

서버프로그램 구현

1. DAO (Data Access Object) / VO (Value Object)

✓ DAO

- DB 질의를 통해 데이터에 접근하는 객체
- DB의 한 테이블당 DAO 클래스를 하나씩 만들어 두면 유지보수가 쉬워집니다.
 - 예를 들어, User, Post 라는 두 테이블이 있을 때, User 테이블의 데이터를 접근하고 싶다면 UserDao 클래스를 만들고, 마찬가지로 Post 테이블에 접근하고 싶다면 PostDAO 클래스를 만들어서 관리를 할 수 있습니다.

✓ VO

- DB의 한 테이블에 존재하는 컬럼들을 멤버 변수로 작성하여, 테이블의 컬럼 값을 java에서 객체로 다루기 위해 사용합니다.
- 즉, 데이터들을 캡슐화 해서 객체로 만든 것입니다.
 - 예를 들어 User 테이블에 id, name, phone 컬럼이 있다면, UserVO 클래스에는 id, name, phone 멤버 변수가 존재하고 이에 대한 접근은 getter, setter로 다룹니다.

2. DAO와 VO의 흐름

- ✓ DAO와 VO를 어떻게 다루는지 살펴보기 위해, 회원가입 상황을 가정
- ✓ 회원가입 페이지에는 아이디, 비밀번호와 같은 데이터들을 입력할 수 있는 form이 있을 것이고,
- ✓ Servlet에서는 회원 데이터가 넘어오면 이 값들을 DB의 User 테이블에 저장해야 합니다.
- ✓ User 테이블에는 id와 password라는 칼럼이 있다고 가정하겠습니다.
- ✓ UserVO와 UserDAO는 다음과 같은 작업을 수행

2. DAO와 VO의 흐름

✓ UserVO

- id, password 멤버 변수를 갖고 있음
- 사용자의 입력 값을 각 변수에 할당 시키는 작업을 수행

✓ UserDAO

- ✓ DB에 JDBC로 연결하여, Connection을 생성
- ✓ 회원 정보를 VO 객체로 받아서 DB에 INSERT 하기 위한 쿼리를 작성하여 실행

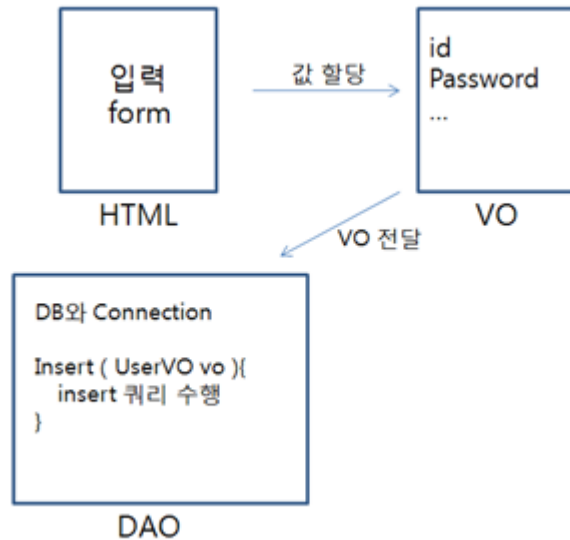
2. DAO와 VO의 흐름

✓ UserVO

- id, password 멤버 변수를 갖고 있음
- 사용자의 입력 값을 각 변수에 할당 시키는 작업을 수행

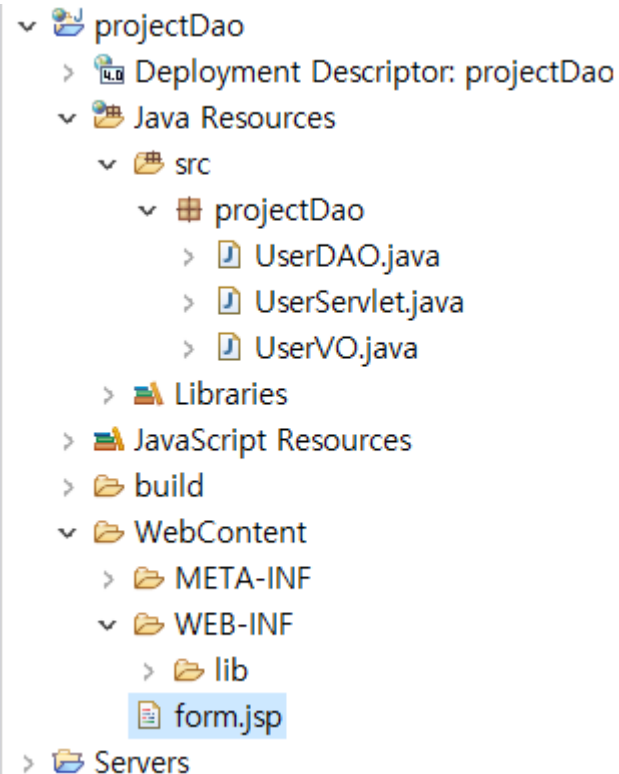
✓ UserDao

- ✓ DB에 JDBC로 연결하여, Connection을 생성
- ✓ 회원 정보를 VO 객체로 받아서 DB에 INSERT 하기 위한 쿼리를 작성하여 실행



3. DAO , VO 구현

- ✓ 이제 위의 상황을 구현해
- ✓ MVC 패턴에 맞게 Servlet과 JSP을 사용
- ✓ 여기서 JSP는 단순 HTML 문서와 같고, Servlet은 Controller답게 요청을 처리하고 데이터를 추가하는 코드가 작성
- ✓ 디렉토리 구조는 다음과 같습니다.



3. DAO , VO 구현

1) userTbl Table생성

```
SQL> create table userTbl(no number not null primary key,  
2 name varchar2(20) not null,  
3 email varchar2(40),  
4 pwd varchar2(20));
```

테이블이 생성되었습니다.

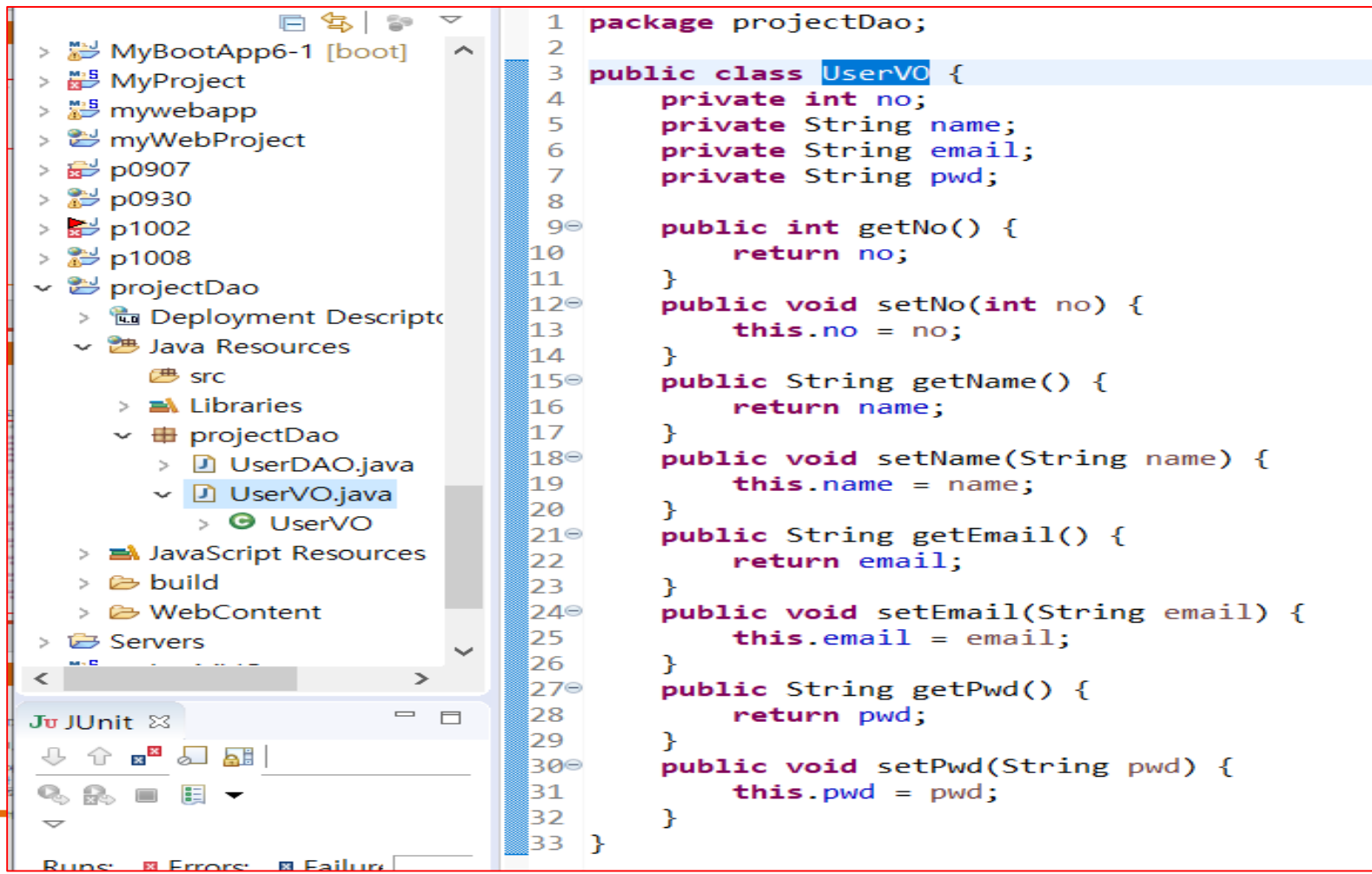
```
SQL> _
```

서버프로그램 구현

3. DAO , VO 구현

2) UserVO 클래스 생성(UserVO.java)

- ✓ 먼저 VO 클래스를 만듭니다.
- ✓ VO라고 해서 특별한 파일이 아니고, 그냥 클래스 파일을 만들면 됨



The screenshot shows an IDE with a project structure on the left and a code editor on the right. The project structure on the left shows a hierarchy starting with 'MyBootApp6-1 [boot]', followed by 'MyProject', 'mywebapp', 'myWebProject', 'p0907', 'p0930', 'p1002', 'p1008', and 'projectDao'. Under 'projectDao', there are 'Deployment Descriptors', 'Java Resources' (containing 'src', 'Libraries', and 'projectDao'), 'JavaScript Resources', 'build', and 'WebContent'. The 'projectDao' folder is expanded, showing 'UserDAO.java' and 'UserVO.java'. The 'UserVO.java' file is selected, and its code is displayed in the editor on the right.

```
1 package projectDao;
2
3 public class UserVO {
4     private int no;
5     private String name;
6     private String email;
7     private String pwd;
8
9     public int getNo() {
10         return no;
11     }
12     public void setNo(int no) {
13         this.no = no;
14     }
15     public String getName() {
16         return name;
17     }
18     public void setName(String name) {
19         this.name = name;
20     }
21     public String getEmail() {
22         return email;
23     }
24     public void setEmail(String email) {
25         this.email = email;
26     }
27     public String getPwd() {
28         return pwd;
29     }
30     public void setPwd(String pwd) {
31         this.pwd = pwd;
32     }
33 }
```


3. DAO , VO 구현

2) UserVO 클래스 생성

- ✓ VO 클래스는 캡슐화 하고자 하는 DB 테이블의 컬럼명과 동일하게 멤버 변수를 갖으면 된다.
 - 예제에서는 no, name, email, pwd 컬럼이라고 가정
 - private으로 선언
- ✓ getter와 setter를 통해 멤버 변수에 접근
참고로 Spring이나 JSP를 보완한 JSTL라이브러리에서는 VO 멤버변수를 getter, setter로 접근하도록 자동으로 처리하기 때문에, getter와 setter 메서드를 작성하는 것이 좋습니다.
빠르게 getter, setter를 작성하려면 IDE를 이용합니다.
이클립스 상단 탭 -> Source
Generate Getters and Setters

3. DAO , VO 구현

3) UserDAO 클래스 생성(UserDAO.java)

✓ 다음으로 UserDAO 클래스를 만듦

```
1 package projectDao;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.PreparedStatement;
6 import java.sql.SQLException;
7
8 public class UserDAO {
9     private Connection getConnection() throws SQLException {
10         Connection conn = null;
11
12         try {
13             String url = "jdbc:oracle:thin:@//localhost:1521/orcl";
14             String user = "scott";
15             String pwd = "1234";
16             Class.forName("oracle.jdbc.OracleDriver");
17
18             conn = DriverManager.getConnection(url, user, pwd);
19             System.out.println(" 드라이버 로딩 성공 ");
20         }
21         catch (ClassNotFoundException e) {
22             System.out.println(" 드라이버 로딩 실패 ");
23         }
24
25         return conn;
26     }
27 }
```

3. DAO , VO 구현

3) UserDAO 클래스 생성

✓ 다음으로 UserDAO 클래스를 만듦

```
28 public boolean insert(UserVO vo ) {
29     boolean result = false;
30     Connection conn = null;
31     PreparedStatement pstmt = null;
32
33     try {
34         conn = getConnection();
35
36         // Column
37         // PK , name , email , password
38         String sql = "INSERT INTO userTbl VALUES (?, ?, ?, ?)";
39         pstmt = conn.prepareStatement(sql);
40         pstmt.setInt(1, vo.getNo());
41         pstmt.setString(2, vo.getName());
42         pstmt.setString(3, vo.getEmail());
43         pstmt.setString(4, vo.getPwd());
44         int count = pstmt.executeUpdate();
45         result = (count == 1);
46     }
47     catch (SQLException e) {
48         e.printStackTrace();
49     }
```

3. DAO , VO 구현

3) UserDao 클래스 생성

✓ 다음으로 UserDao 클래스를 만듦

```
51         finally {  
52             try {  
53                 if( conn != null ) {  
54                     conn.close();  
55                 }  
56                 if( pstmt != null ) {  
57                     pstmt.close();  
58                 }  
59             }  
60             catch(SQLException e) {  
61                 e.printStackTrace();  
62             }  
63         }  
64         return result;  
65     }  
66 }  
67 }
```

3. DAO , VO 구현

3) UserDAO 클래스 생성

- ✓ DB에 Connection을 하고, 쿼리를 수행
- ✓ 보시는 바와 같이 DAO의 역할은 DB에 접근해서 쿼리를 수행하는 것이 전부
- ✓ 여기서 insert() 메서드는 인자로 UserVO 객체를 받고 있다는 점에 주목
- ✓ 뒤에서 작성할 Servlet에서 UserDAO.insert() 메서드를 호출하는데, 이 때 클라이언트가 입력한 정보들을 전달해주기 때문

3. DAO , VO 구현

4) Servlet 작성

✓ 이어서 Servlet 클래스를 작성

The screenshot shows the 'Create Servlet' wizard in an IDE. The main window is titled 'Create Servlet' and contains the following fields:

- Name:** UserServlet
- Description:** (empty)
- Initialization parameters:** (empty)
- URL mappings:** /UserServlet
- ☐ Asynchronous Support

A 'URL Mappings' dialog box is open over the 'URL mappings' field. It contains:

- Pattern:** /add
- Buttons:** OK, Cancel

At the bottom of the 'Create Servlet' window, there are navigation buttons: < Back, Next >, Finish (highlighted), and Cancel.

3. DAO , VO 구현

4) Servlet 작성(UserServlet.java)

✓ 이어서 Servlet 클래스를 작성

```
3 import java.io.IOException;
4 import javax.servlet.ServletException;
5 import javax.servlet.annotation.WebServlet;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.http.HttpServletRequest;
8 import javax.servlet.http.HttpServletResponse;
9
10 @WebServlet("/add")
11 public class UserServlet extends HttpServlet {
12     private static final long serialVersionUID = 1L;
13     protected void doGet(HttpServletRequest request, HttpServletResponse response)
14         throws ServletException, IOException {
15         request.setCharacterEncoding("utf-8");
16         UserVO vo = new UserVO();
17         vo.setNo(Integer.parseInt(request.getParameter("no")));
18         vo.setName(request.getParameter("name"));
19         vo.setEmail(request.getParameter("email"));
20         vo.setPwd(request.getParameter("pwd"));
21         UserDAO dao = new UserDAO();
22         dao.insert(vo);
23         response.sendRedirect("/projectDao/form.jsp");
24     }
25     protected void doPost(HttpServletRequest request, HttpServletResponse response)
26         throws ServletException, IOException {
27         doGet(request, response);
28     }
29 }
```

3. DAO , VO 구현

4) Servlet 작성

- ✓ 요청 객체의 `getParameter()` 메서드를 호출하여, 클라이언트로부터 회원정보 데이터(no, name , email , password)의 정보들을 받은 뒤에 그 값들을 UserVO 객체에 할당합니다.
- ✓ UserDao 객체의 `insert()` 메서드를 호출하여 인자로 UserVO 객체를 전달합니다.
- ✓ DAO 객체에서는 쿼리를 날려서 DB에 해당 유저 정보를 insert 합니다.
- ✓ 회원 정보가 등록되면, 클라이언트에게 응답 페이지를 보여줘야 하므로 jsp파일로 redirect 합니다.
- ✓ redirect된 jsp파일은 회원가입 입력 폼으로 다음으로 살펴 봄

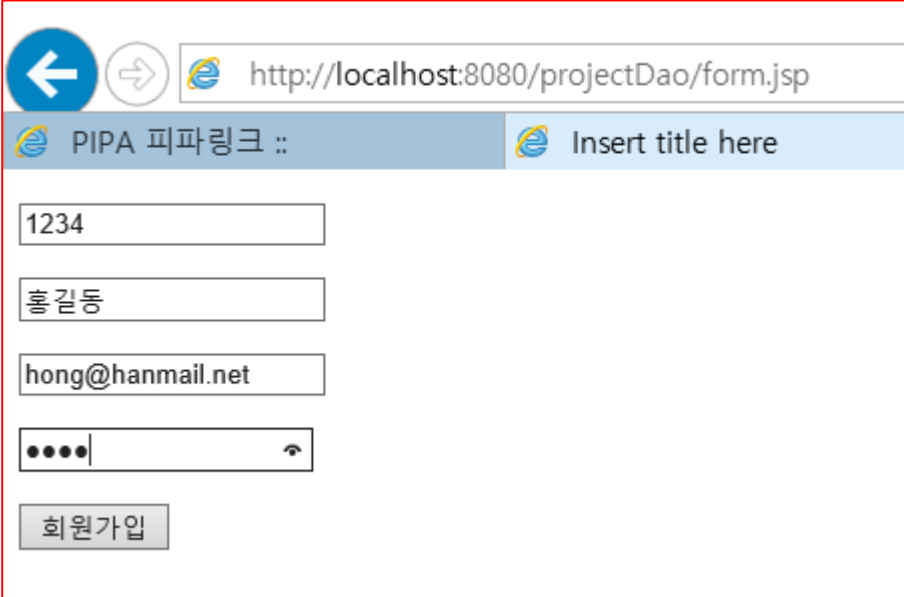
5) JSP 파일 생성

회원가입 입력 form인 JSP파일(form.jsp)

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2 <!DOCTYPE html>
3 <html>
4 <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
6     <title>Insert title here</title>
7 </head>
8 <body>
9     <form action="/projectDao/add" method="get">
10         <p><input type="text" name="no" placeholder="회원번호"></p>
11         <p><input type="text" name="name" placeholder="이름"></p>
12         <p><input type="text" name="email" placeholder="이메일"></p>
13         <p><input type="password" name="pwd" placeholder="비밀번호"></p>
14
15         <p><input type="submit" value="회원가입"></p>
16     </form>
17 </body>
18 </html>
```

6) 테스트

- ✓ 이제 톰캣에 projoectDa 웹 프로젝트를 등록하고, URL에 localhost:8080/projectDao/form.jsp 를 입력해서 회원가입 하면, DB에 데이터가 추가되는 것을 확인



A screenshot of a web browser window. The address bar shows the URL `http://localhost:8080/projectDao/form.jsp`. The browser's title bar displays "PIPA 피파링크 ::" and "Insert title here". The page content includes a registration form with the following fields and controls:

- A text input field containing the number "1234".
- A text input field containing the Korean text "홍길동".
- A text input field containing the email address "hong@hanmail.net".
- A password input field with four dots and a toggle icon.
- A button labeled "회원가입" (Sign Up).

6) 테스트

- ✓ 이제 톰캣에 projoectDa 웹 프로젝트를 등록하고, URL에 localhost:8080/projectDao/form.jsp 를 입력해서 회원가입 하면, DB에 데이터가 추가되는 것을 확인

```
SQL> select * from usertbl;
```

NO	NAME	EMAIL	PWD
1234	홍길동	k001	1234

```
SQL> _
```