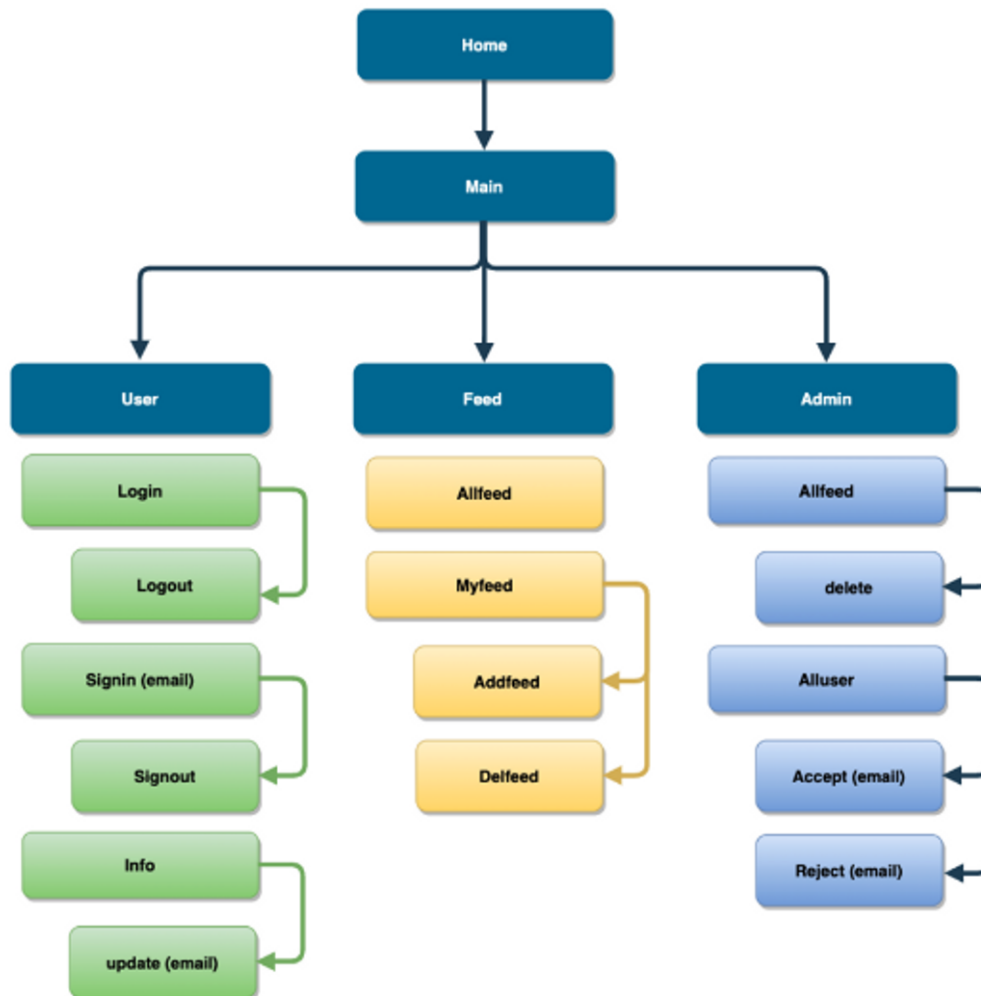
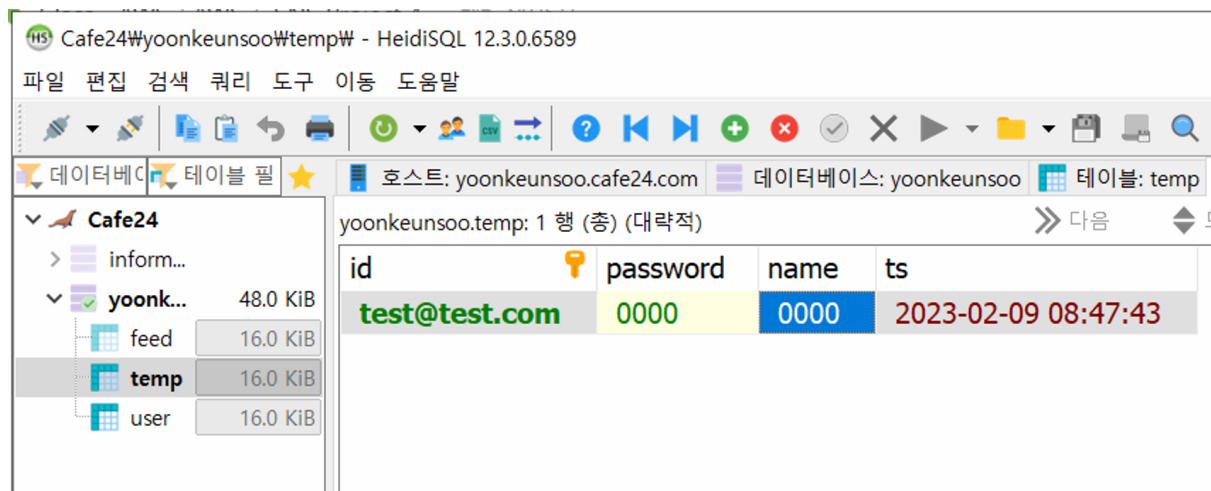


20230209 SNS Project4

IA Information Architecture





▼ userDAO

```
package jdbc;

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

import javax.naming.NamingException;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;

import util.ConnectionPool;

public class UserDAO {

    //회원 가입
    public static int insert(String id, String password, String name) throws NamingException, SQLException {

        String sql = "INSERT INTO user(id, password, name) VALUES(?, ?, ?)";

        Connection conn = null;
        PreparedStatement pstmt = null;

        try {
            conn = ConnectionPool.get();
            pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, id);
            pstmt.setString(2, password);
            pstmt.setString(3, name);

            return pstmt.executeUpdate();
        } finally {
```

```

        if(pstmt!= null) pstmt.close();
        if(conn != null) conn.close();

    }
}

//회원 가입 신청(승인 /거부 )
public static int inserttemp(String id, String password, String name) throws NamingException, SQLException {

    String sql = "INSERT INTO temp(id, password, name) VALUES(?, ?, ?)";

    Connection conn = null;
    PreparedStatement pstmt = null;

    try {
        conn = ConnectionPool.get();
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, id);
        pstmt.setString(2, password);
        pstmt.setString(3, name);

        return pstmt.executeUpdate();
    } finally {
        if(pstmt!= null) pstmt.close();
        if(conn != null) conn.close();
    }
}

//관리자가 임시 테이블 자료를 유저 테이블로 이동시키는 가입 승인 절차
public static int insertAdmin(String id) throws NamingException, SQLException {

    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {

        //temp에서 id 값으로 데이터 조회 해오기
        String sql1 = "SELECT * FROM temp WHERE id=?";

        conn = ConnectionPool.get();
        pstmt = conn.prepareStatement(sql1);
        pstmt.setString(1, id);

        rs = pstmt.executeQuery();
        rs.next();

        String tid = rs.getString(1);
        String tpassword = rs.getString(2);
        String tname = rs.getString(3);

        // 조회한 데이터를 user 테이블에 넣기
        String sql2 = "INSERT INTO user(id, password, name) VALUES(?, ?, ?)";

        pstmt = conn.prepareStatement(sql2);
    }
}

```

```

        pstmt.setString(1, tid);
        pstmt.setString(2, tpassword);
        pstmt.setString(3, tname);

        pstmt.executeUpdate();

        // temp에서 해당 데이터 삭제

        String sql3 = "DELETE FROM temp WHERE id=?";

        pstmt = conn.prepareStatement(sql3);
        pstmt.setString(1, id);

        return pstmt.executeUpdate(); //성공 1, 실패 0
    } finally {
        if(rs != null) rs.close();
        if(pstmt != null) pstmt.close();
        if(conn != null) conn.close();
    }
}

// 회원가입시 아이디가 이미 존재하는지 여부 확인
public static boolean exist(String id) throws NamingException, SQLException {

    String sql = "SELECT id FROM user WHERE id=?";

    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {
        conn = ConnectionPool.get();
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, id);

        rs = pstmt.executeQuery();

        return rs.next(); // 조회한 아이디가 DB에 존재하면 true, 없으면 false
    } finally {
        if(rs != null) rs.close();
        if(pstmt != null) pstmt.close();
        if(conn != null) conn.close();
    }

}

//회원 탈퇴
public static int delete(String id) throws NamingException, SQLException {

    String sql = "DELETE FROM user WHERE id=?";
    Connection conn = null;
    PreparedStatement pstmt = null;

    try {

```

```

        conn = ConnectionPool.get();
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, id);

        return pstmt.executeUpdate(); //성공 1, 실패 0
    } finally {
        if(pstmt != null) pstmt.close();
        if(conn != null) conn.close();
    }
}

//로그인
public static int login(String id, String upassword) throws NamingException, SQLException {

    String sql = "SELECT id, password FROM user WHERE id=?";
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {
        conn = ConnectionPool.get();
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, id);
        rs = pstmt.executeQuery();

        if(!rs.next()) return 1; // 아이디가 존재하지 않는 경우
        if(!upassword.equals(rs.getString("password"))) return 2; //아이디는 존재하지만
비번이 일치하지 않는 경우
        return 0; //로그인 성공

    } finally {
        if(rs != null) rs.close();
        if(pstmt != null) pstmt.close();
        if(conn != null) conn.close();
    }
}

//회원 목록
public static ArrayList<UserDTO> list() throws NamingException, SQLException {
    String sql = "SELECT * FROM user ORDER BY ts DESC";

    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

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

        rs = pstmt.executeQuery();

        ArrayList<UserDTO> users = new ArrayList<UserDTO>();

        while(rs.next()) {
            users.add(new UserDTO(rs.getNString(1),
                rs.getString(2),
                rs.getString(3),

```

```

        rs.getString(4)));
    }
    return users;

} finally {
    if(rs != null) rs.close();
    if(pstmt != null) pstmt.close();
    if(conn != null) conn.close();
}

}

//임시 테이블 회원 목록
public static String getTemp() throws NamingException, SQLException {
    String sql = "SELECT * FROM temp ORDER BY ts DESC";

    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

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

        rs = pstmt.executeQuery();

        JSONArray users = new JSONArray();

        while(rs.next()) {
            JSONObject obj = new JSONObject();
            obj.put("id", rs.getString(1));
            obj.put("password", rs.getString(2));
            obj.put("name", rs.getString(3));
            obj.put("ts", rs.getString(4));

            users.add(obj);
        }
        return users.toJSONString();

    } finally {
        if(rs != null) rs.close();
        if(pstmt != null) pstmt.close();
        if(conn != null) conn.close();
    }

}

//회원 수정
public static int edit(String id, String password, String name) throws NamingException, SQLException {

    String sql = "UPDATE user SET password=?, name=? WHERE id=? ";

    Connection conn = null;
    PreparedStatement pstmt = null;

```

```

        try {
            conn = ConnectionPool.get();
            pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, password);
            pstmt.setString(2, name);
            pstmt.setString(3, id);

            return pstmt.executeUpdate();
        } finally {
            if(pstmt!= null) pstmt.close();
            if(conn != null) conn.close();
        }
    }
}

} // END

```

▼ temp.jsp

```

<%@page import="jdbc.*"%>
<%@page import="java.util.ArrayList"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@page errorPage = "page_error_page.jsp" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Group Talk</title>
</head>
<body>
<%@include file="/header.jsp" %>

<%
    sid = (String) session.getAttribute("id");
    if (!sid.equals("admin")){
%>
        <!-- Modal -->
        <div class="modal fade" id="exampleModal2" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1" aria-labelledby="staticBackdropLabel" aria-hidden="true">
            <div class="modal-dialog">
                <div class="modal-content">
                    <div class="modal-header">
                        <h1 class="modal-title fs-5" id="exampleModalLabel">회원 전용 메뉴</h1>
                        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
                    </div>
                    <div class="modal-body">
                        관리자만 사용 가능한 메뉴입니다.<br>
                        로그인 화면으로 이동합니다.
                    </div>
                    <div class="modal-footer">

```



```

        <button onclick="location.href='/user/login.jsp'" class="btn btn-primary">로그인 이동</button>

    </div>
</div>
</div>
</div>

<script>
    $(function() {
        $("#exampleModal2").modal("show");
    });
</script>

<%} else {
    session.setAttribute("id", sid);
}
%>

<style>
@font-face {
    font-family: 'KorailRoundGothicBold';
    src: url('https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_2212@1.0/KorailRoundGothicBold.woff2') format('woff2');
    font-weight: 700;
    font-style: normal;
}

body {
    display: flex;
    align-items: center;
    padding-top: 40px;
    padding-bottom: 40px;
    background-color: #f5f5f5;
    font-family: 'KorailRoundGothicBold';
}
</style>

<div class="container bg-warning shadow mx-auto p-5 w-75">
<h2>가입 신청자 목록</h2>
<hr>
<div class="input-group justify-content-center">
</div>
<hr>

<table align=center width="400">
<thead>
</thead>
<tbody id="ajaxTable">
</tbody>

```

```

</table>
</div>

<script>
function addItem(id) {
    $.ajax({
        type:"post",
        url: "userAdd.jsp",
        data : {id:id

    },
    dataType:"text",

    success:function(data) {
        alert('관리자 회원 등록 성공');
        searchFunction(); //등록하면 화면에 바로 나올수 있도록 호출.

    }
    });
}

function delItem(id) {
    $.ajax({
        type:"post",
        url: "userAdd.jsp",
        data : {id:id
    },
    dataType:"text",

    success:function(data) {
        alert('관리자 회원 삭제');
        searchFunction();
    }
    });
}

function searchFunction(){
    $.ajax({
        type:"post",
        url:"/admin/tempAll.jsp",
        success:function(data){
            var users = JSON.parse(data.trim());
            var str="";
            for(var i=0; i < users.length; i++){
                str += "<tr><td><small>" + users[i].id + "</small></td>";
                str += "<td><small>&nbsp;" + users[i].name + "</small></td>";
                str += "<td><div onclick='delItem(\"" + users[i].id + "\")'><span class";
                str += "<td><div onclick='addItem(\"" + users[i].id + "\")'><span class";
                str += "</tr>";
                str += "<tr><td colspan=4 height=40><hr></td></tr>"
            }
            $("#ajaxTable").html(str);
        }
    });
}

```

```

    });
}

window.onload = function(){
    searchFunction();
}

</script>
<%@include file="/footer.jsp" %>
</body>
</html>

```

눈누 폰트중에 패스워드 입력시 ****으로 나타나오는게 있음..



```

input[type="password"] {
    font-family: sans-serif;
}

```

- CSS 안에 넣어주기 ..

▼ userDAO

```

package jdbc;

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

import javax.naming.NamingException;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;

import util.ConnectionPool;

public class UserDAO {

    //회원 가입
    public static int insert(String id, String password, String name) throws NamingException, SQLException {

```

```

String sql = "INSERT INTO user(id, password, name) VALUES(?, ?, ?)";

Connection conn = null;
PreparedStatement pstmt = null;

try {
    conn = ConnectionPool.get();
    pstmt = conn.prepareStatement(sql);
    pstmt.setString(1, id);
    pstmt.setString(2, password);
    pstmt.setString(3, name);

    return pstmt.executeUpdate();
} finally {
    if(pstmt != null) pstmt.close();
    if(conn != null) conn.close();
}
}

//회원 가입 신청(승인 /거부 )
public static int insertTemp(String id, String password, String name) throws NamingException, SQLException {

    String sql = "INSERT INTO temp(id, password, name) VALUES(?, ?, ?)";

    Connection conn = null;
    PreparedStatement pstmt = null;

    try {
        conn = ConnectionPool.get();
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, id);
        pstmt.setString(2, password);
        pstmt.setString(3, name);

        return pstmt.executeUpdate();
    } finally {
        if(pstmt != null) pstmt.close();
        if(conn != null) conn.close();
    }
}

//관리자가 임시 테이블 자료를 유저 테이블로 이동시키는 가입 승인 절차
public static int insertAdmin(String id) throws NamingException, SQLException {

    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {

        //temp에서 id 값으로 데이터 조회 해오기
        String sql1 = "SELECT * FROM temp WHERE id=?";

        conn = ConnectionPool.get();

```

```

        pstmt = conn.prepareStatement(sql1);
        pstmt.setString(1, id);

        rs = pstmt.executeQuery();
        rs.next();

        String tid = rs.getString(1);
        String tpassword = rs.getString(2);
        String tname = rs.getString(3);

        // 조회한 데이터를 user 테이블에 넣기
        String sql2 = "INSERT INTO user(id, password, name) VALUES(?, ?, ?)";

        pstmt = conn.prepareStatement(sql2);
        pstmt.setString(1, tid);
        pstmt.setString(2, tpassword);
        pstmt.setString(3, tname);

        pstmt.executeUpdate();

        // temp에서 해당 데이터 삭제

        String sql3 = "DELETE FROM temp WHERE id=?";

        pstmt = conn.prepareStatement(sql3);
        pstmt.setString(1, id);

        return pstmt.executeUpdate(); //성공 1, 실패 0
    } finally {
        if(rs != null) rs.close();
        if(pstmt != null) pstmt.close();
        if(conn != null) conn.close();
    }
}

// 회원가입시 아이디가 이미 존재하는지 여부 확인
public static boolean exist(String id) throws NamingException, SQLException {

    String sql = "SELECT id FROM user WHERE id=?";

    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {
        conn = ConnectionPool.get();
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, id);

        rs = pstmt.executeQuery();

        return rs.next(); // 조회한 아이디가 DB에 존재하면 true, 없으면 false
    } finally {
        if(rs != null) rs.close();
        if(pstmt != null) pstmt.close();
    }
}

```

```

        if(conn != null) conn.close();
    }

}

//회원 탈퇴
public static int delete(String id) throws NamingException, SQLException {

    String sql = "DELETE FROM user WHERE id=?";
    Connection conn = null;
    PreparedStatement pstmt = null;

    try {
        conn = ConnectionPool.get();
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, id);

        return pstmt.executeUpdate(); //성공 1, 실패 0
    } finally {
        if(pstmt != null) pstmt.close();
        if(conn != null) conn.close();
    }
}

//관리자 승인 거부
public static int deleteAdmin(String id) throws NamingException, SQLException
{

    String sql = "DELETE FROM temp WHERE id=?";
    Connection conn = null;
    PreparedStatement pstmt = null;

    try {
        conn = ConnectionPool.get();
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, id);

        return pstmt.executeUpdate(); //성공 1, 실패 0
    } finally {
        if(pstmt != null) pstmt.close();
        if(conn != null) conn.close();
    }
}

//로그인
public static int login(String id, String upassword) throws NamingException, SQLException {

    String sql = "SELECT id, password FROM user WHERE id=?";
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {
        conn = ConnectionPool.get();
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, id);

```

```

        rs = pstmt.executeQuery();

        if(!rs.next()) return 1; // 아이디가 존재하지 않는 경우
        if(!upassword.equals(rs.getString("password"))) return 2; //아이디는 존재하지만
비번이 일치하지 않는 경우
        return 0; //로그인 성공

    } finally {
        if(rs != null) rs.close();
        if(pstmt!= null) pstmt.close();
        if(conn != null) conn.close();
    }
}

//회원 목록
public static ArrayList<UserDTO> list() throws NamingException, SQLException {
    String sql = "SELECT * FROM user ORDER BY ts DESC";

    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

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

        rs = pstmt.executeQuery();

        ArrayList<UserDTO> users = new ArrayList<UserDTO>();

        while(rs.next()) {
            users.add(new UserDTO(rs.getNString(1),
                rs.getString(2),
                rs.getString(3),
                rs.getString(4)));
        }
        return users;

    } finally {
        if(rs != null) rs.close();
        if(pstmt!= null) pstmt.close();
        if(conn != null) conn.close();
    }
}

//임시 테이블 회원 목록
public static String getTemp() throws NamingException, SQLException {
    String sql = "SELECT * FROM temp ORDER BY ts DESC";

    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

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

```

```

        rs = pstmt.executeQuery();

        JSONArray users = new JSONArray();

        while(rs.next()) {
            JSONObject obj = new JSONObject();
            obj.put("id", rs.getString(1));
            obj.put("password", rs.getString(2));
            obj.put("name", rs.getString(3));
            obj.put("ts", rs.getString(4));

            users.add(obj);
        }
        return users.toJSONString();

    } finally {
        if(rs != null) rs.close();
        if(pstmt != null) pstmt.close();
        if(conn != null) conn.close();
    }
}

//회원 수정
public static int edit(String id, String password, String name) throws NamingException, SQLException {

    String sql = "UPDATE user SET password=?, name=? WHERE id=? ";

    Connection conn = null;
    PreparedStatement pstmt = null;


    try {
        conn = ConnectionPool.get();
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, password);
        pstmt.setString(2, name);
        pstmt.setString(3, id);

        return pstmt.executeUpdate();
    } finally {
        if(pstmt != null) pstmt.close();
        if(conn != null) conn.close();
    }
}

} // END

```


Google Colaboratory

 https://colab.research.google.com/drive/1XIGRzsiHgytYkr-5NsOFUg_jv8xhTiYa?usp=sharing

