

Class230116

Connection Pool - Basic

JDBC 를 사용할때 가장 많이 리소스 즉 자원이 소모되는 부분이 디비 연동에 필요한 Connection 객체를 생성하는 부분이다. 지금까지 방법들은 모두 JSP 에서 SQL 구문을 수행하기 위해서 Conneciton 객체를 생성하고 사용후 제거하는 과정을 반복해왔다. 접속자가 많아질 경우 시스템의 성능을 급격하게 저하시키게 된다.

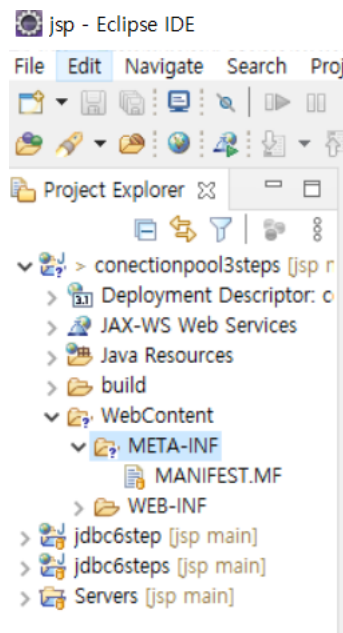
따라서 이러한 문제점을 해결하기 위한 방법으로 커넥션 풀을 이용하게 된다. 사용자가 접속 할때 마다 매번 새로운 connection객체를 생성하는것이 아니라 일정 개수의 Connection 객체를 미리 생성해 놓고 사용자의 요청이 있을 때마다 가용한 객체를 할당하고 다시 회수하는 방식이다.

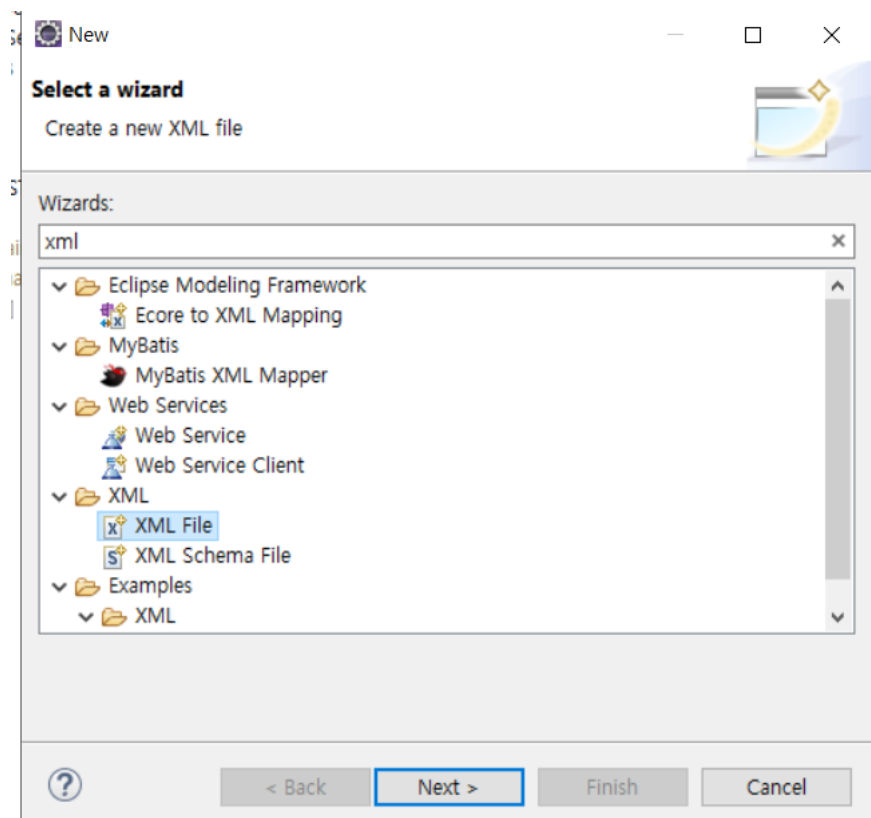
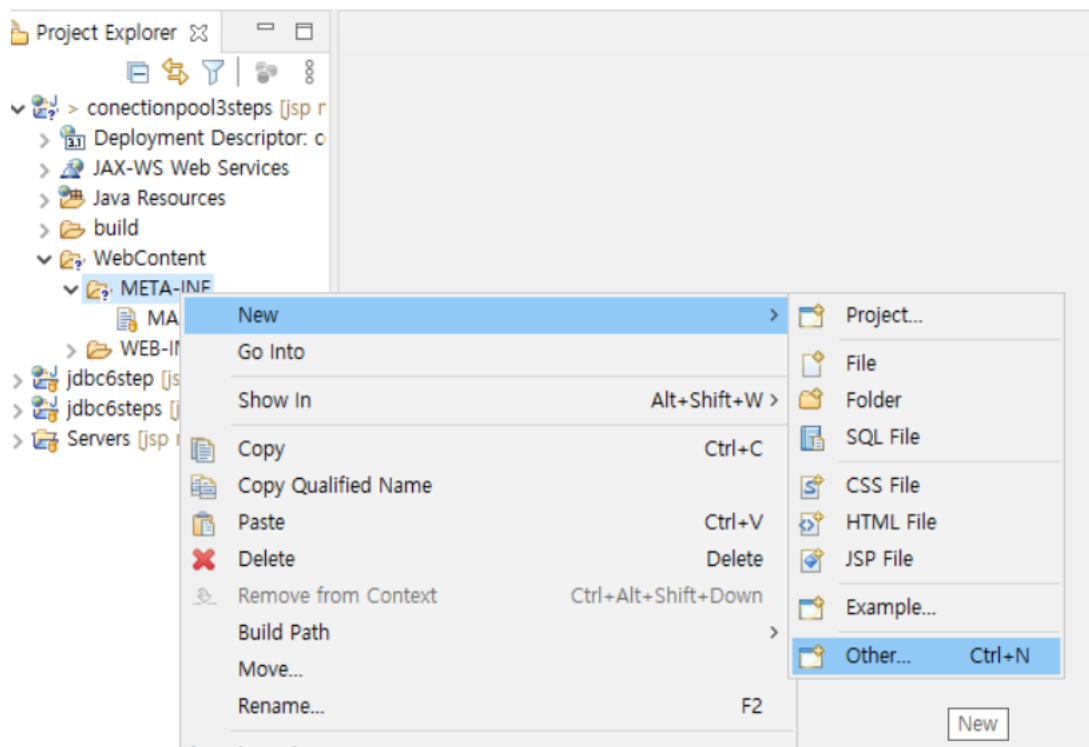
Connection Pool 설정

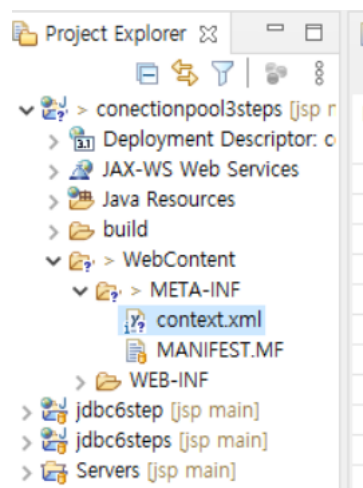
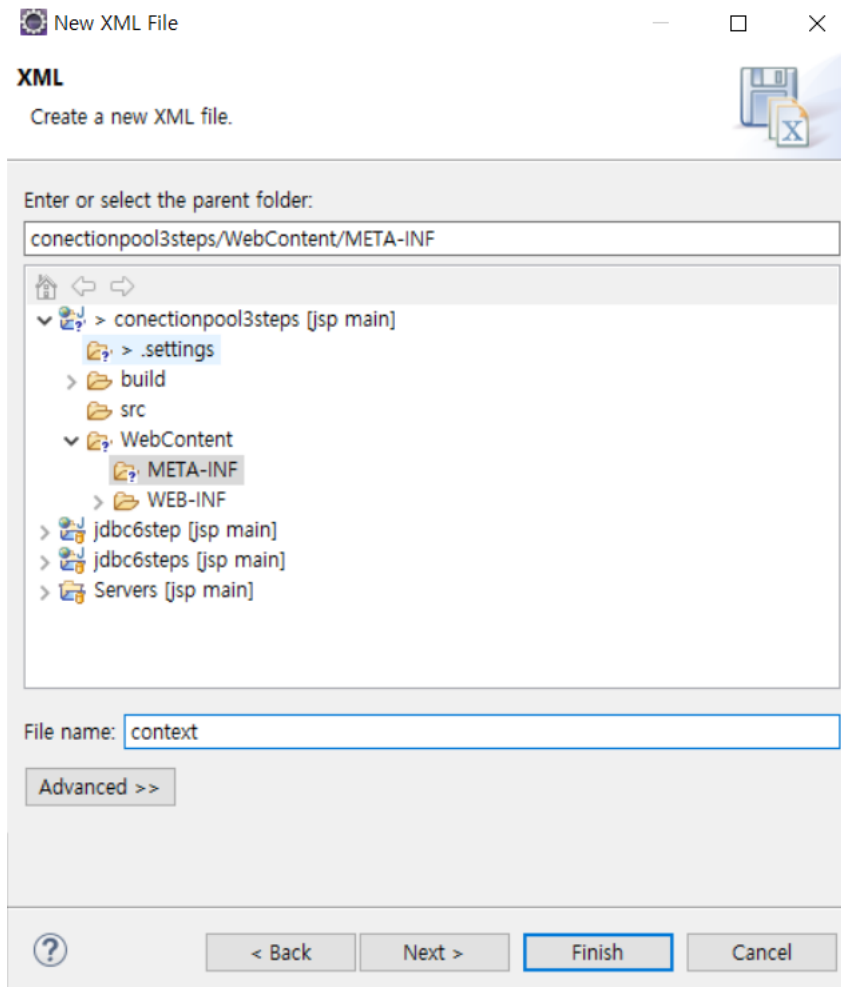
▼ 1. context.xml[Connection Pool 설정 정의]

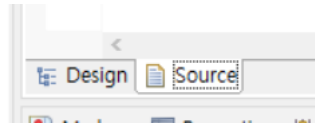
데이터 베이스에 대한 커넥션 풀을 사용하기 위한 **설정을 정의**한다.

위치는 WebContent > META-INF > **context.xml**









source 클릭

```
*context.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <Context>
3     <Resource name="jdbc/univ"
4         auth="Container"
5         type="javax.sql.DataSource"
6         driverClassName="com.mysql.jdbc.Driver"
7         url="jdbc:mysql://localhost:3306/univ?serverTimezone=UTC"
8         username="root"
9         password="0000"
10        maxTotal="16"
11        maxIdle="4"
12        maxWaitMillis="-1" />
13 </Context>
```

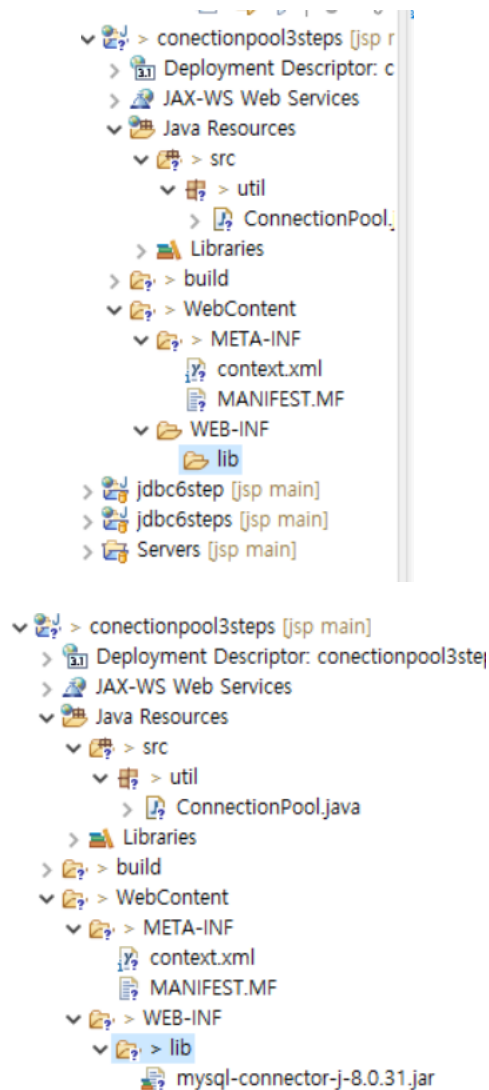
```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
  <Resource name="jdbc/univ"    <- univ 사용할 디비명
    auth="Container"
    type="javax.sql.DataSource"
    driverClassName="com.mysql.jdbc.Driver"    <- 연결할 DB mysql, maria, oracle .....
    url="jdbc:mysql://localhost:3306/univ?serverTimezone=UTC" <- univ 사용할 디비명
    username="root"    <- 디비 아이디 ( 호스팅 업체에 업로드시에는 변경)
    password="0000"    <- 디비 패스워드 ( 호스팅 업체에 업로드시에는 변경)
    maxTotal="16"    <- 미리 생성할 커넥션의 갯수
    maxIdle="4"    <- 최저 유지 커넥션 갯수
    maxWaitMillis="-1" />    <- 항상 -1, 기다리는 시간 기다리지않고 바로바로 처리
</Context>

// ?serverTimezone=UTC    특정 서버에서 타임존 설정을 하지 않으면 동작하지 않을때가 있다.
```



JDBC connector driver

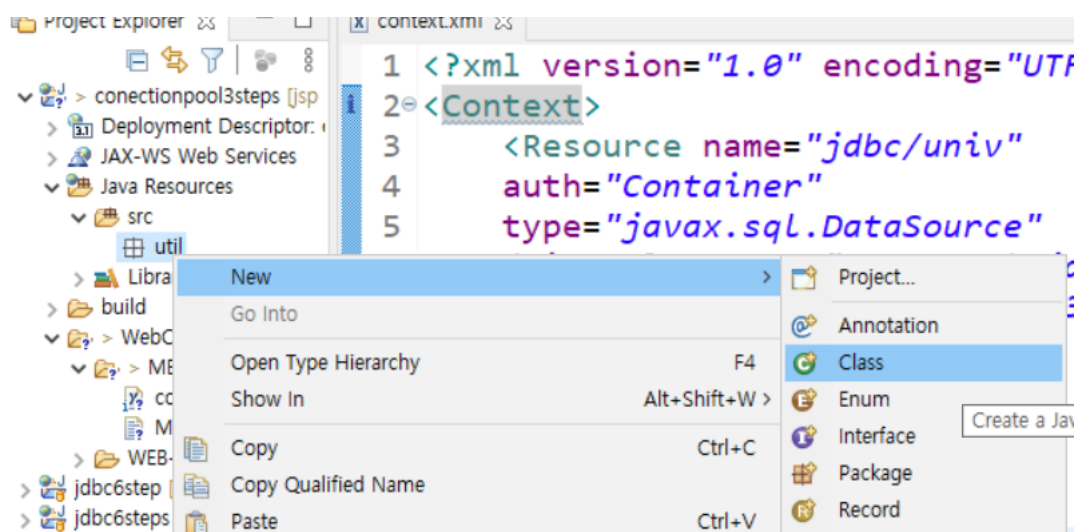
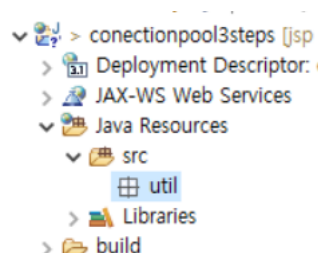
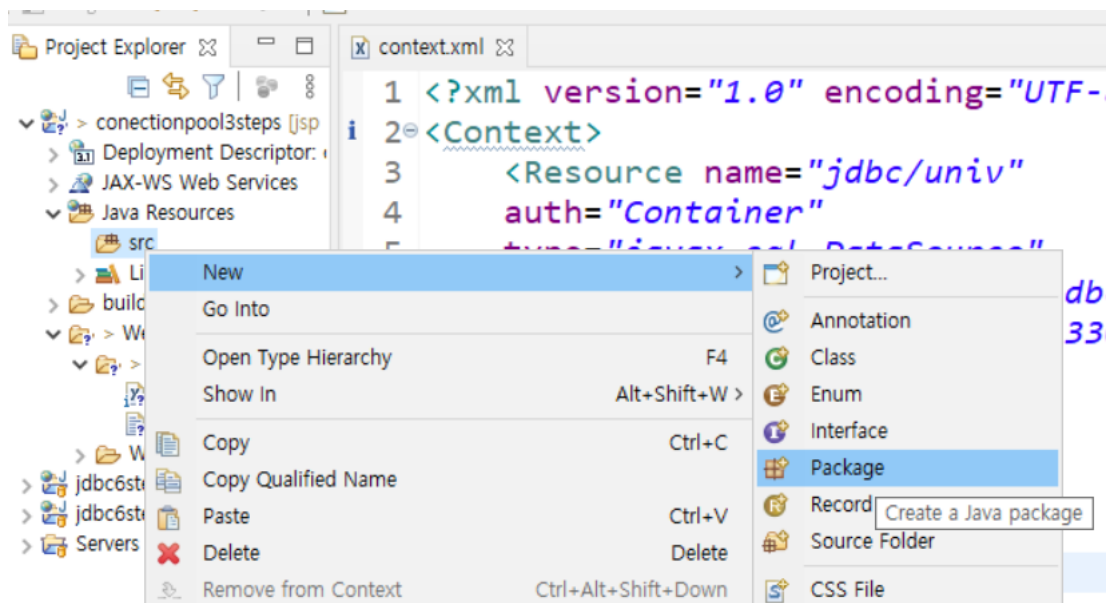
위치는 WebContent > WEB-INF > **lib**

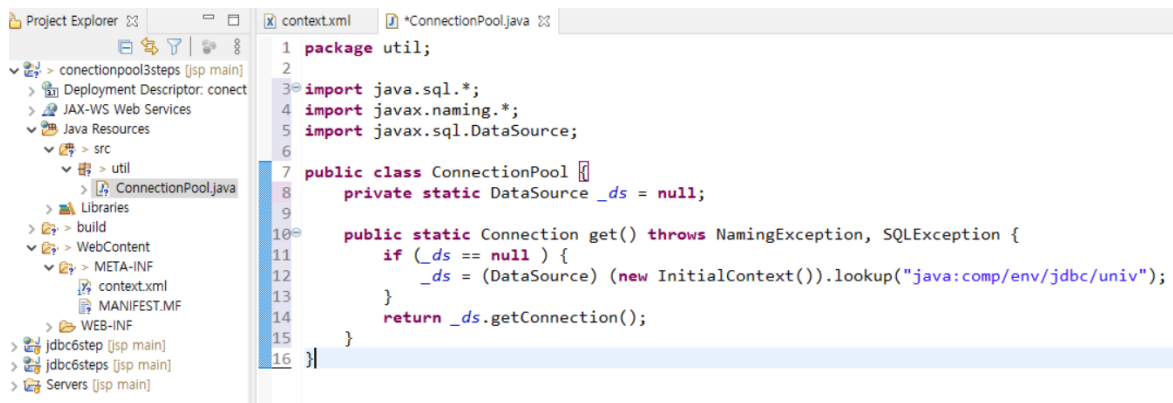


▼ 2. ConnectionPool.java

정의된 내용으로 실제 디비와 연결 해주는 객체를 생성하기 위한 클래스 작성

위치 src - util 패키지 생성





```
package util;

import java.sql.*;
import javax.naming.*;
import javax.sql.DataSource;

public class ConnectionPool {
    private static DataSource _ds = null;

    public static Connection get() throws NamingException, SQLException {
        if (_ds == null ) {
            _ds = (DataSource) (new InitialContext()).lookup("java:comp/env/jdbc/univ"); <- 디비명만 바뀐다.
        }
        return _ds.getConnection();
    }
}
```

위 3단계로 Connection Pool 사용 설정 완료

▼ 3. JDBC connector driver

Connection Pool 적용

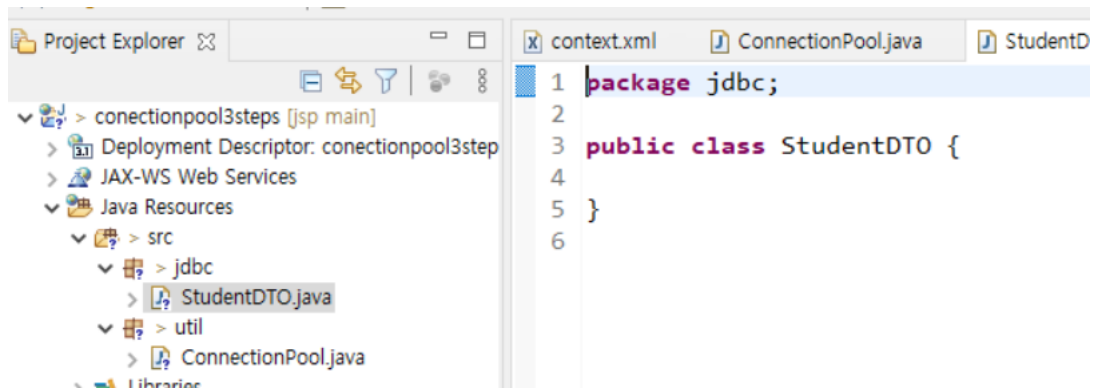
항상 DB 설계부터 시작!

▼ 1. DTO(Data Transfer Object)

DTO > Getter + Setter + 생성자

VO > Getter + 생성자

사실 DTO 는 디비에서 데이터를 꺼낼때만 사용된다. DTO 파일은 데이터베이스의 테이블의 필드와 일대일 매칭이 되게 설계한다.



테이블의 필드명으로 변수를 private 접근제한자로 생성하고 Getter와 Setter 그리고 생성자를 만든다.

```
package jdbc;

public class StudentDTO {

    private String hakbun;
    private String name;
    private String dept;
    private String addr;

    public String getHakbun() {
        return hakbun;
    }
    public void setHakbun(String hakbun) {
        this.hakbun = hakbun;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getDept() {
        return dept;
    }
    public void setDept(String dept) {
        this.dept = dept;
    }
    public String getAddr() {
        return addr;
    }
    public void setAddr(String addr) {
        this.addr = addr;
    }

    public StudentDTO(String hakbun, String name, String dept, String addr) {
        super();
        this.hakbun = hakbun;
        this.name = name;
        this.dept = dept;
        this.addr = addr;
    }

}
```

▼ 2. DAO(Data Access Object)

실제 DB와 연결되는 매서드 등과 SQL 쿼리 등을 작성하게 된다.

```
package jdbc;

import java.sql.*;
import javax.naming.NamingException;
import util.*;
```



```

public class StudentDAO {

    //테이블에 데이터를 입력하는 매서드
    public static int insert(String hakbun, String name, String dept, String addr)
        throws NamingException, SQLException {

        //C R U D

        String sql = "INSERT INTO student VALUES(?,?,?,?)";

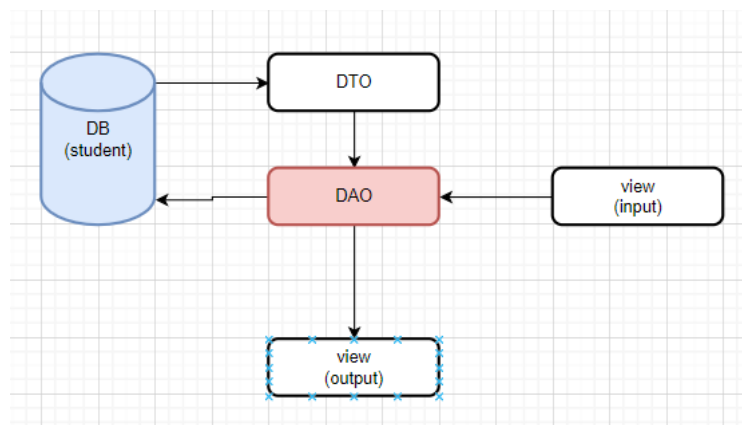
        Connection conn = ConnectionPool.get(); //커넥션 풀 사용

        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, hakbun);
        pstmt.setString(2, name);
        pstmt.setString(3, dept);
        pstmt.setString(4, addr);

        int result = pstmt.executeUpdate();
        // SQL 구문 실행 성공 여부가 1과 0으로 돌아온다.

        return result;
    }
}

```



▼ 3. View

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<form action="TBInsert.jsp" method="get">
학번 <input type="text" name="hakbun"> <br><br>
이름 <input type="text" name="name"> <br><br>
전공 <input type="text" name="dept"> <br><br>
주소 <input type="text" name="addr"> <br><br>
<button type="submit">insert</button>
</form>

</body>
</html>

```

```

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

```

```

    pageEncoding="UTF-8"%>

<%-- <!-- Step 1 import SQL Packages -->
<%@ page import="java.sql.*" %> --%>

<% // 전송 받는 데이터 한글 처리
    request.setCharacterEncoding("UTF-8");
%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
/* //Step 2 load JDBC Driver
    try {
        Class.forName("com.mysql.jdbc.Driver");
    }catch(ClassNotFoundException err) {
        out.print("JDBC Driver loading error<br>" + err.getMessage());
    }

// Step 3 create Connection Object

    Connection conn = null;

    try {
        conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/univ","root","0000");
    }catch(SQLException err) {
        out.print("Connection Object error<br>" + err.getMessage());
    } */

// Step 4 create Statement Object

/*    String hakbun = "1111";
    String name = "홍길동";
    String dept = "컴공";
    String addr = "서울"; */

    String hakbun = request.getParameter("hakbun");
    String name = request.getParameter("name");
    String dept = request.getParameter("dept");
    String addr = request.getParameter("addr");

/*    String sql ="INSERT student VALUES(?, ?, ?, ?)" ;

    PreparedStatement pstmt = conn.prepareStatement(sql);
    pstmt.setString(1, hakbun);
    pstmt.setString(2, name);
    pstmt.setString(3, dept);
    pstmt.setString(4, addr);
    */

// Step 5 excute SQL Query

/*    pstmt.executeUpdate(); */

// Step 6 close Connection
/*
    pstmt.close();
    conn.close(); */    <- 커넥션 풀 사용에 따라 전부 필요 없어짐

    int result = StudentDAO.insert(hakbun, name, dept, addr);

    if (result == 1) {
        out.print("등록 성공");
    } else {
        out.print("등록 실패");
    }

%>
</body>
</html>

```

▼ 학생테이블 커넥션 풀 적용(Example)

TBform.jsp

TBinsert.jsp

DAOstudent.java

DTOstudent.java

TBlist.jsp

한 명 한 명의 데이터를 하나의 객체로 만들어 배열로 담는다.

```
public static ArrayList<StudentDTO> getList()
    throws NamingException, SQLException {

    String sql = "SELECT * FROM student";

    Connection conn = ConnectionPool.get();

    PreparedStatement pstmt = conn.prepareStatement(sql);

    ResultSet rs = pstmt.executeQuery();

    ArrayList<StudentDTO> students = new ArrayList<StudentDTO>();

    while(rs.next()) {
        students.add(new StudentDTO(rs.getString(1),
                                    rs.getString(2),
                                    rs.getString(3),
                                    rs.getString(4)));
    }

    return students;
}
```

```
<%@page import="jdbc.*"%>
<%@page import="java.util.ArrayList"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>학생 목록</title>
</head>
<body>
<%

ArrayList<StudentDTO> students = StudentDAO.getList();

for (StudentDTO student : students) {
%>

<%=student.getHakbun() %>|
<%=student.getName() %>|
|<br>

<%
}
%>

</body>
</html>
```

TBdetail.jsp

학생 목록에서 각각의 학번에 링크 걸기

```
<a href="TBDetail.jsp?hakbun=<%=student.getHakbun() %>"><%=student.getHakbun() %></a>|  
<%=student.getName() %><br>
```

```
public static StudentDTO getDetail(String hakbun)  
    throws NamingException, SQLException {  
  
    String sql = "SELECT * FROM student WHERE hakbun=?";  
  
    Connection conn = ConnectionPool.get();  
  
    PreparedStatement pstmt = conn.prepareStatement(sql);  
    pstmt.setString(1, hakbun);  
  
    ResultSet rs = pstmt.executeQuery();  
  
    rs.next();  
  
    String name = rs.getString(2);  
    String dept = rs.getString(3);  
    String addr = rs.getString(4);  
  
    StudentDTO student = new StudentDTO(hakbun, name, dept, addr);  
  
    return student;  
}
```

▼ 커넥션 풀 적용 게시판테이블(새로운 DB)

Board Table

DB 설계

테이블 명 : board

글번호 bno 100

제목 btitle 100

작성자 bwriter 50

내용 bcontent 500

날짜 bdate x

테이블에 자동 증가 번호 넣기

- 데이터 유형 - INT
- 기본값 - AUTO_INCREMENT

이름: board

코멘트:

열: + 추가 x 제거 ▲ 위로 ▼ 아래로

#	이름	데이터 유형	길이/설정	부호 없...	NULL 허...	0으로...	기본값	코멘트
1	bno	INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/> 기본값 없음 <input type="radio"/> Custom text: <input type="text"/> <input type="radio"/> NULL <input type="radio"/> 표현식: <input type="text"/> On update: <input type="text"/> <input checked="" type="radio"/> AUTO_INCREMENT <input type="button" value="확인"/> <input type="button" value="취소"/>	
2	btitle	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
3	bwriter	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
4	bcontent	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
5	bdate	TIMESTAMP		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		

기본키 설정

이름: board

코멘트:

열: + 추가 x 제거 ▲ 위로 ▼ 아래로

#	이름	데이터 유형	길이/설정	부호 없...	NULL 허...	0으로...	기본값	코멘트
1	bno	INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT	
2	btitle	VARCHAR		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
3	bwriter	VARCHAR		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
4	bcontent	VARCHAR		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
5	bdate	TIMESTAMP		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMEST...	

복사(C) Ctrl+C
선택한 열 복사(S)

열 붙여넣기(T)
열 추가(U) Ctrl+Ins

열 제거(V) Ctrl+Del
위로(W) Ctrl+U

아래로(X) Ctrl+D
새 인덱스 생성(Y)

인덱스에 추가(Z)

PRIMARY
KEY
UNIQUE
FULLTEXT
SPATIAL

테이블에 데이터가 입력될때 자동으로 그시간을 입력한다.

- 데이터 유형 - TIMESTAMP
- 기본값 - 표현식 - CURRENT_TIMESTAMP()

이름:

코멘트:

결: + 추가 x 제거 ▲ 위로 ▼ 아래로

#	이름	데이터 유형	길이/설정	부호 없...	NULL 허...	0으로...	기본값	코멘트
1	bno	INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	기본값 없음	
2	btitle	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
3	bwriter	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
4	bcontent	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
5	bdate	TIMESTAMP		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		

○ 기본값 없음

○ Custom text:

○ NULL

● 표현식:

CURRENT_TIMESTAMP

On update:

○ AUTO_INCREMENT

BoardDTO 작성

```
package jdbc;

public class BoardDTO {

    private String bno;
    private String btitle;
    private String bwriter;
    private String bcontent;
    private String bdate;

    public String getBno() {
        return bno;
    }
    public String getBtitle() {
        return btitle;
    }
    public String getBwriter() {
        return bwriter;
    }
    public String getBcontent() {
        return bcontent;
    }
    public String getBdate() {
        return bdate;
    }

    public BoardDTO(String bno, String btitle, String bwriter, String bcontent, String bdate) {
        super();
    }
}
```

```

        this.bno = bno;
        this.btitle = btitle;
        this.bwriter = bwriter;
        this.bcontent = bcontent;
        this.bdate = bdate;
    }

}

```

Connection Pool 설정 3단계

context.xml

connectionPool.java

JDBC Driver

BoardDAO 작성

```

package jdbc;

import java.sql.*;

import javax.naming.NamingException;

import util.*;

public class BoardDAO {

    public static int insert(String btitle, String bwriter, String bcontent) throws NamingException, SQLException {

        String sql = "INSERT INTO board (btitle, bwriter, bcontent) VALUES(?,?,?)";

        Connection conn = ConnectionPool.get();

        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, btitle);
        pstmt.setString(2, bwriter);
        pstmt.setString(3, bcontent);

        int result = pstmt.executeUpdate();

        return result;
    }

}

```

BoardInsert.jsp

```

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

<%
    String btitle = request.getParameter("btitle");
    String bwriter = "작성자";
    String bcontent = request.getParameter("bcontent");

    int result = BoardDAO.insert(btitle, bwriter, bcontent);

    if (result == 1) {
        out.print("등록 성공");
    } else {
        out.print("등록 실패");
    }

%>

```

BoardForm.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-Gh"

<div class="container">

<form action="BoardInsert.jsp">
<div class="mb-3">
    <label for="exampleFormControlInput1" class="form-label">제목</label>
    <input type="text" name = "btitle" class="form-control" id="exampleFormControlInput1">
</div>
<div class="mb-3">
    <label for="exampleFormControlTextarea1" class="form-label">내용</label>
    <textarea class="form-control" name = "bcontent" id="exampleFormControlTextarea1" rows="3"></textarea>
</div>
<button type="submit" class="btn btn-primary">등록</button>

</form>

</div>

</body>
</html>
```

BoardList.jsp

```
<%@page import="jdbc.*"%>
<%@page import="java.util.ArrayList"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-Gh"

<div class="container">
<table class="table table-hover">
    <thead>
        <tr>
            <th scope="col">번호</th>
            <th scope="col">제목</th>
            <th scope="col">작성자</th>
            <th scope="col">날짜</th>
        </tr>
    </thead>
    <tbody>
<%

    ArrayList<BoardDTO> boards = BoardDAO.getList();

    for (BoardDTO board : boards) {

%>
        <tr>
            <th scope="row"><%=board.getBno() %></th>
            <td><%=board.getBtitle() %></td>
            <td><%=board.getBwriter() %></td>
            <td><%=board.getBdate() %></td>
        </tr>

<%
    }
}
```



```
%>

</tbody>
</table>
</div>
</body>
</html>
```

BoardDetail.jsp

TBlist.jsp

Summernote 적용

Summernote - Super Simple WYSIWYG editor

Super Simple WYSIWYG Editor on Bootstrap Summernote is a JavaScript library that helps you create WYSIWYG editors online.

<https://summernote.org/>



[summernote](#)
[Getting started](#)
[Deep dive](#)
[Examples](#)
[Plugins](#)
[Team](#)

[#summernote](#)
[10,998](#)

Get Summernote

Installation

[Requires HTML5 doctype](#)
[Include js/css](#)
[Embed](#)
[Run summernote](#)
[Simple example](#)
[For Bootstrap 4](#)

For bootstrap 5

[Without Bootstrap \(file\)](#)

[Basic API](#)
[i18n support](#)
[Integration](#)

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Summernote with Bootstrap 5</title>
  <!-- include Libraries(jQuery, bootstrap) -->
  <script type="text/javascript" src="//code.jquery.com/jquery-3.6.0.min.js"></script>

  <link rel="stylesheet" href="//cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" />
  <script type="text/javascript" src="cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"></script>

  <!-- include summernote css/js -->
  <link href="summernote-bs5.css" rel="stylesheet">
  <script src="summernote-bs5.js"></script>

</head>
<body>
  <div id="summernote"></div>
  <script>
    $('#summernote').summernote({
      placeholder: 'Hello Bootstrap 5',
      tabsize: 2,
      height: 100
    });
  </script>
</body>
</html>
```

- 주의사항


1. DB에 필드 사이즈를 크게 LONGTEXT 로 사용해야만 하고
2. 전송 방식을 "post"로 설정해야만 한다.

- 모바일 화면 보기

```
<meta name="viewport" content="width=device-width, initial-scale=1" />
```


경기도지식(GSEEK)

경기도 무료 온라인 평생학습서비스, 외국어, 자격취득, 생활/취미, 부모교육, 청소년 등 제공

 <https://www.gseek.kr/member/rl/search/result.do?menuId=&menuStep=&pMenuId=&searchText=%EC%A0%95%EB%B3%B4%EC%B2%98%EB%A6%AC%EA%B8%B0%EC%82%AC>



TomatoVocachipMiniBook.pdf

 <https://drive.google.com/file/d/13q1mK8XfaPJbKLBpaJILZbsRkx57N5FA/view?usp=drivesdk>

