20230116

■ Hour 1

Connection Pool - Basic

JDBC를 사용할 때 가장 많이 리소스가 소모되는 부분은 Connection 객체 생성이다.

기존의 방법은 JSP에서 SQL구문을 수행하기 위해서 Connection 객체를 생성하고 사용 후 제거하는 과정을 반복해 접속자가 많아질 경우 시스템의 성능이 급격하게 저하된다.

따라서 이러한 문제점을 해결하기 위해 Connection Pool을 사용한다. 사용자가 접속할 때마다 새로운 Connection 객체를 생성하는 것이 아닌 일정 개수의 Connection 객체를 미리 생성해 요청이 있을 때마다 객체를 할당하고 다시 회수한다.

Connection Pool 설정

1. context.xml
2. ConnectionPoll.java
3. JDBC connector driver

context.xml

데이터 베이스에 대한 Connection Pool을 사용하기 위한 설정을 정의한다.

WebContent > META-INF > context.xml

```
<?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 않을 때도 있다
```

■ Hour 2

ConnectionPool.java

정의된 내용으로 실제 디비와 연결 해주는 객체를 생성하기 위핸 클래스 작성

src > util (package 생성) > ConnectionPool.java

```
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();
```

```
  }
}
```

JDBC connector driver

WebContent > WEB-INF > lib

Connection Pool 적용

항상 DB 설계 먼저

4. DTO (Data Transfer Object)
   - DB에서 데이터를 꺼낼 때 사용
   - DTO 파일은 DB의 필드와 일대일 매칭이 되게 설계
   - 테이블의 필드명으로 변수를 private으로 생성하고 getter, setter, constructor 생성

```
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;
  }


}
```

5. DAO (Data Acess Object)

- 실제 DB와 연결되는 메소드 등과 SQL 쿼리등을 작성

```
package jdbc;

import java.sql.*;

import javax.naming.NamingException;

import util.ConnectionPool;

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

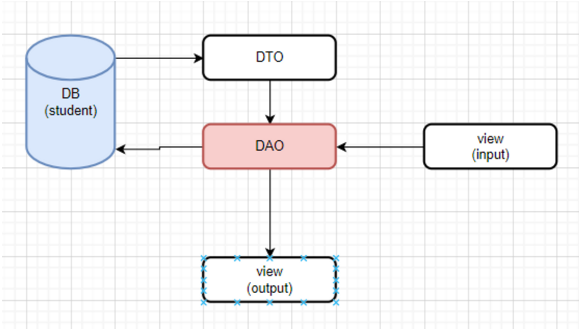
```
    // CRUD
    String sql = "INSERT INTO student VALUES(?,?,?,?)";

    Connection conn = ConnectionPool.get();   // 커넥션 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;
  }

}
```



■ Hour 3

6. 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">
  학번 <input type="text" name="hakbun"> <br>
  이름 <input type="text" name="name"> <br>
  전공 <input type="text" name="dept"> <br>
  주소 <input type="text" name="addr"> <br>
  <button type="submit">전송</button>
</form>
</body>
</html>
```

```
<%@ page import ="jdbc.*" %>
<%@ 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>

<%
  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("success");
  } else {
```

```
    out.print("fail");
  }

%>
</body>
</html>
```

■ Hour 4

Connection Pool 적용

- TBList.jsp
  - 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 링크를 걸어서 각 학생의 세부정보 표시

```
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;

  }
```

■ Hour 5

Connection Pool을 적용하는 게시판 테이블 (new DB) 생성

DB 설계

테이블 명 : board

글 번호 : bno 10

제목 : btitle 100

작성자 : bwriter 10

내용 : bcontent 500

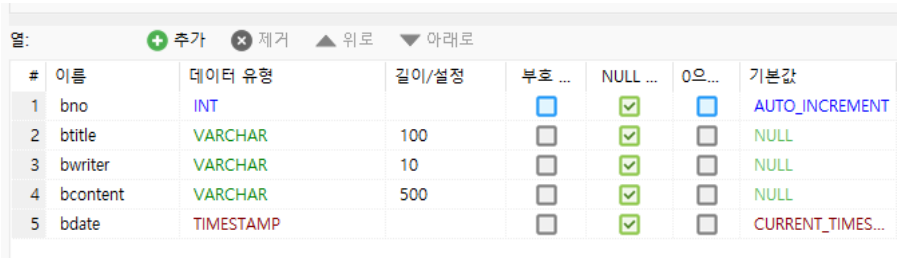
날짜 : bdate

테이블에 자동 증가 번호 넣기

- 데이터 유형 - INT
- 기본값 - AUTO_INCREMENT

테이블에 데이터가 입력될 때의 시간 입력

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



#	이름	데이터 유형	길이/설정	부호 ...	NULL ...	0으...	기본값
1	bno	INT		☐	☑	☐	AUTO_INCREMENT
2	btitle	VARCHAR	100	☐	☑	☐	NULL
3	bwriter	VARCHAR	10	☐	☑	☐	NULL
4	bcontent	VARCHAR	500	☐	☑	☐	NULL
5	bdate	TIMESTAMP		☐	☑	☐	CURRENT_TIMES...

■ Hour 6

- 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-GLhlTQ8

<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 (bootstrap 적용)

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

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>글 목록</title>
</head>
<body>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-GLhlTQ8
  <table class="table table-hover">
   <thead>
    <tr>
      <th scope="col">#</th>
      <th scope="col">title</th>
      <th scope="col">writer</th>
      <th scope="col">content</th>
    </tr>
```

```
  </thead>
  <tbody>
<%
  ArrayList<BoardDTO> boards = BoardDAO.getList();
  for(BoardDTO board : boards){
%>
    <tr>
      <td><%=board.getBno() %></td>
      <td><%=board.getBtitle() %></td>
      <td><%=board.getBwriter()%></td>
      <td><%=board.getBcontent() %></td>
    </tr>
<%  }%>
  </tbody>
</table>

</body>
</html>
```

■ Hour 7

Summernote 적용

- 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" />