

Class 230116(Connection Pool)

Connection Pool

왜 Connection Pool인가?

- JDBC를 사용할 때 리소스가 가장 많이 소모되는 부분이 DB연동에 필요한 Connection 객체를 생성하는 부분임
- 지금까지 방법들은 모두 JSP에서 SQL구문을 수행하기 위해서 Connection 객체를 생성하고 사용 후 제거하는 과정을 반복해왔음
- 접속자가 많아질 경우 시스템의 성능이 급격히 떨어짐
- 이러한 문제점을 해결하기 위한 방법으로 커넥션 풀을 이용하게 됨

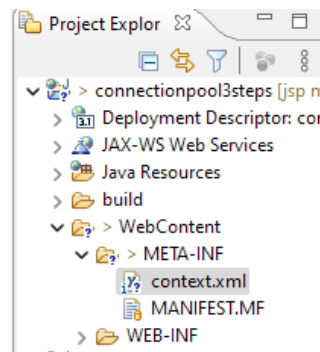
원리

- 사용자가 접속할 때마다 매번 새로운 Connection 객체를 생성하는 것이 아니라
- 일정 갯수의 Connection 객체를 미리 생성해 놓고
- 사용자의 요청이 있을 때마다 가용한 객체를 할당하고
- 다시 회수하는 방식

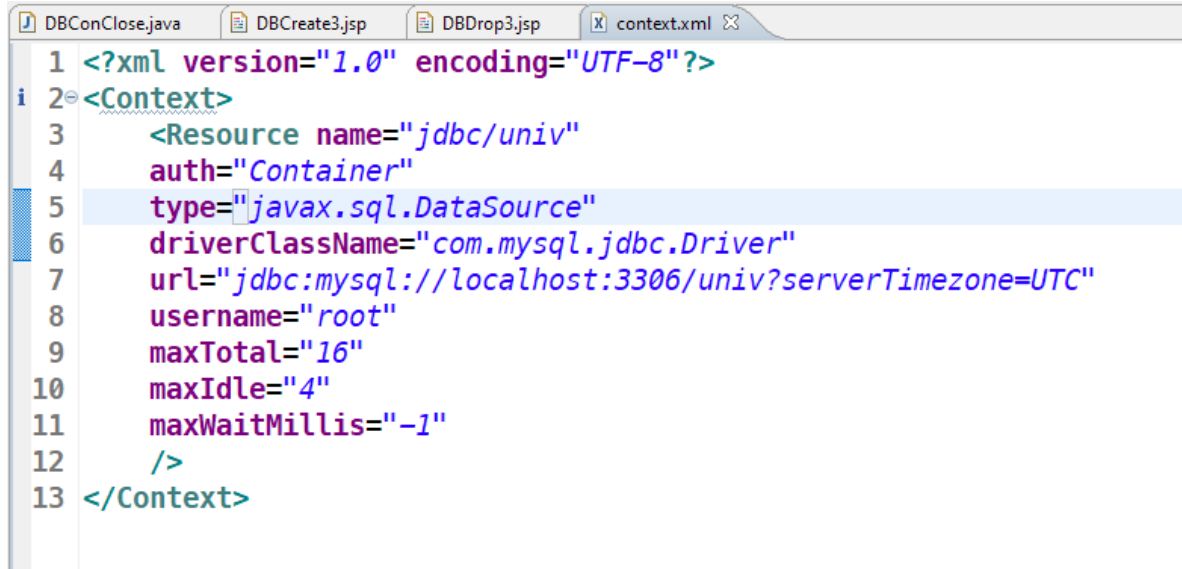
Connection Pool 설정

1. context.xml

- 데이터베이스에 대한 커넥션 풀을 사용하기 위한 **설정을 정의**한다
- 위치 : WebContent → META-INF → context.xml



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     maxTotal="16"
10    maxIdle="4"
11    maxWaitMillis="-1"
12  />
13 </Context>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
  <Resource name="jdbc/univ"
    auth="Container"
    type="javax.sql.DataSource"
    driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/univ?serverTimezone=UTC"
    username="root"
    password="00000"
    maxTotal="16"
    maxIdle="4"
    maxWaitMillis="-1"
  />
</Context>
```

- `name` : 사용할 DB이름
- `auth/type` : 그냥 언제나 똑같음
- `driverClassName` : 연결할 DB
- `url` : 주소
- `serverTimezone` : 시간대 설정(여기서는 그리니치 표준시로 설정)
- `username/password` : DB아이디/비밀번호(호스팅 업체 업로드시에는 변경해야 함)
- `maxTotal` : 미리 생성할 커넥션의 갯수
- `maxIdle` : 최저 유지 커넥션 갯수
- `maxWaitMillis` : 기다리는 시간, -1하면 기다리지 않고 바로바로 처리

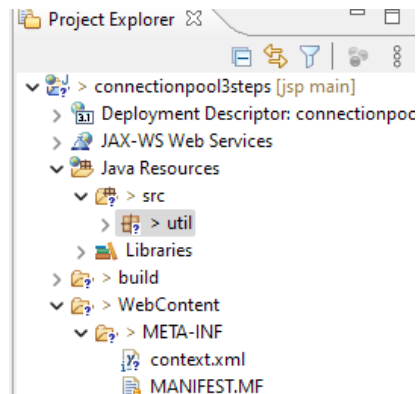
`serverTimezone`

- 생략해도 되는 경우도 있음

- 그러나 특정 서버에서는 타임존 설정을 하지 않으면 동작하는 않는 경우가 있음. 나중에 찾아라 고생하지 말고 미리미리 써 두자

2. ConnectionPool.java

- 정의된 내용으로 실제 DB와 연결해주는 객체를 생성하기 위한 클래스
- 위치 : Java Resources → src → util → ConnectionPool.java



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

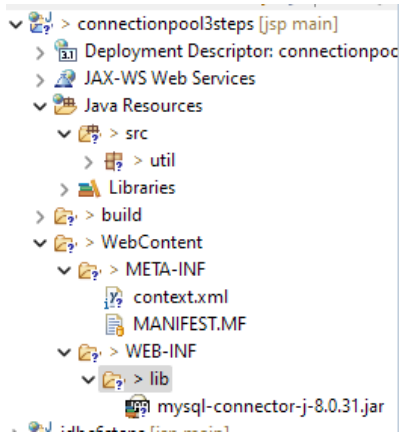
public class ConnectionPool {
    private static DataSource _ds = null;

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

- 데이터베이스 이름 외에 달라질 게 전혀 없음

3. JDBC connector driver

- WEB-INF → lib 안에 jar파일 넣어주기

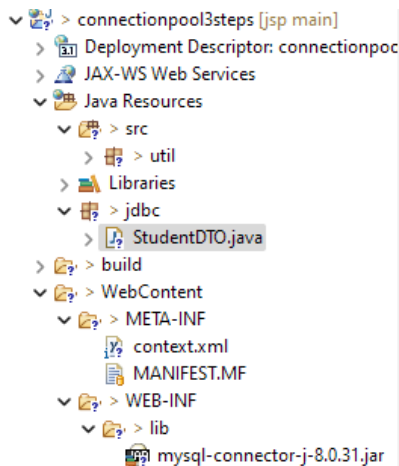


위 3단계로 Connection Pool 사용 설정 완료!!

Connection Pool 적용

4. DTO (Data Transfer Object)

- 원래는 DB에서 데이터를 꺼낼때만 사용됨
- 데이터베이스의 테이블의 필드와 일대일 매칭이 되게 설계
- 테이블을 Class화 한다(이전에 VO로 쓰던것과 동일한듯)
- private로 변수 만들어두고 Getter/Setter/Constructor 생성



▼ StudentDTO

```

package jdbc;

public class StudentDTO {

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

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

```

Q. 왜 변수가 private인가?

- 외부에서 이 값들을 직접 건드리지 않도록 함
- setter와 getter메서드를 통해서만 접근이 가능하므로 **보안**에 훨씬 나옴

5. DAO (Data Access Object)

- 실제 DB와 연결되는 메서드, SQL 쿼리 등을 작성하게 됨

```

Connection conn = ConnectionPool.get();

```

- 애만 쓰면 데이터베이스와 연결된다

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

PreparedStatement pstmt = conn.prepareStatement(sql);
```

- 쿼리를 적고 `conn` 객체의 메서드로 `PreparedStatement` 객체를 받아온다
- 이후에는 값 세팅하고 `executeUpdate()` 하면 됨

▼ StudentDAO

```
package jdbc;

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

public class StudentDAO {

    public static int insert(String hakbun, String name, String dept, String addr) throws NamingException, SQLException {

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

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

        PreparedStatement pstmt = conn.prepareStatement(sql);

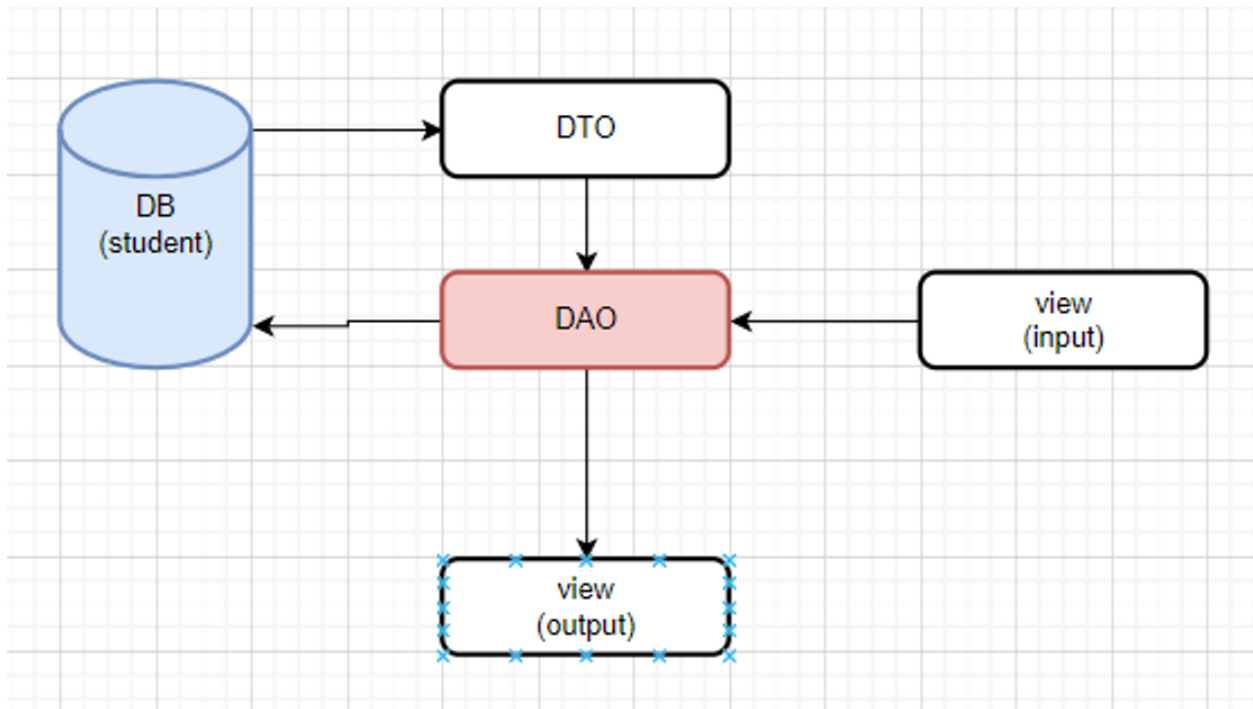
        pstmt.setString(1, hakbun);
        pstmt.setString(2, name);
        pstmt.setString(3, dept);
        pstmt.setString(4, addr);

        int result = pstmt.executeUpdate();

        return result;
    }
}
```

DB 접근 때 주로 쓰는 구조

- 데이터를 넣을 때는 `View` → `DAO` → `DB`
- 데이터를 꺼낼 때는 `DB` → `DTO` → `DAO` → `View` 구조로



6. View

Insert

- 저번에 썼던 `TBForm` 과 `TBInsert` 를 그대로 가져와서 해보자

▼ TBForm

```

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

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

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

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

```

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

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

%>

</body>
</html>

```

| `StudentDAO` 가 있는 패키지를 꼭 임포트 할 것

| `StudentDAO` 를 `static` 처리했으므로 클래스명으로 바로 불러서 쓸 수 있음

| 매개변수로 넣어주는 것 만으로 끝!

SelectAll

- 값들 하나하나를 studentDTO객체로 받아와서
- 그걸 Array에 넣어서 받아올 것임

▼ StudentDAO

```

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;
}

```

| 결과를 받아와야 할 때는 `executeQuery()`

| 결과값은 `ResultSet` 객체에 들어감

| `rs.next()` 다음 값이 있다면 `true`

| 생성자를 바로 써서 값을 받아줌

| `rs.getString()` 에 매개변수로 `int` 를 넣어주면 그 순서대로 가져옴

▼ TBList

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

    for (StudentDTO student : students) {
        %>

        <tbody>
            <tr>
                <td><%=student.getHakbun()%></td>
                <td><%=student.getName()%></td>
                <td><%=student.getDept()%></td>
                <td><%=student.getAddr()%></td>
            </tr>
        </tbody>
        <%
            }
        %>
```

| `ArrayList` 인 `students` 에서 for each 구문으로 하나씩 받아오기

| `DTO` 의 `Getter` 로 정보를 받아온다는 것 기억하기

- 목록에는 두개만 뜸
- 학번을 누르면 디테일이 뜨도록 해보자

하나만 가져오기

```
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();

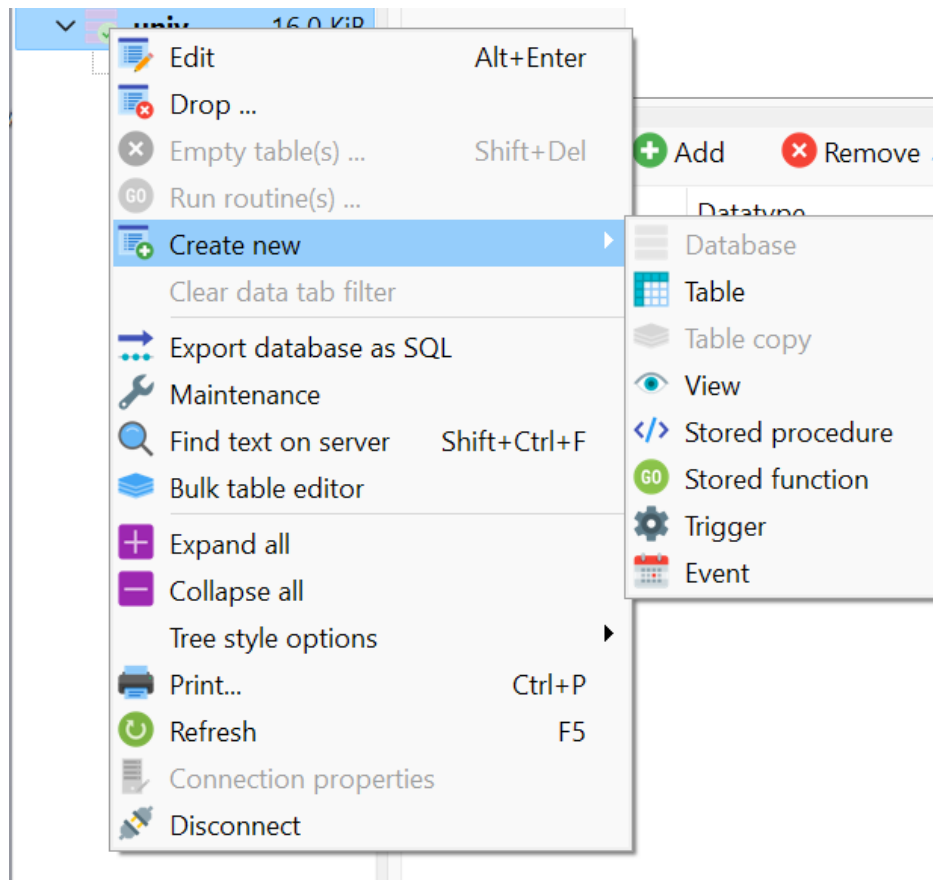
    StudentDTO student = new StudentDTO(rs.getString(1), rs.getString(2), rs.getString(3), rs.getString(4));
    return student;
}
```

어차피 한개니까 `rs.next()` 한 번만 해줘도 됨

Connection Pool 적용하여 게시판 만들어보기

1. DB만들기

- Heidi로 만들면 몹시몹시 간단하다



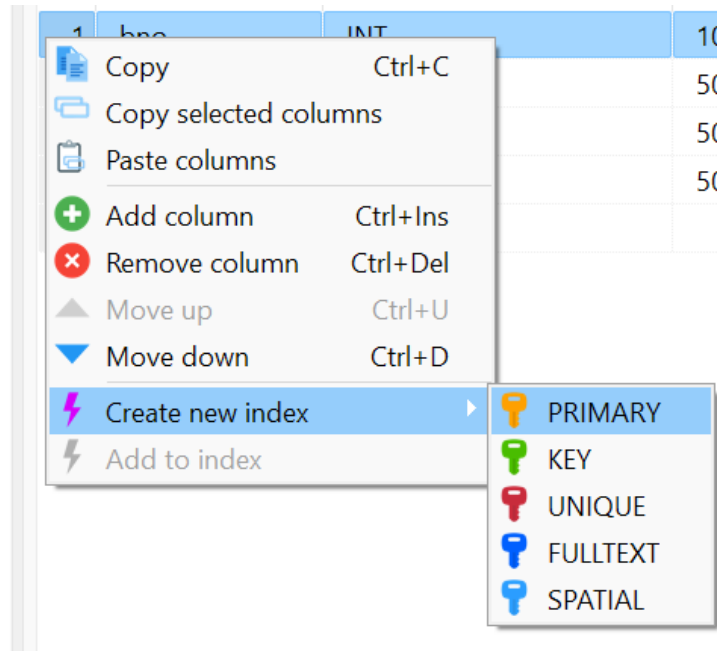
- 데이터베이스에서 **Create new** → **Table**

#	Name	Datatype	Length/Set	Unsigned	Allow NU...	Zerofill	Default	Comment
1	bno	INT	10	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="radio"/> No default value <input type="radio"/> Custom text: <input type="text"/> <input type="radio"/> NULL <input type="radio"/> Expression: <input type="text"/>	
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"/>	<input checked="" type="radio"/> AUTO_INCREMENT <input type="button" value="OK"/> <input type="button" value="Cancel"/>	

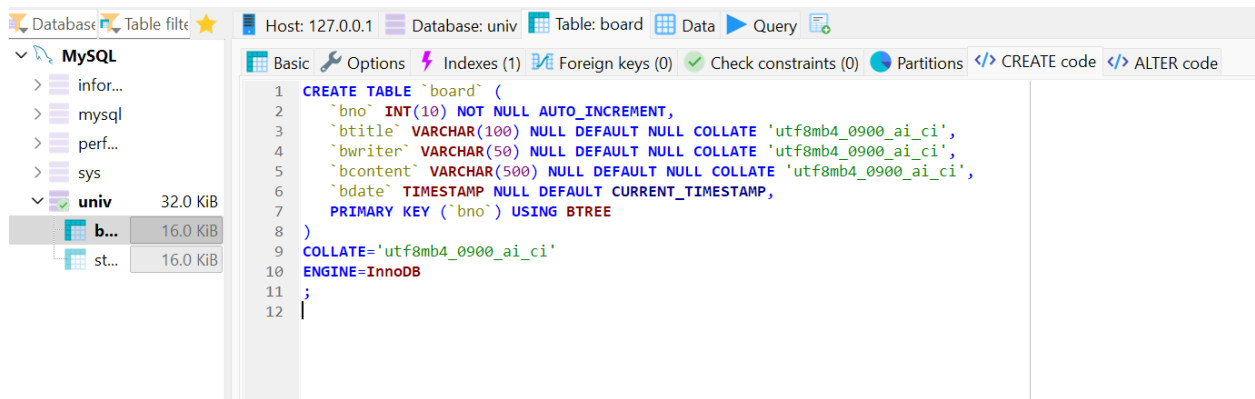
- **Default** → **AUTO_INCREMENT** 설정해주면 자동으로 하나씩 올라가게 할 수 있다

Host: 127.0.0.1 Database: univ Table: [Untitled] Query									
Basic Options Indexes (0) Foreign keys (0) Check constraints (0) Partitions CREATE code									
Name: board Comment:									
Columns: + Add - Remove ▲ Up ▼ Down									
#	Name	Datatype	Length/Set	Unsigned	Allow NU...	Zerofill	Default	Comment	Collation
1	bno	INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	No default		
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"/>	CURRENT_TIMEST...		

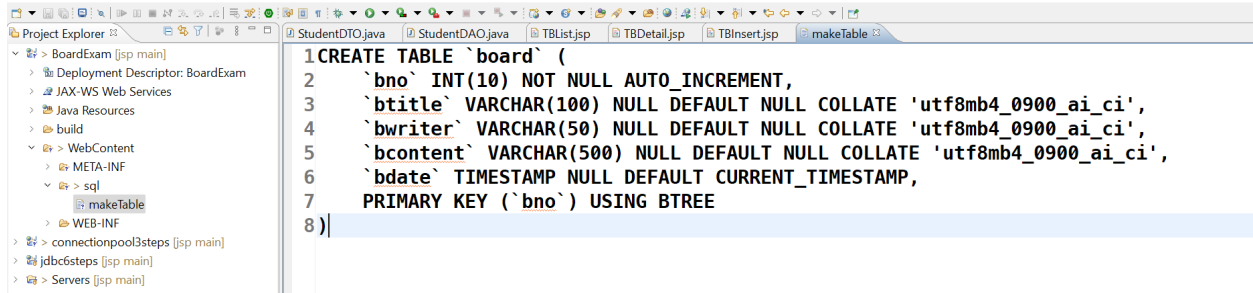
- 날짜의 Datatype을 **Timestamp**
- Default → Expression에 **CURRENT_TIMESTAMP**
- 데이터가 들어올 때의 시간을 자동으로 찍어준다



- Primary key 설정하기



- Table → CREATE code 로 들어가면 현재 테이블을 만들 수 있는 코드가 그대로 나온다



- 이렇게 하나 넣어두면 다음에도 그대로 쓰면 되니 만들기 쉬워서 좋다

Q. DTO에서 번호나 날짜를 그냥 String으로 받나?

- DTO에서 꺼내고 넣을 때마다 자료형을 바꾸는 것보다
- String으로 쓰고 필요할 때 형변환 하는게 낫다

▼ BoardDTO

```
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;
    }
}
```

Tip

- 솔직히 setter를 쓸 일이 없으면 없애는 게 보안에 더 좋다.
- DAO에서 바로 데이터베이스 업데이트를 하고 생성자로 데이터를 받아오면 setter 쓸 필요도 없다. 없애자.

Insert 만들기

▼ BoardDAO, Insert 메서드

```
public static int insert(String title, String writer, String contents) throws SQLException, NamingException {  
  
    String sql = "INSERT INTO board (btitle, bwriter, bcontent) VALUES (?, ?, ?)";  
  
    Connection conn = ConnectionPool.get();  
  
    PreparedStatement pstmt = conn.prepareStatement(sql);  
    pstmt.setString(1, title);  
    pstmt.setString(2, writer);  
    pstmt.setString(3, contents);  
  
    int result = pstmt.executeUpdate();  
  
    return result;  
  
}
```

▼ BoardForm.jsp (부트스트랩까지 써봄)

```
<%@ 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>  
    <link  
        href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css"  
        rel="stylesheet"  
        integrity="sha384-rbsA2VBKQhggwzXH7pPCaAq046Mgn0M80zw1RWuH61DGLwZJEdK2Kadq2F9CUG65"  
        crossorigin="anonymous">  
    </head>  
<body>  
    <div class="container">  
        <form action="BoardInsert.jsp">  
            <div class="mb-3">  
                <label for="btitle" class="form-label">제목</label> <input type="text"  
                    class="form-control" name="btitle" id="btitle">  
            </div>  
            <div class="mb-3">  
                <label for="bcontent" class="form-label">내용</label>  
                <textarea class="form-control" id="bcontent"  
                    rows="3" name="bcontent"></textarea>  
            </div>  
        </form>  
    </div>  
</body>  
</html>
```

```

</div>
<div class="text-end">
  <button type="submit" class="btn btn-info">제출</button>
</div>
</form>
</div>
</body>
</html>

```

Select List 만들기

▼ BoardDAO → getList

```

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

    String sql = "SELECT * FROM board";

    Connection conn = ConnectionPool.get();
    PreparedStatement pstmt = conn.prepareStatement(sql);

    ResultSet rs = pstmt.executeQuery();

    ArrayList<BoardDTO> boards = new ArrayList<BoardDTO>();

    while (rs.next()) {
        boards.add(new BoardDTO(rs.getString(1), rs.getString(2), rs.getString(3), rs.getString(4), rs.getString(5)));
    }

    return boards;
}

```

▼ BoardList (부트스트랩 이용)

```

<%@page import="java.util.ArrayList"%>
<%@page import="jdbc.*"%>
<%@ 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>
<link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css"
    rel="stylesheet"
    integrity="sha384-rbsA2VBKQhggwzxH7pPCaAq046MgnOM802w1RWuH61DGLwZJEdK2Kadq2F9CUG65"
    crossorigin="anonymous">
</head>
<body>
<div class="container mt-3">
<table class="table table-striped table-hover">
  <thead>
    <tr>
      <th scope="col">번호</th>

```

```

        <th scope="col">제목</th>
        <th scope="col">내용</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.getBcontent()%></td>
            <td><%=board.getBdate()%></td>
        </tr>

<%
    }
%>
</table>
</div>
</body>
</html>

```

번호	제목	내용	날짜
1	안녕하세요	안녕안녕하세요	2023-01-16 15:21:58
2	인사 한번 더 올립니다	반갑습니다 회원님들^^	2023-01-16 15:30:26
3	인사 한번 더 올립니다	반갑습니다 회원님들^^	2023-01-16 15:30:30
4	인사 두 번 더 올립니다	오늘도 반갑습니다 여러분^^	2023-01-16 15:30:43
5	인사 세 번 올립니다	세 번 봐도 반갑습니다^^	2023-01-16 15:30:58

SummerNote















맛보기

썸머노트 링크

- Getting started 들어가서 안내대로만 하면

title

content

Hello stand alone ui

submit

- 짜잔 이런 노트를 내 웹페이지에 넣을 수 있다


주의사항

썸ernote로 들어간 정보를 보면 색이나 데이터나 등등 온갖 정보를 다 코드로 넣는 것을 볼 수 있다.

이 때문에 주의사항이 몇 가지 생긴다

1. DB에 Datatype LONGTEXT 로 사용해야만 함

- 많은 정보가 들어가도록 컬럼의 사이즈를 왕창 크게 만들기 위함.

Columns: + Add × Remove ▲ Up		
#	Name	Datatype
 1	bno	INT
2	btitle	VARCHAR
3	bwriter	VARCHAR
4	bcontent	LONGTEXT
5	bdate	TIMESTAMP

2. 전송 방식을 “post”로 설정해야만 함.

- get으로 하면 header가 너무 커질 수 있기 때문

뷰포트 설정에 관해

- vsCode에서 html 기본설정을 하면

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

애를 기본으로 준다. 이게 뭘까?

- viewport란 화면 상의 화상 표시 영역을 뜻한다
- 데스크탑에 기반하여 설계된 웹페이지를 그냥 모바일에서 보면 기본 뷰포트가 980px로 커서 내용이 작게 보인다
- 즉 뷰포트 설정으로 다양한 기기에서 페이지의 너비나 화면 배율을 설정하는 것

content="width=device-width"

- **웹 페이지의 너비**를 장치 너비에 맞추어 표시한다는 뜻
- 모바일 등 다양한 환경에서 화면 너비에 맞는 페이지를 보여줄 수 있음

initial-scale=1.0

- 초기 화면 배율을 100%, **장치의 화면 크기**와 **뷰포트 크기**를 동일하게 맞춘다는 뜻
- 웹페이지가 처음 로드될 때 크기가 맞춰져 나오므로 확대, 축소 없이 화면에 페이지가 꽉 채워져 나옴