

Class 230116 - Connection Pool

Connection Pool

JDBC를 사용할 때 가장 많이 사용되는 리소스, 즉 자원이 소모되는 부분이 DB연동에 필요한 Connection 객체를 생성하는 부분이다.

지금까지 방법은 모두 JSP에서 SQL구문을 수행하기 위해서 Connection 객체를 생성하고 사용 후 제거하는 과정을 반복해왔으며, 이러한 과정은 접속자가 많아질 경우 시스템의 성능을 급격하게 저하시키게 된다.

따라서, 이러한 문제점을 해결하기 위한 방법으로 Connection Pool을 이용하게 된다. 사용자가 접속 할 때마다 새로운 Connection 객체를 생성하는 것이 아니라, 일정 개수의 Connection 객체를 미리 생성해 놓고, 사용자의 요청이 있을 때마다 가용한 객체를 할당하고 다시 회수하는 방식이다.

▼ Connection Pool 사용 설정

1. context.xml 설정

- context.xml 데이터 베이스에 대한 커넥션 풀을 사용하기 위한 설정을 정의한다.

위치 : WebContent > META-INF > context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
  <Resource name="jdbc/univ" <- univ : 사용할 DB명
    auth="Container"
    type="javax.sql.DataSource"
    driverClassName="com.mysql.jdbc.Driver" <- 연결할 때 사용하는 DB (ex; mysql, oracle, maria..)
    url="jdbc:mysql://localhost:3306/univ?serverTimezone=UTC" <- univ : 사용할 DB명
    username="root" <- DB ID (호스팅 업체에 업로드시에는 변경)
    password="00000" <- DB PW (호스팅 업체에 업로드시에는 변경)
    maxTotal="16" <- 미리 생성할 커넥션의 개수
    maxIdle="4" <- 최저 유지 커넥션 개수
    maxWaitMillis="-1"/> <- 기다리는 시간 (항상 -1로 설정해두면, 기다리지 않고 바로 처리)
  </Resource>
</Context>

// ?serverTimezone=UTC : 특정 서버에서 타임존 설정을 하지 않으면 동작하지 않는 경우가 있다.
```

2. 정의된 내용으로 실제 DB와 연결 해주는 객체를 생성하기 위한 클래스 ConnectionPool.java 작성

```
package util; // util 패키지 내부에 생성

















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"); // univ : DB명
        }
        return _ds.getConnection();
    }
}
```

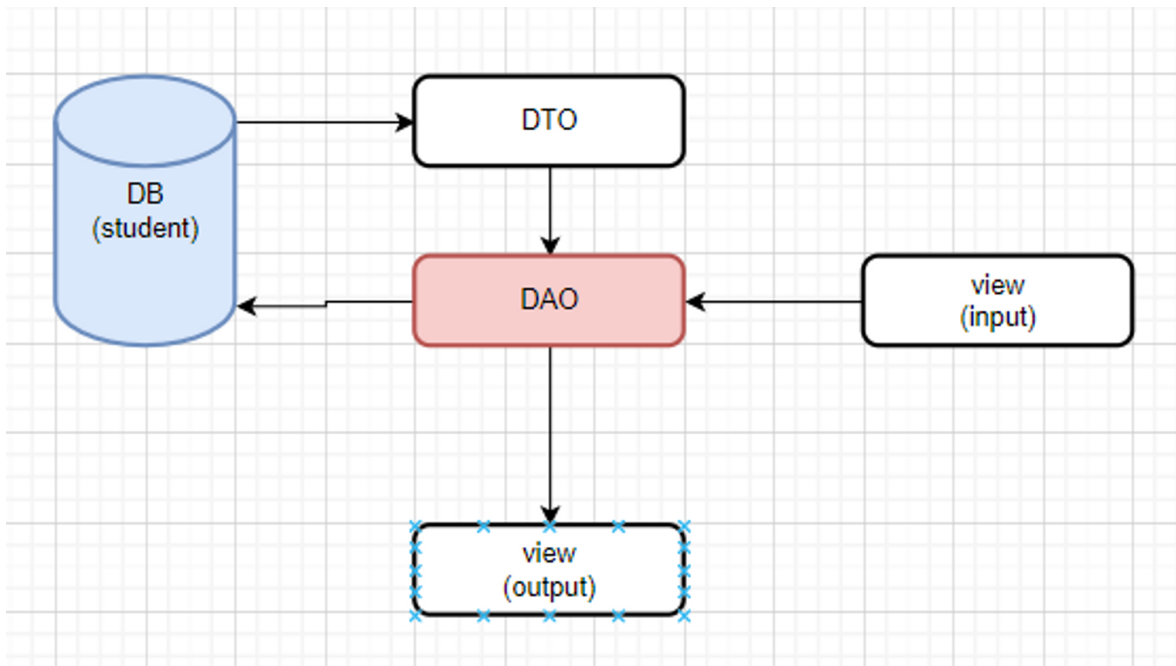
3. JDBC connector driver 설정

WebContent > WEB-INF > lib > mysql-connector jar파일 넣기

- ▼  > conectionpool3steps [jsp main]
 - >  Deployment Descriptor: conectionpool3ste
 - >  JAX-WS Web Services
 - ▼  Java Resources
 - ▼  > src
 - ▼  > util
 - >  ConnectionPool.java
 - >  Libraries
 - >  > build
 - ▼  > WebContent
 - ▼  > META-INF
 -  context.xml
 -  MANIFEST.MF
 - ▼  > WEB-INF
 - ▼  > lib
 -  mysql-connector-j-8.0.31.jar

▼ Connection Pool 적용

- 구조



항상 DB 설계부터 시작해야 하나, 이번에는 지난 시간에 만든 DB와 테이블을 이용하여 Connection Pool 사용

1. DTO(Data Transfer Object) 생성

- DTO는 DB에서 데이터를 꺼낼 때만 사용한다. (따라서 객체에 접근하지 못하도록 getter 메서드만 생성하여도 무방함— 이 경우 VO) DTO파일은 데이터 베이스의 테이블의 필드와 1:1 매칭이 되게 설계한다.
- 테이블의 필드명으로 변수를 private 키워드로 생성하고 getter 와 setter 그리고 생성자를 만든다. (— DTO는 setter와 getter 를 모두 사용)

>> 보안을 위해 setter, getter 를 통해서만 접근할 수 있도록 private 변수로 생성

- 테이블 구조

Table: student

Columns:

<u>hakbun</u>	varchar(10) PK
name	varchar(10)
dept	varchar(20)
addr	varchar(30)

- StudentDTO code

```

package jdbc;

public class StudentDTO {
    //DB 컬럼명에 맞게 필드 생성
    private String hakbun;
    private String name;
    private String dept;
    private String addr;

    //필드를 모두 사용하는 생성자 오버로딩
    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;
    }
}

```

2. DAO(Data Access Object) 생성

- 실제 DB와 연결되는 메서드 등과 SQL 쿼리 등을 작성하게 된다.

▼ INSERT 예시

```

package jdbc;

import java.sql.*;

import javax.naming.NamingException;

import util.*;

public class StudentDAO {

    //테이블에 데이터를 입력하는 메서드
    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;
    }

}

```

▼ SELECT 예시

```

public static ArrayList<StudentDTO> getList() throws NamingException, SQLException {
    String sql = "SELECT * FROM student";

    // 커넥션 풀 사용

```

```

Connection conn = ConnectionPool.get();

PreparedStatement pstmt = conn.prepareStatement(sql);
ResultSet rset = pstmt.executeQuery();

// 한 명 한 명의 데이터를 하나의 객체로 만든 뒤, 배열로 생성하여 받아온다.
ArrayList<StudentDTO> students = new ArrayList<StudentDTO>();

while(rset.next()) {
    String hakbun = rset.getString(1);
    String name = rset.getString(2);
    String dept = rset.getString(3);
    String addr = rset.getString(4);

    students.add(new StudentDTO(hakbun, name, dept, addr));
}

return students;
}

```

3. View JSP 작성

▼ INSERT 예시

• INSERT Form

```

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

```

• Insert JSP (insert 처리)

```

<%@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>

```

▼ SELECT 예시

- Student목록 JSP

```

<%@ page import="jdbc.*, java.util.ArrayList" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
    ArrayList<StudentDTO> students = StudentDAO.getList();

%>
<table border="1">
<tr>
<th>학번</th>
<th>이름</th>
</tr>
<%
    for (StudentDTO student : students){

%>
<tr>
<!-- 학번 클릭 시 detail 화면으로 이동하여 해당 학생에 대한 정보 전부 출력 -->
<td><a href="TBDetail.jsp?hakbun=<%=student.getHakbun() %>"><%=student.getHakbun() %></a></td>
<td><%=student.getName() %></td>

```

```

<!-- <td><%=student.getDept() %></td>
<td><%=student.getAddr() %></td> -->
</tr>
<%
}
%>
</table>
</body>
</html>

```

- Student detail JSP

```

<%@ page import="jdbc.*, java.util.ArrayList" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<style>
#detail {display:none};
</style>
</head>
<body>
<%
String hakbun = request.getParameter("hakbun");
StudentDTO student = StudentDAO.getDetail(hakbun);

%>
<table border="1">
<tr>
<th>학번</th>
<th>이름</th>
<th>전공</th>
<th>주소</th>
</tr>
<tr>
<td><%=student.getHakbun() %></td>
<td><%=student.getName() %></td>
<td><%=student.getDept() %></td>
<td><%=student.getAddr() %></td>
</tr>
</table>
</body>
</html>

```

▼ 실습

Board 테이블을 신규 생성하고, Connection Pool을 적용하여 DB연결 실습

1. DB 설계

- 테이블 명 : board

컬럼명	데이터 타입(크기)	설명	추가 설정
bno	INT(100)	글번호	Primary Key 기본값 : AUTO_INCREMENT >> 해당 컬럼에 자동으로 증가되는 번호를 생성
btitle	VARCHAR(100)	제목	
bwriter	VARCHAR(50)	작성자	
bcontent	VARCHAR(500)	내용	
bdate	TIMESTAMP	작성일	기본값 : CURRENT_TIMESTAMP() >> 테이블에 데이터가 입력될 때 자동으로 해당 시간을 입력함

2. Connection Pool 설정

Connection Pool 설정 이동

3. BoardDTO 작성

- DB데이터는 TIMESTAMP, INT 등 데이터 타입이 다르나, JAVA 안에서 처리는 문자열로 처리되므로 String 객체로 선언해준다.

```

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;
    }

}

```

4. BoardDAO 작성

```

package jdbc;

import java.sql.*;
import java.util.ArrayList;

import javax.naming.NamingException;
import util.ConnectionPool;

public class BoardDAO {
    // Insert 메서드
    public static int insert(String title, String writer, String content) throws NamingException, SQLException {

        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, content);

        int result = pstmt.executeUpdate();
        return result;
    }

    // Select All 메서드
    public static ArrayList<BoardDTO> getList() throws SQLException, NamingException{

        String sql = "select bno, btitle, bwriter, bcontent, DATE_FORMAT(bdate, '%Y-%m-%d %H:%i:%s') as bdate from board order by bno";

        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;
}
}

```

5. View 작성

- BoardInsert 처리 JSP

```

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

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

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

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

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

```

- BoardForm

```

<html>
<head>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/bootstrap.min.css" integrity="sha384-xOoLHfLEh0"
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<div class="container">
<form action="BoardInsert.jsp" method="POST">
<div class="form-group">
<label for="exampleFormControlInput1">Title</label>
<input name="title" type="text" class="form-control" id="exampleFormControlInput1">
</div>
<div class="form-group">
<label for="exampleFormControlTextarea1">Content</label>
<textarea name="content" class="form-control" id="exampleFormContr
olTextarea1" rows="3"></textarea>
</div>
<div class="text-right">
<button type="submit" class="btn btn-primary">글 작성</button>
</div>
</form>
</div>
</body>
</html>

```

- BoardList

```

<html>
<head>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kWBq7"
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<% ArrayList<BoardDTO> boardList = BoardDAO.getList(); %>
<div class="container">
<table class="table table-hover">
<thead>
<tr>

```

```

        <th scope="col">글번호</th>
        <th scope="col">제목</th>
        <th scope="col">작성자</th>
        <th scope="col">작성시간</th>
    </tr>

    <tbody>
    <%
        for(BoardDTO board : boardList){
    %>
        <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>

```

Summernote 적용

- link : <https://summernote.org/>
- 주의사항
 - summernote의 경우 code로 데이터를 전송하기 때문에 DB필드 사이즈를 크게 LONGTEXT로 지정해야 한다.
 - 헤더에 긴 데이터를 모두 담을 수 없기 때문에, 전송 방식을 "POST"로 설정해야만 한다.
- 모바일 화면 적용 코드

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

- 설정 : Without Bootstrap (lite) 코드에 bootstrap5를 따로 적용하여 설정한다.
- 설정 예시
 - summernote로 사용할 input 태그의 id에 summernote를 추가한다.

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>without bootstrap</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kWBq7"
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6qa4849b1E2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0"
    <link href="https://cdn.jsdelivr.net/npm/summernote@0.8.18/dist/summernote-lite.min.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/summernote@0.8.18/dist/summernote-lite.min.js"></script>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
</head>
<body>
    <div class="container">
    <form action="BoardInsert.jsp" method="POST">
    <div class="form-group">
        <label for="exampleFormControlInput1">Title</label>
        <input name="title" type="text" class="form-control" id="exampleFormControlInput1">
    </div>
    <div class="form-group">
        <label for="exampleFormControlTextarea1">Content</label>
        <textarea name="content" class="form-control" id="summernote" rows="3"></textarea>
        <!-- summernote를 사용하는 경우 code로 내용이 전달되기 때문에 저장되는 테이블 컬럼의 크기를 크게 잡아두어야 함 -->
        <!-- id값에 summernote를 추가 -->
    </div>
    <div class="text-right">
        <button type="submit" class="btn btn-primary">글 작성</button>
    </div>
    </form>
    </div>
    <script>
        $('#summernote').summernote({

```

```
placeholder: 'Hello stand alone ui',
tabsize: 2,
height: 120,
toolbar: [
  ['style', ['style']],
  ['font', ['bold', 'underline', 'clear']],
  ['color', ['color']],
  ['para', ['ul', 'ol', 'paragraph']],
  ['table', ['table']],
  ['insert', ['link', 'picture', 'video']],
  ['view', ['fullscreen', 'codeview', 'help']]
]
});
</script>
</body>
</html>
```