07. 오라클 DB 연동

1) JDBC 개요 및 특성

(1) 자바 App 에서 표준화 된 데이터 베이스에 연동하는 방법을 제공해주는 기술
(2) 각 데이터베이스 접속에 대한 상세한 정보를 알 피료는 없음 → 인터페이스로 구성
➔ import.java.sql 클래스로 제공 ➔ DB 연동하는 method 는 다르다
** 인터페이스로 구성(공통된 method 만 사용)
(3) JDBC 드라이버를 설정

2) JDBC 프로그래밍의 순서

(1) Class.forName("oracle.jdbc.driver.OracleDriver"); → 메모리에 올리는 작업
: DB 에 관련된 클래스로를 메모리에 올려 객체를 생성함.
Com.mysql.jdbc.driver(mysql) ← MySQL

(2) Connection conn = DriverManager.getConnection(JDBC_url,"아이디","비밀번호");
→ JDBC_url 구성 : jdbc:oracle.thin:@IP 주소:포트:SID

(3) SQL 구문을 작성하기 위해서 필요로 하는 객체 : 2 가지

★ **Statement** 생성 및 쿼리 실행하는 **문장객체 : DML 에 많이쓰임**
➔이 전의 SQL 구문을 지우고 새로 작성 한 후 실행
★ **PreparedStatement** 생성 및 쿼리 실행 ➔ 실행 처리 속도가 빠른 객체
➔**필요로 하는 부분만 변경해서 작성이 가능** ➔ **입력받는 부분은 ? 로 표시**

(4) SQL 문장을 실행하기 위해 사용되는 method
★ **executeQuery()** method : **SELECT 문** 일 때 사용
➔ return 값은 **ResultSet 클래스의 객체로 반환**
➔ **테이블의 구조에 영향을 미치지 않는 SQL 구문에 사용**
★ **excuteUpdate()** method : **INSERT, UPDATE, DELETE 문** 일 때 사용
➔ return 값은 처리된 데이터의 수를 정수형으로 반환
➔ **테이블의 구조에 영향을 미치는 SQL 구문에 사용**
➔1: 실행성공 / 0 : 실행 실패
(5) 문자, 숫자 데이터를 가져오기 ➔ varchar2 , varchar

ResultSet 객체 rs 선언 후

rs.getString("불러올 컬럼명") ex) rs.getString("name")

rs.getInt("불러올 칼럼명") ex) rs.getInt("age")
rs.getDouble ~ rs.getFloat~

rs.get 자료형("필드명") or **rs.get 자료형("인덱스번호")**
SQL 구문에 따라서 변동이 있을 수 있어서 추천하지 않음

(6) DB 연결 해제 구문 => 생성된 객체의 역순으로 메모리를 해제시킴
rs.close();
pstmt.close(); // stmt.close();
conn.close();

dbselect.jsp : Database 에 접속하여, SQL 구문을 실행해보는 예제

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page import = "java.sql.*" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
        Connection con = null;
        // 형식) jdbc:oracle:thin:@localhost:1521:orcl";
        String url="jdbc:oracle:thin:@localhost:1521:orcl";

        Statement stmt = null;
        PreparedStatement pstmt = null;
        Statement stmt2=null;
        ResultSet rs = null;
        String sql = null;
        try{
                // 1. 접속드라이버 -> 메모리 로드
                Class.forName("oracle.jdbc.driver.OracleDriver");
                // 2. 접속 인증 (계정)
                con = DriverManager.getConnection(url,"scott","tiger");
                System.out.println("접속 con:"+con);
                // 3. 테이블 생성 : create table MyTest2(name varchar2(20), age number);
                stmt = con.createStatement();
                sql = "create table MyTest2(name varchar2(20), age number)";
                int create = stmt.executeUpdate(sql);
                System.out.println("테이블 성공 유무 (create):"+create); // 0

                // 4. 삽입
                pstmt = con.prepareStatement("insert into MyTest2 values(?,?)");
                pstmt.setString(1,"Park");
                pstmt.setInt(2,33);
                int insert = pstmt.executeUpdate();
                System.out.println("데이터 입력 성공 유무 (insert):"+insert); // 1
```

문장객체들과
selet의 결과를 받는 ResultSet 객체 생성

```
                //5. select
                stmt2 = con.createStatement();
                String sql2 = "select * from MyTest2";

                rs = stmt2.executeQuery(sql2); // select로 찾은 자료를 담는 객체 rs
%>
        <table border="1" cellspacing="0" cellpadding="0">
                <tr bgcolor ="pink">
                        <th> name </th>
                        <th> age </th>
                </tr>
<%
                        while(rs.next()){
%>
                <tr>
                        <td><%=rs.getString("name") %></td>
                        <td><%=rs.getInt("age") %></td>
                </tr>
<%
                }
                }catch(Exception e){
                        System.out.println("DB연결 오류:"+e);
                }
                finally{
                        //db연결과 메모리를 해제
                        rs.close();
                        stmt2.close();
                        pstmt.close();
                        stmt.close();
                        con.close();
                }
%>
</table>
</body>
</html>
```

3) 모델 1 방식의 프로그램 작성법

(1) 장점

① 적은 규모의 사이트에 적합

② 적은 인원으로 웹사이트 개발에 적합 : 개인프로젝트

(2) 단점

① 대규모 사이트에는 적합하지 않다.

② 페이지 수가 많아지면 모델 1 방식으로는 유지보수가 어렵다.

4) 모델 1 방식

(1) jsp 에서 자바코드를 사용하지 않는다. (가능한한 적게)

(2) DB 연동과 관련된 테이블에 다른 **자바빈즈를 따로 작성한다.**

(3) 자바빈즈의 종류

① DB 연결빈 : DB 연결 및 연결 해제 기능 ➔ (DBConnectoinMgr.java)

② 데이터 저장빈 : 테이블에 데이터를 저장 및 조회(Setter,Getter) 하는 기능

➔ 테이블 설계가 10 개가 나온다면, 클래스가 10 개 ➔ DTO (Data Transfer Object)

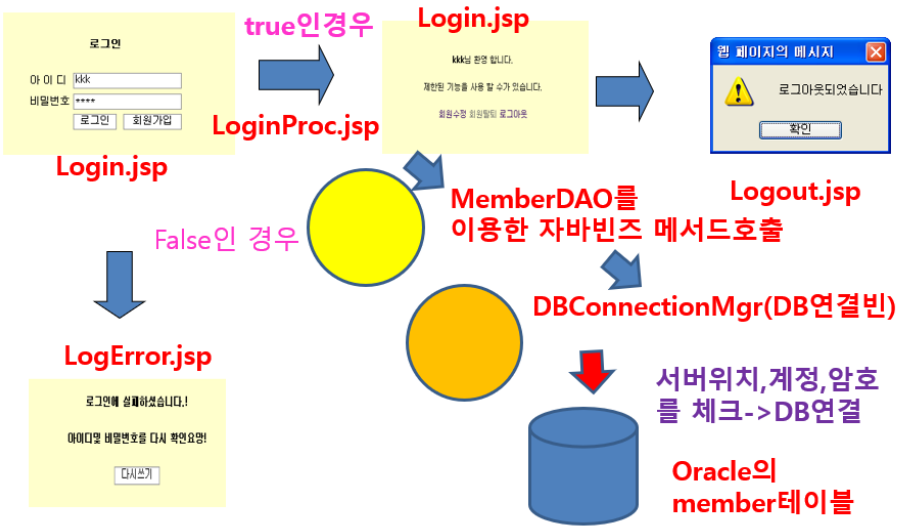
➔ 테이블의 '필드'들을 선언하고 set,get 을 하는 객체로 보면 된다.

➔MemberDTO(혹은 VO)

③ 비즈니스 로직 빈 : 웹상에서의 method 호출 기능

➔ DB 와 접속해서 움직이는 실질적인 method ➔ DAO(Data Access Object)

➔ MemberDAO



DBConnectionMgr.java : DB 연결빈 , 중요한 부분만 따온 내용

```
// [DB연결]
/*    MySQL 드라이버 연결
        private String _driver = "com.mysql.jdbc.Driver",
    _url = "jdbc:mysql://localhost:3306/mydb?useUnicode=true&characterEncoding=UTF-8",
    _user = "root",
    _password = "1234"; */
    // Oracle 드라이버 연결
    private String _driver = "oracle.jdbc.driver.OracleDriver",
    _url = "jdbc:oracle:thin:@localhost:1521:orcl",   ← 자신의 ip 혹은 localhost를 꼭! 넣어줄것
    _user = "scott",
    _password = "tiger";
    //DB연결 싱글톤 패턴
        public static DBConnectionMgr getInstance() {
            if (instance == null) {    //만약에 객체가 생성이 된 상태가 아니라면
            synchronized (DBConnectionMgr.class) {      //동기화 처리
                if (instance == null) {    //생성해서 반환해줘라
                        instance = new DBConnectionMgr();
                }
            }
        }
        return instance;//반환값
    }
```

```
// [커넥션풀]
public synchronized Connection getConnection() throws Exception {
                ➔메모리에올려주고 Connection객체를 얻어오는 메서드
        if (!initialized) { //초기화 되지 않은 상태 : initialized 값
        //접속할 DB의 드라이버를 메모리에 로드
            Class c = Class.forName(_driver);
        //자동 등록(드라이버클래스)
            DriverManager.registerDriver((Driver) c.newInstance());
            initialized = true;//접속상태
        }
//DB연결을 해제해주는 메서드
    public void freeConnection(Connection c, PreparedStatement p, ResultSet r) {
        try {
            if (r != null) r.close();
            if (p != null) p.close();
            freeConnection(c);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

08. 회원 로그인

1) DB 연결 : DBConnectionMgr.java 를 이용 → 클래스로 따로 작성

```
1.드라이버명 2.url(접속경로) 3.접속계정명 4.접속할 암호
private String _driver = "oracle.jdbc.driver.OracleDriver",
_url = "jdbc:oracle:thin:@localhost:1521:orcl",
_user = "scott",
_password = "tiger";
```

2) 데이터 저장빈 : 테이블에 데이터를 저장 및 조회

→ 테이블 DTO (Data Transfer Obect) : MemberDTO

```
public class MemberDTO {
        // 입력받은 갯수만큼 멤버변수를 저장(input type="text" name와 같아야된다.)
        private String mem_id;
        private String mem_passwd;
        private String mem_name; // 이름
        private String mem_email;// 메일
        private String mem_phone;// 전번
        private String mem_zipcode;// 우편번호
        private String mem_address;// 주소
        private String mem_job;// 직업
'''
```

3) 비즈니스 로직빈 : 웹상에서 method 선언 호출

→ DAO(Data Access Object) 테이블명 DAO : MemberDAO

loginProc.jsp

```
<jsp:useBean id="memMgr" class="member.MemberDAO"/>
<%
        String mem_id = request.getParameter("mem_id");
        String mem_passwd = request.getParameter("mem_passwd");
        System.out.println("id:"+mem_id+", passwd:"+mem_passwd);

        //MemberDAO memMgr = new MemberDAO();
        boolean check = memMgr.loginCheck(mem_id, mem_passwd);

        //check -> LoginSuccess.jsp(인증화면), LogError.jsp(에러메세지)
        if(check){ //logind이 성공이면
                session.setAttribute("idKey", mem_id);
                response.sendRedirect("LogSuccess.jsp");
        }else{//
                response.sendRedirect("LogError.jsp");
        }
%>
```

Bean을 통해 객체 생성

login.jsp 로부터 Id값과 passwd값을 getParameter로 가져옴

두 값을 MemberDAO의 method loginCheck()로 확인

Session 메모리에 id 값을 메모리에 저장 (페이지를 옮겨도 서버측 메모리에 남는다)

```
function loginCheck(){
        if(document.login.mem_id.value==""){
                alert("아이디를를 입력해 주세요.");
                document.login.mem_id.focus();
                return; //return false;
        }
        if(document.login.mem_passwd.value==""){
                alert("비밀번호를 입력해 주세요.");
                document.login.mem_passwd.focus();
                return;
        }
        document.login.submit();
}
```

별도의 Javascript 에 있는 loginCheck() function **Submit() 은 여기서 이루어짐**

LoginSuccess.jsp // 로그인 성공시 page

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
        //웹상에서 강제로 접근할 가능성 > 인증 처리가 필요 >
        // session.setAttribute("idKey",mem_id);
        String mem_id = (String)session.getAttribute("idKey");
%>
<body>
<%
        if(mem_id!=null){ // mem_id 값이 null 이 아니면 ➔ 로그인이 되어있다면
%>
        <b><%=mem_id %></b>님 환영합니다.   </p>
        당신은 제한된 기능을 사용할 수 있습니다 </p>
        <a href="Logout.jsp">로그아웃</a>
<% } %>
<script>
        location.href="Login.jsp"; // 자기 자신page로 돌아감 → self
</script>
</body>
```

LoginError.jsp // 로그인 실패시 page

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
</head>
<body>
        로그인에 실패하셨습니다.<br>
        <input type="button" value="다시 쓰기" onclick="history.back()">
</body>
</html>
```

LogOut.jsp // 로그아웃

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
 <%
        //세션 연결 상태를 종료시킨다. --> 메모리 상에서의 계정 정보(idKey)를 삭제한다
        session.invalidate();
 %>
<script>
        alert("정상적으로 로그아웃 되었습니다.");
        location.href="Login.jsp";
</script>
```