, phone, address,to_char(joindate,'YYYY-MM-DD') joindate,"
```

```
                                + "decode(grade, 'A', 'VIP', 'B', '일반', '직원')  
grade,city from member_tbl_02 order by custno asc";
```

```
// 목록에서 회원번호를 선택하면, 회원수정화면 보여주기
```

```
// list.jsp 의 일부분,
```

```
<a href="memberViewAction.me?id=${member.custno}">  
    <!-- 수정전 회원 정보 보기.  
    어떤 회원을 표시할지는 회원번호로 구별 -->  
    ${member.custno }</a>
```

```
// MemberViewAction.java
```

```
// 클래스 생성시 첫글자는 모두 대문자로 통일.
```

```
package action;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
import svc.MemberViewService;
```

```
import vo.ActionForward;
```

```
import vo.MemberBean;
```

```
public class MemberViewAction implements Action {
```

```
    @Override
```

```
    public ActionForward execute(HttpServletRequest req, HttpServletResponse  
resp) throws Exception {
```

```
        ActionForward forward = null;
```

```
        forward = new ActionForward();
```

```
        String viewId = req.getParameter("id");
```

```
        MemberViewService memberViewService = new MemberViewService();
```

```
        MemberBean member = memberViewService.getMember(viewId);

        // 호출한 메소드에서 다시 다오를 호출해서 쿼리를 수행하고,
        // 결과를 어레이리스트로 받아 옴.

        req.setAttribute("member", member);
        forward.setPath("./member_mod.jsp");

        return forward;
    }

}
```

// MemberViewService.java

```
package svc;
```

```
import static db.JdbcUtil.*;
```

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

```
import dao.MemberDAO;
```

```
import vo.MemberBean;
```

```
public class MemberViewService {
```

```
    public MemberBean getMember(String viewId) {
        Connection con=null;
        try {
            con=getConnection();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

```
    MemberDAO memberDAO = MemberDAO.getInstance();
```

```

// 싱글톤, 캘린더가 대표 예제.

// 객체를 계속 재생성 하는 것이 아니라,

// 객체를 1개만 생성하고, 참조해서 사용하는 것.

// 쿼리 작업을 위한 준비.

memberDAO.setConnection(con);

MemberBean member

= memberDAO.selectMember(viewId);

close(con);

return member;

}

}

```

// MemberDAO.java 에 selectMember() 메소드 추가.

```

public MemberBean selectMember(String id) {
    String sql = "select * from member_tbl_02 where custno=?";
    MemberBean mb = null;
    try {
        pstmt = con.prepareStatement(sql);
        pstmt.setString(1, id);
        rs = pstmt.executeQuery();

        if (rs.next()) {
            mb = new MemberBean();
            mb.setCustno(rs.getInt("custno"));
            mb.setCustname(rs.getString("custname"));
            mb.setPhone(rs.getString("phone"));
            mb.setAddress(rs.getString("address"));
            mb.setGrade(rs.getString("grade"));
            mb.setCity(rs.getString("city"));
            mb.setJoindate(rs.getString("joindate"));
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        close(rs);
        close(pstmt);
    }

    return mb;
}

```

// MemberController.java 에 회원 상세 보기에 대한 코드 추가.

```

else if(command.equals("/memberViewAction.me")) {

```



```

        action=new MemberViewAction();
        try {
            forward=action.execute(request, response);
        }catch(Exception e) {
            e.printStackTrace();
        }
    }
}

```

// member_mod.jsp 에 조회 결과 표시.

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ include file="header.jsp"%>
<h2>홈쇼핑 회원 정보 수정</h2>
<script src="./js/default.js"></script>
<form name="regMem" method="post"
    onsubmit="return checkForm();">
    <!-- action.jsp에서 actino 값을 통하여 sql을 handling 한다. -->
    <table border="1">
        <tr>
            <th>회원번호</th>
            <td width="500px"><input type="text" size="20" name="custno"
                value=${member.custno } readonly></td>
        </tr>
        <tr>
            <th>회원성명</th>
            <td><input type="text" name="custname" size="20"
                value=${member.custname }></td>
        </tr>
        <tr>
            <th>회원전화</th>
            <td><input type="text" name="phone" size="30"
                value=${member.phone }></td>
        </tr>
        <tr>
            <th>회원주소</th>
            <td><input type="text" name="address" size="40"
                value="${member.address }"></td>
        </tr>
        <tr>
            <th>가입일자</th>
            <td><input type="text" name="joindate" size="20"
                value=${member.joindate }></td>
        </tr>
        <tr>
            <th style="padding: 0 20px;">고객등급 [A:VIP, B:일반,
                C:직원]</th>
            <td><input type="text" name="grade" size="20"
                value=${member.grade }></td>
        </tr>
    </table>

```

```

        <th>도시코드</th>
        <td><input type="text" name="city" size="20"
            value=${member.city }></td>
    </tr>
    <tr>
        <td colspan="2" align="center"><button
type="submit">수정</button>
            <button type="button">조회</button></td>
    </tr>
</table>
</form>

<%@ include file="footer.jsp"%>

```

// 수정 버튼을 누르면, 각 항목 유효성 확인.

// js/default.js

```

function checkForm(){
    if(document.regMem.custname.value==''){
//        이름을 입력하지 않았다면,
        window.alert("회원성명이 입력되지 않았습니다.");
        // 알림창을 띄우고,
        document.regMem.custname.value="";
        // 이름창은 공백으로 초기화.
        document.regMem.custname.focus();
        // 이름창에 커서를 이동.
        return false;
        // form action 취소.
    }

    if(document.regMem.phone.value==''){
        window.alert("회원전화가 입력되지 않았습니다.");
        document.regMem.phone.value="";
        document.regMem.phone.focus();
        return false;
    }

    if (document.regMem.address.value == '' ) {
        window.alert("회원주소가 입력되지 않았습니다.");
        document.regMem.address.value = "";
        document.regMem.address.focus();
        return false;
    }

    if (document.regMem.joindate.value == '' ) {
        window.alert("가입일자가 입력되지 않았습니다.");
        document.regMem.joindate.value = "";
        document.regMem.joindate.focus();
        return false;
    }
}

```

```

        if (document.regMem.grade.value == '') {
            window.alert("고객등급이 입력되지 않았습니다.");
            document.regMem.grade.value = "";
            document.regMem.grade.focus();
            return false;
        }

        if (document.regMem.city.value == '') {
            window.alert("도시코드가 입력되지 않았습니다.");
            document.regMem.city.value = "";
            document.regMem.city.focus();
            return false;
        }
    }
}

```

// 오늘은 목록에 링크를 클릭하면, 1명의 회원정보가 보이고, 수정을 누르면 유효성 검사를 하는 부분 이후로 확인해 볼게요.

// member_mod.jsp 의 폼에 action 속성을 추가합니다.

```

<form name="regMem" method="post"
onsubmit="return checkForm();"
action="./memberUpdateAction.me">

```

//MemberController.java 에 분기 처리를 추가합니다.

```

else if(command.equals("/memberUpdateAction.me")) {
    action=new MemberUpdateAction();
    try {
        forward=action.execute(request, response);
    }catch(Exception e) {
        e.printStackTrace();
    }
}
}

```

// MemberUpdateAction.java

```

package action;

```

```

import java.io.PrintWriter;

```

```

import javax.servlet.http.HttpServletRequest;

```

```

import javax.servlet.http.HttpServletResponse;

```

```

import svc.MemberUpdateService;

import vo.ActionForward;

import vo.MemberBean;


public class MemberUpdateAction implements Action {

    // insert 작업과 비슷.

    @Override

    public ActionForward execute(

        HttpServletRequest req,

        HttpServletResponse resp) throws Exception {

        MemberBean member=new MemberBean();

        boolean updateResult=false;

        member.setCustno(Integer.parseInt(

            req.getParameter("custno")));

        member.setCustname(req.getParameter("custname"));

        member.setPhone(req.getParameter("phone"));

        member.setAddress(req.getParameter("address"));

        member.setJoindate(req.getParameter("joindate"));

        member.setGrade(req.getParameter("grade"));

        member.setCity(req.getParameter("city"));

        // 수정 정보를 모두 담아서 회원 객체로 저장.


        MemberUpdateService memberUpdateService

```

```

=new MemberUpdateService();

updateResult

=memberUpdateService.updateMember(member);


ActionForward forward=null;

if(updateResult==false) {

    // 수정이 안 됐을때,

    resp.setContentType("text/html;charset=utf-8");

    PrintWriter out = resp.getWriter();

    out.println("<script>");

    out.println("alert('회원 정보 수정 실패')");

    out.println("history.back()");

    // 이전 화면으로 이동.

    out.println("</script>");

}else {

    // 수정이 됐다면,

    forward=new ActionForward();

    forward.setRedirect(true);

    // 공유하는 값이 없으므로, 리다이렉트로 화면 이동.

    forward.setPath("./memberListAction.me");

    // 이동할 주소 설정.

}

return forward;

}

```

```
}
```

```
// MemberUpdateService.java
```

```
package svc;
```

```
import dao.MemberDAO;
```

```
import vo.MemberBean;
```

```
import static db.JdbcUtil.*;
```

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

```
public class MemberUpdateService {
```

```
    public boolean updateMember(MemberBean member) {
```

```
        Connection con = null;
```

```
        try {
```

```
            con = getConnection();
```

```
        } catch (Exception e) {
```

```
            e.printStackTrace();
```

```
        }
```

```
        boolean updateSuccess = false;
```

```
        MemberDAO memberDAO
```

```
        = MemberDAO.getInstance();
```

```

        memberDAO.setConnection(con);

        int updateCount

        = memberDAO.updateMember(member);

        if(updateCount > 0) {

            updateSuccess=true;

            commit(con);

        }else {

            rollback(con);

        }

        close(con);

        return updateSuccess;

    }

}

// MemberDAO.java 에 updateMember 메소드 추가.

public int updateMember(MemberBean updateMember) {
    int updateCount = 0;
    String sql = "update member_tbl_02 set custname=?, " +
        "phone=?,address=?,joindate=?,grade=?, "
        + "city=? where custno=?";

    try {
        pstmt = con.prepareStatement(sql);
        pstmt.setString(1, updateMember.getCustname());
        pstmt.setString(2, updateMember.getPhone());
        pstmt.setString(3, updateMember.getAddress());
        pstmt.setString(4, updateMember.getJoindate());
        pstmt.setString(5, updateMember.getGrade());
        pstmt.setString(6, updateMember.getCity());
        pstmt.setInt(7, updateMember.getCustno());
        updateCount = pstmt.executeUpdate();
        // 정상 처리된다면, 0 이외의 숫자값 리턴.
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

```

    } finally {
        close(pstmt);
    }

    return updateCount;
}

```

// 수정 화면에서 날짜에 시분초 표시되는 부분 수정.

selectMember메소드의 쿼리문 수정.

```

String sql = "select custno,custname,phone,address,"
            + "grade,city,"
            + "to_char(joindate,'YYYY-MM-DD') joindate "
            + "from member_tbl_02 where custno=?";

```

// 다음으로 처리 완료후, 수정 완료 메시지 띄우기.

// default.js 에 추가.

```

alert("회원 정보 수정이 완료 되었습니다.");

```

// 안내창

```

setTimeout(function(){
},3000);
// 주어진 시간 (3초) 이후에 동작.

```

// 오전까지한 전체 코드는 카톡으로 공유할게요.

// 오후 시간을 활용하여 회원별 판매금액의 합계를 구하는 기능을 구현해 보세요^^

// 회원별 판매 금액 합계 구하기부터,

// header.jsp 메뉴에 링크 추가.

```

<li><a href="moneyViewAction.me">회원매출조회</a></li>

```

// MemberController.java 에 else if 추가.

```

else if (command.equals("/moneyViewAction.me")) {
    action = new MoneyViewAction();
}

```



```

        try {
            forward = action.execute(request, response);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

// MoneyViewAction.java

```
package action;
```

```
import java.util.ArrayList;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
import svc.MoneyViewService;
```

```
import vo.ActionForward;
```

```
import vo.MoneyBean;
```

```
public class MoneyViewAction implements Action {
```

```
    @Override
```

```
    public ActionForward execute(
```

```
        HttpServletRequest req,
```

```
        HttpServletResponse resp)
```

```
        throws Exception {
```

```
        ActionForward forward = null;
```

```
        forward = new ActionForward();
```

```

        MoneyViewService mvs

        = new MoneyViewService();

        ArrayList<MoneyBean> moneyList

        = mvs.getMoneyView();

        // 호출한 메소드에서 다시 다오를 호출해서 쿼리를 수행하고,

        // 결과를 어레이리스트로 받아 옴.

        req.setAttribute("moneyList", moneyList);

        forward.setPath("./money.jsp");


        return forward;

    }

}

```

// MoneyViewService.java

```
package svc;
```

```
import static db.JdbcUtil.close;
```

```
import static db.JdbcUtil.getConnection;
```

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

```
import java.util.ArrayList;
```

```
import dao.MemberDAO;
```

```
import vo.MoneyBean;
```

```

public class MoneyViewService {

    public ArrayList<MoneyBean> getMoneyView() {

        Connection con = null;

        try {

            con = getConnection();

//            오라클과 연결 만들기.

        } catch (Exception e) {

            e.printStackTrace();

        }

        MemberDAO memberDAO = MemberDAO.getInstance();

        // 싱글톤, 캘린더가 대표 예제.

        // 객체를 계속 재생성 하는 것이 아니라,

        // 객체를 1개만 생성하고, 참조해서 사용하는 것.

        // 쿼리 작업을 위한 준비.

        memberDAO.setConnection(con);

        // 오라클과의 커넥션을 다오에 전달.

        ArrayList<MoneyBean> moneyList

        = memberDAO.selectMoneyList();

        // selectMemberList() 수행한 결과를 어레이리스트로 받기.

        close(con);

        // 커넥션 객체 닫기.

        return moneyList;

    }
}

```

```
}
```

// MemberDAO.java 에 selectMoneyList 메소드 구현.

```
public ArrayList<MoneyBean> selectMoneyList(){
    String sql
    ="select mn.custno, mb.custname, "
        + "decode(mb.grade,'A','VIP','B',"
        + "'일반','직원') grade,"
        + " sum(mn.price) total ";
    sql += "from money_tbl_02 mn,member_tbl_02 mb ";
    sql += "where mn.custno=mb.custno ";
    sql += "group by mn.custno, mb.custname,"
        + " mb.grade ";
    sql += "order by total desc";

    ArrayList<MoneyBean> moneyList
    = new ArrayList<>();
    MoneyBean mb=null;

    try {
        pstmt=con.prepareStatement(sql);
        rs=pstmt.executeQuery();

        if(rs.next()) {
            do {
                mb=new MoneyBean();
                mb.setCustno(rs.getInt("custno"));
                mb.setCustname(rs.getString("custname"));
                mb.setGrade(rs.getString("grade"));
                mb.setMoney(rs.getInt("total"));

                moneyList.add(mb);
            }while(rs.next());
        }
    }catch(Exception e) {
        e.printStackTrace();
    }finally {
        close(rs);
        close(pstmt);
    }

    return moneyList;
}
```

// 매출 목록을 저장할 MoneyBean.java 구현

```
package vo;

public class MoneyBean {
    private int custno;
    private String custname;
    private String grade;
```

```

        private int money;

        public int getCustno() {
            return custno;
        }
        public void setCustno(int custno) {
            this.custno = custno;
        }
        public String getCustname() {
            return custname;
        }
        public void setCustname(String custname) {
            this.custname = custname;
        }
        public String getGrade() {
            return grade;
        }
        public void setGrade(String grade) {
            this.grade = grade;
        }
        public int getMoney() {
            return money;
        }
        public void setMoney(int money) {
            this.money = money;
        }
    }
}

```

// 다음으로 회원 등록창 보이기.

// header.jsp 에 링크 추가.

```
<a class="active" href="memberAddAction.me">회원등록</a></li>
```

// MemberController.java 에 else if 추가.

```

else if (command.equals("/memberAddAction.me")) {
    action = new MemberAddAction();
    try {
        forward = action.execute(request, response);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

// MemberAddAction.java 구현

```
package action;
```

```

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;


import svc.FindSeqService;

import vo.ActionForward;


public class MemberAddAction implements Action {

    @Override

    public ActionForward execute(HttpServletRequest req,

                                HttpServletResponse resp) throws Exception {

        ActionForward forward=null;


        forward=new ActionForward();

        FindSeqService fss = new FindSeqService();

        int findSeq = fss.getSeq();

        req.setAttribute("custno", findSeq);

        forward.setPath("./member_add.jsp");

        return forward;

    }

}

```

// FindSeqService.java 구현

```

package svc;

```

```

import java.sql.Connection;

import dao.MemberDAO;

import static db.JdbcUtil.*;

public class FindSeqService {

    public int getSeq() {

        Connection con=null;

        try {

            con=getConnection();

        } catch (Exception e) {

            e.printStackTrace();

        }

        MemberDAO mDAO = MemberDAO.getInstance();

        mDAO.setConnection(con);

        int seq = mDAO.getSeq();

        close(con);

        return seq;

    }

}

```

// MemberDAO 에 getSeq 메소드 추가.

```

public int getSeq() {

```

```

String sql="select max(custno) custno from "
        + "member_tbl_02";
int custno=100001;

try {
    rs=con.prepareStatement(sql).executeQuery();
    if(rs.next())
        custno=rs.getInt(1)+1;
} catch (SQLException e) {
    e.printStackTrace();
} finally {
    close(rs);
}

return custno;
}

```

// member_add.jsp

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ include file="header.jsp"%>
<h2>홈쇼핑 회원 등록</h2>
<script src="./js/default.js"></script>
<form name="regMem" method="post" onsubmit="return checkForm();"
    action="./memberJoinAction.me">
    <input type="hidden" name="action" value="insert">
    <table border="1">
        <tr>
            <th>회원번호(자동발생)</th>
            <td width="500px"><input type="text" size="20" name="custno"
                value=${custno} readonly></td>
        </tr>
        <tr>
            <th>회원성명</th>
            <td><input type="text" name="custname" size="20"
                ></td>
        </tr>
        <tr>
            <th>회원전화</th>
            <td><input type="text" name="phone" size="30"
                ></td>
        </tr>
        <tr>
            <th>회원주소</th>
            <td><input type="text" name="address" size="40"
                ></td>
        </tr>
        <tr>
            <th>가입일자</th>
            <td><input type="date" name="joindate" size="20"
                ></td>
        </tr>
    </table>

```



```

        <th style="padding: 0 20px;">고객등급 [A:VIP, B:일반,
c:직원]</th>
        <td><input type="text" name="grade" size="20"
        ></td>
    </tr>
    <tr>
        <th>도시코드</th>
        <td><input type="text" name="city" size="20"
        ></td>
    </tr>
    <tr>
        <td colspan="2" align="center"><button
type="submit">등록</button>
        <button type="button">조회</button></td>
    </tr>
</table>
</form>
<%@ include file="footer.jsp"%>

```

// default.js 수정

```

function checkForm() {
    if (document.regMem.custname.value == '') {
        // 이름을 입력하지 않았다면,
        window.alert("회원성명이 입력되지 않았습니다.");
        // 알림창을 띄우고,
        document.regMem.custname.value = "";
        // 이름창은 공백으로 초기화.
        document.regMem.custname.focus();
        // 이름창에 커서를 이동.
        return false;
        // form action 취소.
    }

    if (document.regMem.phone.value == '') {
        window.alert("회원전화가 입력되지 않았습니다.");
        document.regMem.phone.value = "";
        document.regMem.phone.focus();
        return false;
    }

    if (document.regMem.address.value == '') {
        window.alert("회원주소가 입력되지 않았습니다.");
        document.regMem.address.value = "";
        document.regMem.address.focus();
        return false;
    }

    if (document.regMem.joindate.value == '') {
        window.alert("가입일자가 입력되지 않았습니다.");
    }
}

```

```

        document.regMem.joindate.value = "";
        document.regMem.joindate.focus();
        return false;
    }

    if (document.regMem.grade.value == '') {
        window.alert("고객등급이 입력되지 않았습니다.");
        document.regMem.grade.value = "";
        document.regMem.grade.focus();
        return false;
    }

    if (document.regMem.city.value == '') {
        window.alert("도시코드가 입력되지 않았습니다.");
        document.regMem.city.value = "";
        document.regMem.city.focus();
        return false;
    }

    if (document.regMem.action.value == 'insert') {
        alert("회원 등록이 완료 되었습니다.");
    } else {
        alert("회원 정보 수정이 완료 되었습니다.");
        // 안내창
    }

    setTimeout(function() {
    }, 3000);
    // 주어진 시간 (3초) 이후에 동작.
}

```

// MemberController.java 에 else if 추가

```

else if (command.equals("/memberJoinAction.me")) {
    action = new MemberJoinAction();
    try {
        forward = action.execute(request, response);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

// MemberJoinAction.java 구현

```

package action;

```

```
import java.io.PrintWriter;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
import svc.MemberJoinService;
```

```
import vo.ActionForward;
```

```
import vo.MemberBean;
```

```
public class MemberJoinAction implements Action {
```

```
    @Override
```

```
    public ActionForward execute(HttpServletRequest req,
```

```
        HttpServletResponse resp) throws Exception {
```

```
        MemberBean member=new MemberBean();
```

```
        boolean joinResult = false;
```

```
        member.setCustname(req.getParameter("custname"));
```

```
        member.setPhone(req.getParameter("phone"));
```

```
        member.setAddress(req.getParameter("address"));
```

```
        member.setJoindate(req.getParameter("joindate"));
```

```
        member.setGrade(req.getParameter("grade"));
```

```
        member.setCity(req.getParameter("city"));
```

```
        // 멤버빈을 가져오는 것인가?
```

```
        // 폼에서 생성된 정보는 리퀘스트 객체에 담겨서 목적 페이지로 전달.
```

```
// 새로운 멤버빈 객체를 생성하고, 각각의 항목에 값을 전달.  
  
// 각각의 입력 요소를 가지고 멤버빈을 만드는 부분.  
  
// 생성된 멤버빈을 각 클래스에 전달하고 오라클에 저장하면 됨.  
  
// 서비스와 다오를 거쳐서 처리.
```

```
MemberJoinService mjs
```

```
= new MemberJoinService();
```

```
joinResult=mjs.joinMember(member);
```

```
ActionForward fo = null;
```

```
if(joinResult==false) {
```

```
    resp.setContentType("text/html;charset=utf-8");
```

```
    PrintWriter out=resp.getWriter();
```

```
    out.println("<script>");
```

```
    out.println("alert('회원 등록에 실패 했습니다.');
```

```
    out.println("history.back()");
```

```
    out.println("</script>");
```

```
}else {
```

```
    fo=new ActionForward();
```

```
    fo.setRedirect(true);
```

```
    fo.setPath("./memberAddAction.me");
```

```
}
```

```
return fo;
```

```
}
```

```
}
```

```
// MemberJoinService.java 구현
```

```
package svc;
```

```
import static db.JdbcUtil.close;
```

```
import static db.JdbcUtil.commit;
```

```
import static db.JdbcUtil.getConnection;
```

```
import static db.JdbcUtil.rollback;
```

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

```
import dao.MemberDAO;
```

```
import vo.MemberBean;
```

```
public class MemberJoinService {
```

```
    public boolean joinMember(MemberBean member) {
```

```
        boolean joinSuccess = false;
```

```
        MemberDAO dao = MemberDAO.getInstance();
```

```
        Connection con = null;
```

```
        try {
```

```
            con = getConnection();
```

```
        } catch (Exception e) {
```

```

        e.printStackTrace();

    }

    dao.setConnection(con);

    int insertCount = dao.insertMember(member);

    if (insertCount > 0) {

        joinSuccess = true;

        commit(con);

    } else {

        rollback(con);

    }

    close(con);

    return joinSuccess;

}

}

```

// MemberDAO.java 에 insertMember 메소드 추가.

```

public int insertMember(MemberBean member) {
    String sql = "insert into member_tbl_02 values ("
        + "custno_seq.nextval,?,?,?,?,?,?)" ;
    int insertCount=0;

    try {
        pstmt=con.prepareStatement(sql);
        pstmt.setString(1, member.getCustname());
        pstmt.setString(2, member.getPhone());
        pstmt.setString(3, member.getAddress());
        pstmt.setString(4, member.getJoindate());
        pstmt.setString(5, member.getGrade());
        pstmt.setString(6, member.getCity());
        insertCount = pstmt.executeUpdate();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {

```

```
        close(pstmt);  
    }  
    return insertCount;  
}
```

// 오늘은 회원별 매출 합계,

// 회원번호 +1 조회.

// 새로운 회원 등록을 살펴 봤어요.

// 지금부터, 외부평가 문제를 반복 숙달하여, 3시간 이내에 외워서 해결할 수 있도록 연습
바래요. 이번주 금요일 까지 3일간은 따로 진도를 나가지 않고, 여러분이 연습하는 시간이에요.
총 4시간의 시간중 1시간을 여유로 두는 것은 돌발 상황에 대처하기 위함 입니다.

// cq-net에 공유된 문제가 숙달된 분은 지난번 공유했던 올해 2월 버전도 해결해 보세요.