

Class 230209

SNS Project (계속)

관리자 가입 승인 처리

▼ TempDAO

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

import util.ConnectionPool;

public class TempDAO {
    // 회원 가입
    public static boolean insert(String id, String password, String name) {
        boolean result = false;
        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);

            result = pstmt.executeUpdate() == 1 ? true : false;
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            close(conn, pstmt, null);
        }
        return result;
    }

    // 회원 정보 수정
    public static boolean edit(String id, String password, String name) {
        boolean result = false;
        String sql = "UPDATE temp 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);

            result = pstmt.executeUpdate() == 1 ? true : false;
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            close(conn, pstmt, null);
        }
        return result;
    }

    // 회원 정보 가져오기
    public static UserDTO getUser(String id) {
        UserDTO result = null;
        String sql = "SELECT * FROM temp WHERE id=?";
        Connection conn = null;
        PreparedStatement pstmt = null;
        ResultSet rSet = null;
        try {
            conn = ConnectionPool.get();
```

```

        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, id);
        rSet = pstmt.executeQuery();
        if (rSet.next()) {
            result = new UserDTO(
                rSet.getString("id"),
                rSet.getString("password"),
                rSet.getString("name"),
                rSet.getString("ts")
            );
        }
    } catch (SQLException e) {
        e.printStackTrace();
        result = null;
    } finally {
        close(conn, pstmt, rSet);
    }
    return result;
}

// 회원 목록 가져오기
public static List<UserDTO> selectList() {
    ArrayList<UserDTO> list = new ArrayList<>();
    String sql = "SELECT * FROM temp ORDER BY ts DESC";
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rSet = null;
    try {
        conn = ConnectionPool.get();
        pstmt = conn.prepareStatement(sql);
        rSet = pstmt.executeQuery();

        while (rSet.next()) {
            UserDTO udto = new UserDTO(
                rSet.getString("id"),
                rSet.getString("password"),
                rSet.getString("name"),
                rSet.getString("ts")
            );
            list.add(udto);
        }

    } catch (SQLException e) {
        e.printStackTrace();
        list = null;
    } finally {
        close(conn, pstmt, rSet);
    }

    return list;
}

// 회원 목록 가져오기 (AJAX)
public static String selectListAJAX() {
    JSONArray ja = new JSONArray();
    String sql = "SELECT * FROM temp ORDER BY ts DESC";
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rSet = null;
    try {
        conn = ConnectionPool.get();
        pstmt = conn.prepareStatement(sql);
        rSet = pstmt.executeQuery();

        while (rSet.next()) {
            JSONObject jo = new JSONObject();
            jo.put("id", rSet.getString("id"));
            jo.put("password", rSet.getString("password"));
            jo.put("name", rSet.getString("name"));
            jo.put("ts", rSet.getString("ts"));

            ja.add(jo);
        }

    } catch (SQLException e) {
        e.printStackTrace();
        ja = null;
    } finally {
        close(conn, pstmt, rSet);
    }

    return ja.toJSONString();
}

// 회원 확인
public static boolean exist(String id) {
    boolean result = false;

```

```

String sql = "SELECT * FROM temp WHERE id=?";
Connection conn = null;
PreparedStatement pstmt = null;
ResultSet rSet = null;
try {
    conn = ConnectionPool.get();
    pstmt = conn.prepareStatement(sql);
    pstmt.setString(1, id);
    rSet = pstmt.executeQuery();

    result = rSet.next() ? true : false;
} catch (SQLException e) {
    e.printStackTrace();
} finally {
    close(conn, pstmt, rSet);
}
return result;
}

// 게시물 삭제
public static boolean delete(String id) {
    boolean result = false;
    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);

        result = pstmt.executeUpdate() == 1 ? true : false;
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        close(conn, pstmt, null);
    }
    return result;
}

// 객체 닫기
public static void close(Connection conn, PreparedStatement pstmt, ResultSet rSet) {
    try {
        if(rSet!=null) rSet.close();
        if(pstmt!=null) pstmt.close();
        if(conn!=null) conn.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}

```

▼ UserDao

```

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

import util.ConnectionPool;

public class UserDao {
    // 회원 가입
    public static boolean insert(String id, String password, String name) {
        boolean result = false;
        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);

            result = pstmt.executeUpdate() == 1 ? true : false;
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

```

    } finally {
        close(conn, pstmt, null);
    }
    return result;
}

// 회원 가입 승인 처리
public static boolean insertAdmin(String id) {
    boolean result = false;

    String sql = "SELECT * FROM temp WHERE id=?";

    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rSet = null;
    try {
        conn = ConnectionPool.get();
        conn.setAutoCommit(false);

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

        rSet = pstmt.executeQuery();

        if (rSet.next())
            && UserDAO.insert(rSet.getString("id"), rSet.getString("password"), rSet.getString("name"))
            && TempDAO.delete(id) {
                conn.commit();
                result = true;
            } else {
                conn.rollback();
            }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        close(conn, pstmt, null);
    }
    return result;
}

// 회원 정보 수정
public static boolean edit(String id, String password, String name) {
    boolean result = false;
    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);

        result = pstmt.executeUpdate() == 1 ? true : false;
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        close(conn, pstmt, null);
    }
    return result;
}

// 회원 목록 가져오기
public static List<UserDTO> selectList() {
    ArrayList<UserDTO> list = new ArrayList<>();
    String sql = "SELECT * FROM user ORDER BY ts DESC";
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rSet = null;
    try {
        conn = ConnectionPool.get();
        pstmt = conn.prepareStatement(sql);
        rSet = pstmt.executeQuery();

        while (rSet.next()) {
            UserDTO udto = new UserDTO(
                rSet.getString("id"),
                rSet.getString("password"),
                rSet.getString("name"),
                rSet.getString("ts")
            );
            list.add(udto);
        }
    } catch (SQLException e) {
        e.printStackTrace();
        list = null;
    }
}

```

```

    } finally {
        close(conn, pstmt, rSet);
    }

    return list;
}

// 회원 목록 가져오기 (AJAX)
public static String selectListAJAX() {
    JSONArray ja = new JSONArray();
    String sql = "SELECT * FROM user ORDER BY ts DESC";
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rSet = null;
    try {
        conn = ConnectionPool.get();
        pstmt = conn.prepareStatement(sql);
        rSet = pstmt.executeQuery();

        while (rSet.next()) {
            JSONObject jo = new JSONObject();
            jo.put("id", rSet.getString("id"));
            jo.put("password", rSet.getString("password"));
            jo.put("name", rSet.getString("name"));
            jo.put("ts", rSet.getString("ts"));

            ja.add(jo);
        }

    } catch (SQLException e) {
        e.printStackTrace();
        ja = null;
    } finally {
        close(conn, pstmt, rSet);
    }

    return ja.toJSONString();
}

// 회원 확인
public static boolean exist(String id) {
    boolean result = false;
    String sql = "SELECT * FROM user WHERE id=?";
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rSet = null;
    try {
        conn = ConnectionPool.get();
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, id);
        rSet = pstmt.executeQuery();

        result = rSet.next() ? true : false;
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        close(conn, pstmt, rSet);
    }
    return result;
}

// 회원 탈퇴
public static boolean delete(String id) {
    boolean result = false;
    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);

        result = pstmt.executeUpdate() == 1 ? true : false;
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        close(conn, pstmt, null);
    }
    return result;
}

// 회원 로그인
public static int login(String id, String password) {
    int result = 0;
    String sql = "SELECT id, password FROM user WHERE id=?";
    Connection conn = null;
    PreparedStatement pstmt = null;

```

```

ResultSet rSet = null;
try {
    conn = ConnectionPool.get();
    pstmt = conn.prepareStatement(sql);
    pstmt.setString(1, id);
    rSet = pstmt.executeQuery();
    if (rSet.next()) { // id가 일치하고
        result = rSet.getString("password").equals(password)
            ? 1 // password가 일치 할 경우
            : 2; // password가 일치 하지 않을 경우
    }
} catch (SQLException e) {
    e.printStackTrace();
} finally {
    close(conn, pstmt, rSet);
}
return result;
}

// 객체 닫기
public static void close(Connection conn, PreparedStatement pstmt, ResultSet rSet) {
    try {
        if(rSet!=null) rSet.close();
        if(pstmt!=null) pstmt.close();
        if(conn!=null) conn.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}
}

```

▼ tempAll

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>GTalk</title>
</head>
<body>
<%@ include file="/include/header.jsp" %>
<%
    if (sid != null && sid.equals("admin")) {
%>
<div class="container bg-light shadow mx-auto p-5 w-75">
<h3>가입 신청 목록</h3>
<table class="table table-hover table-striped">
<thead>
<tr>
<th scope="col">ID</th>
<th scope="col">이름</th>
<th class="text-end" scope="col">신청일</th>
<th class="text-end" scope="col">승인 여부</th>
</tr>
</thead>
<tbody id="dataTable">
</tbody>
</table>
</div>
<%
    }
%>
<%@ include file="/include/footer.jsp" %>
<script type="text/javascript">

getList();
function getList() {
    fetch('/admin/tempCheck.jsp')
    .then(resp => resp.json())
    .then(data => {
        console.log(data);
        for ( var i = 0; i < data.length; i++) {
            $('#dataTable').append('<tr>'
                + '<td>'+data[i].id+'</td>'
                + '<td>'+data[i].name+'</td>'
                + '<td class="text-end">'+data[i].ts+'</td>'
                + '<td class="text-end">'
                + '<button type="button" class="btn btn-sm btn-outline-success mx-1" onclick="addTemp(\''+data[i].id+'\')">승인</button>'
                + '<button type="button" class="btn btn-sm btn-outline-danger mx-1" onclick="deleteTemp(\''+data[i].id+'\')">거부</button>'
                + '</td>'
                + '</tr>'
            );
        }
    });
}
    
```

```

    });
  }
})
.catch(err => console.log('Error : ', err));
}

function addTemp(id) {
  fetch('/admin/tempAdd.jsp', {
    method: "post",
    body: new URLSearchParams({
      id: id,
    })
  })
  .then(resp => resp.text())
  .then(data => {
    console.log(data);
    $('#dataTable').empty();
    getList();
  })
  .catch(err => console.log('Error : ', err));
}

function deleteTemp(id) {
  fetch('/admin/tempDel.jsp', {
    method: "post",
    body: new URLSearchParams({
      id: id,
    })
  })
  .then(resp => resp.text())
  .then(data => {
    console.log(data);
    $('#dataTable').empty();
    getList();
  })
  .catch(err => console.log('Error : ', err));
}

</script>
</body>
</html>

```

▼ tempAdd

```

<%@page import="jdbc.UserDAO"%>
<%@page import="jdbc.TempDAO"%>
<%@page import="jdbc.UserDTO"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html;charset=utf-8");

    String id = request.getParameter("id");

    if (UserDAO.insertAdmin(id)) {
        out.write("s");
    } else {
        out.write("f");
    }
%>
<%
    /* request.setCharacterEncoding("utf-8");
    response.setContentType("text/html;charset=utf-8");

    String id = request.getParameter("id");

    UserDTO user = TempDAO.getUser(id);
    if (user != null
        && UserDAO.insert(user.getId(), user.getPassword(), user.getName())
        && TempDAO.delete(id)) {

        out.write("s");
    } else {

        out.write("f");
    } */
%>

```

▼ tempCheck


```
<%@page import="jdbc.TempDAO"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
    out.print(TempDAO.selectListAJAX());
%>
```

그 외 기능 추가

- 관리자 멤버/피드 관리
- 내 피드 보기
- 내 피드 삭제

Python

Google Colaboratory

 <https://colab.research.google.com/drive/1q7CS936reUKHldGVRr0ZosNSYUMzpJMs?usp=sharing>

