Class 230116

JSP로 데이터베이스 사용 4가지 방법 (계속)

이전 학습 가기

4. Connection Pool 이용 (권장)

Connection Pool

사용자가 접속 할 때마다 매번 새로운 Connection객처를 생성하는 것이 아니라 일정 개수의 Connection객체를 미리 생성해 놓고 사용자의 요청이 있을 때마다 가용한 객체를 할당하고 다시 회수하는 방식

- JDBC사용 시 리소스(자원)를 가장 많이 소모하는 부분
  → Connection 객체를 생성하는 부분
  → 이전 방식들에서는 JSP에서 SQL구문을 수행하기 위해 Connection 객체를 생성하고 사용 후 제거하는 과정을 반복
  → 접속자가 많아질 경우 시스템의 성능을 급격하게 저하
  → 이러한 문제점을 해결하기 위한 방법으로 커넥션 풀을 이용

Connection Pool 설정 3단계

1. **Connection Pool 설정 정의 context.xml 작성**
   - 데이터베이스에 대한 커넥션 풀을 사용하기 위한 설정을 정의
   - 위치: WebContent > META-INF > context.xml

   ▼ context.xml 보기

```
<?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 업로드 시에는 변경 필요 )
  password="0000"        <- 디비 패스워드 ( 호스팅 업체에 업로드 시에는 변경 필요 )
  maxTotal="16"          <- 미리 생성할 Connection의 개수
  maxIdle="4"            <- 최저 유지 Connection의 개수
  maxWaitMillis="-1" />  <- 항상 -1, 기다리는 시간, -1은 기다리지 않고 바로 처리
</Context>
```

2. **ConnectionPool.java 클래스 작성**
   - 정의된 내용으로 실제 디비와 연결 해주는 객체를 생성하기 위핸 클래스 작성
   - 위치: Java Resources > src > util 패키지 > ConnectionPool.java

   ▼ 코드 보기

```
package util;

import java.sql.Connection;
import java.sql.SQLException;

import javax.naming.InitialContext;
import javax.naming.NamingException;
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 : DB 이름
    return _ds.getConnection();
  }
}
```

3. **JDBC connector driver 가져오기**
   - mysql-connector jar파일(드라이버) 위치 시키기
   - 위치: WebContent > META-INF > lib

Connection Pool 적용

DAO 를 통해 커넥션 풀에서 커넥션 객체를 가져와 사용

용어 정리

	DTO	VO
목적	계층간 데이터 전달	값 자체 표현
동등성	필드값이 같아도 같은 객체가 아님	필드값이 같으면 같은 객체
가변성	setter 존재시 가변 setter 미존재시 불가변	불변
로직	getter/setter 외의 로직이 필요하지 않음	getter/setter 외의 로직이 있어도 무방

1. DTO (Data Transfer Object)
   - 계층간의 데이터를 전송하기 위한 객체
   - 사실 DTO 는 디비에서 데이터를 꺼낼때만 사용. DTO 파일은 데이터베이스의 테이블의 필드와 일대일 매칭이 되게 설계

   ▼ 코드 보기

```
/* studentDTO.java */
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)
   - DB에 접근할 때 사용하는 객체

   ▼ 코드 보기

```
/* studentDAO.java */
package jdbc;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import javax.naming.NamingException;

import util.ConnectionPool;

public class StudentDAO {

  //테이블에 데이터를 입력
  public static int insert(String hakbun, String name, String dept, String addr) throws NamingException, SQLException {
    int result = 0;
    // 커넥션 풀 사용
    try {
      Connection conn = ConnectionPool.get();

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

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

      result = pstmt.executeUpdate();

    } catch (NamingException | SQLException e) {
      e.printStackTrace();
    }

    return result;
  }

  // 테이블 목록을 가져옴
  public static List<StudentDTO> getList() throws NamingException, SQLException {
    ArrayList<StudentDTO> list = new ArrayList<>();
    try {
      Connection conn = ConnectionPool.get();

      String sql = "SELECT * FROM student";

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

      while (rSet.next()) {
        StudentDTO sDto = new StudentDTO(
          rSet.getString(1),
          rSet.getString(2),
          rSet.getString(3),
          rSet.getString(4)
        );
        list.add(sDto);
      }

    } catch (NamingException | SQLException e) {
      e.printStackTrace();
    }
    return list;
  }
```

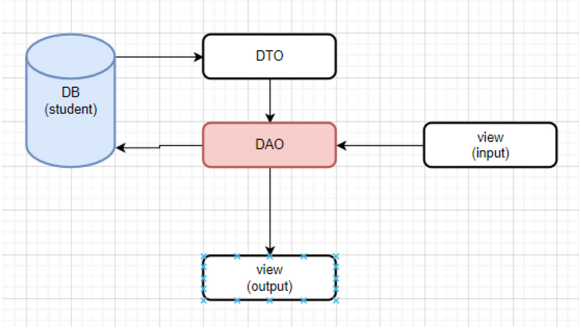
```
  // 학생 정보 하나를 가져옴
  public static StudentDTO getStudent(String hakbun) throws NamingException, SQLException {
    StudentDTO sDto = null;
    try {
      Connection conn = ConnectionPool.get();

      String sql = "SELECT * FROM student WHERE hakbun=?";

      PreparedStatement pstmt = conn.prepareStatement(sql);
      pstmt.setString(1, hakbun);
      ResultSet rSet = pstmt.executeQuery();

      while (rSet.next()) {
        sDto = new StudentDTO(
          rSet.getString(1),
          rSet.getString(2),
          rSet.getString(3),
          rSet.getString(4)
        );
      }

    } catch (NamingException | SQLException e) {
      e.printStackTrace();
    }
    return sDto;
  }
}
```



Connection Pool을 적용한 게시판

Database 테이블 작성

- 글 번호 (bno)
- 작성자 (bwriter)
- 내용 (bcontent)
- 작성일 (bdate)
  - TIMESTEMP에 기본값 CURRENT_TIMESTAMP으로 설정

▼ 코드 보기

```
CREATE TABLE `board` (
  `bno` INT(10) NOT NULL AUTO_INCREMENT,
  `btitle` VARCHAR(100) NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
  `bwriter` VARCHAR(10) NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
  `bcontent` LONGTEXT NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
  `bdate` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`bno`) USING BTREE
)
COLLATE='utf8mb4_0900_ai_ci'
ENGINE=InnoDB
AUTO_INCREMENT=1
;
```

BoardDTO 작성

```
package jdbc;

public class BoardDTO {
  private int bno;
  private String btitle;
  private String bwriter;
  private String bcontent;
  private String bdate;

  // getter/setter/constructor 생략
}
```

BoardDAO 작성

▼ 전체 코드 보기

```
package jdbc;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import util.ConnectionPool;

public class BoardDAO {

  // 작성된 게시글 등록
  public static boolean insert(String btitle, String bwriter, String bcontent) {
    boolean result = false;
    String sql = "INSERT INTO board (btitle, bwriter, bcontent) VALUES (?, ?, ?)";

    try {
      Connection conn = ConnectionPool.get();
      PreparedStatement pstmt = conn.prepareStatement(sql);
      pstmt.setString(1, btitle);
      pstmt.setString(2, bwriter);
      pstmt.setString(3, bcontent);

      result = pstmt.executeUpdate() == 1 ? true : false;

    } catch (SQLException e) {
      e.printStackTrace();
    }

    return result;
  }

  // 게시글 목록 가져오기
  public static List<BoardDTO> selectList() {
    ArrayList<BoardDTO> list = new ArrayList<>();
    String sql = "SELECT * FROM board";

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

      ResultSet rSet = pstmt.executeQuery();

      while (rSet.next()) {
        BoardDTO bdto = new BoardDTO(
          rSet.getInt("bno"),
          rSet.getString("btitle"),
          rSet.getString("bwriter"),
          rSet.getString("bcontent"),
          rSet.getString("bdate")
        );
```

```
        list.add(bdto);
      }

    } catch (SQLException e) {
      e.printStackTrace();
      list = null;
    }

    return list;
  }

  // 게시글 목록 가져오기
  public static BoardDTO select(int bno) {
    BoardDTO bdto= null;
    String sql = "SELECT * FROM board WHERE bno=?";

    try {
      Connection conn = ConnectionPool.get();
      PreparedStatement pstmt = conn.prepareStatement(sql);
      pstmt.setInt(1, bno);
      ResultSet rSet = pstmt.executeQuery();

      while (rSet.next()) {
        bdto = new BoardDTO(
          rSet.getInt("bno"),
          rSet.getString("btitle"),
          rSet.getString("bwriter"),
          rSet.getString("bcontent"),
          rSet.getString("bdate")
        );
      }

    } catch (SQLException e) {
      e.printStackTrace();
    }

    return bdto;
  }
}
```

View (JSP) 작성

▼ BoardInsert.jsp

등록처리 후 메시지만 브라우저에 출력함

```
<%@page import="jdbc.BoardDAO"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
  // 전송 받는 데이터 한글 처리
  request.setCharacterEncoding("UTF-8");

  if (BoardDAO.insert(request.getParameter("btitle"), "작성자", request.getParameter("bcontent"))) {
    out.print("등록 성공");
  } else {
    out.print("등록 실패");
  }
%>
```

▼ BoardForm.jsp

```

```

▼ BoardList.jsp

```
...
<%
  List<BoardDTO> bList = BoardDAO.selectList();
%>
...
<div class="table-responsive">
  <table class="table table-hover">
      <thead>
          <tr>
```

```
            <th>글번호</th>
            <th>제목</th>
            <th>작성자</th>
        </tr>
      </thead>
      <tbody>
<%
        for (BoardDTO bdto : bList) {
%>
        <tr onclick="location.href = 'BoardDetail.jsp?bno=<%= bdto.getBno() %>'">
            <td><%= bdto.getBno() %></td>
            <td><%= bdto.getBtitle() %></td>
            <td><%= bdto.getBcontent() %></td>
        </tr>
<%
        }
%>
      </tbody>
  </table>
  </div>
  <!-- /.table-responsive -->
```

▼ BoardDetail.jsp

```
...
<%
  BoardDTO bdto = BoardDAO.select(Integer.parseInt(request.getParameter("bno")));
%>
...
<form role="form">
  <div class="form-group">
        <label>No. : <%= bdto.getBno() %></label>
        <label class="pull-right">Date : <%= bdto.getBdate() %></label>
    </div>
    <div class="form-group">
        <label>Title</label>
        <input class="form-control" name="btitle" value="<%= bdto.getBtitle() %>" readonly>
    </div>
    <div class="form-group">
        <label>Content</label>
        <input class="form-control" name="bcontent" value="<%= bdto.getBcontent() %>" readonly>
    </div>
    <div class="form-group">
        <label>Writer</label>
        <input class="form-control" name="bwriter" value="<%= bdto.getBwriter() %>" readonly>
    </div>
</form>
```

Summernote 사용

SummerNote는 초간단 WYSIWYG Editor임. 반응형 웹 (모바일) 지원.

> 💡 **WYSIWYG(what you see is what you get)**
> 편집화면에서 입력한 글자, 그림 등의 컨텐츠 모양 그대로 최종산물이 화면상에서, 또는 출력물에서 나타나도록 하는 에디터

부트스트랩 없이 사용 (부트스트랩5 추가 사용)

```
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>without bootstrap</title>
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6qa4849blE2+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>
  </head>
  <body>
    <div id="summernote"></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>
```

→ 여기에 부트스트랩 5를 추가해 사용