Tugas
Praktikum Pemrograman II

UNPAS
Teknik Informatika
Universitas Pasundan

Laporan Tugas Pertemuan ke-12

Identitas

- NPM : 223040024
- Nama : Diaz Alfiari Rachmad
- Kelas : A
- URL Repository Github : https://github.com/Diazalfiari/pp2_223040024_A/tree/main/Latihan/sesi12_PdfReporting

Penjelasan Kode

Jika berkas **tidak memiliki output**, gunakan tabel ini.

mybatis-config.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <environments default="development">
        <environment id="development">
            <transactionManager type="JDBC" />
            <dataSource type="POOLED">
                <property name="driver"
value="com.mysql.cj.jdbc.Driver" />
                <property name="url"
value="jdbc:mysql://localhost:3306/pp2_sesi11" />
                <property name="username" value="root" />
                <property name="password" value="" />
            </dataSource>
        </environment>
    </environments>
    <mappers>
                <mapper class="model.ProductMapper"/>
        <package name="mapper" />
    </mappers>
</configuration>
```

Penjelasan

code diatas merupakan konfigurasi untuk MyBatis,sebuah framework java untuk ORM atau pengelolaan database

User.java

```
package model;

/**
 *
 * @author Diaza
 */
public class User {
    private int id;
    private String name;
    private String email;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

Penjelasan

User.java

code diatas merupakan definisi sebuah kelas Java bernama User,yang berada didalam paket model. Kelas ini digunakan untuk merepresentasikan sebuah entitas atau objek User dalam aplikasi,dengan atribut dan metode terkait.

UserMapper.java

```
package model;

import java.util.List;
import org.apache.ibatis.annotations.*;

/**
 *
 * @author Diaza
 */
public interface UserMapper {
    @Select("SELECT * FROM users")
    List<User> getAllUsers();

    @Insert("INSERT INTO users (name, email) VALUES (#{name},
#{email})")
    void insertUser(User user);
}
```

Penjelasan

code ini merupakan definisi interface java bernama userMapper, yang berada dalam paket model. Interface ini digunakan dalam kerangka MyBatis untuk memetahkan operasi database (SQL) ke metode Java. Tujuannya adalah untuk mempermudah akses manipulasi data di tabel users melalui kode Java.

MyBatisUtil.java

```
package model;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.*;

import java.io.IOException;
```

MyBatisUtil.java

```
/**
 *
 * @author Diaza
 */
public class MyBatisUtil {
    private static SqlSessionFactory sqlSessionFactory;

    static {
        try {
            sqlSessionFactory = new SqlSessionFactoryBuilder()

.build(Resources.getResourceAsStream("mybatis-config.xml"));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static SqlSession getSqlSession() {
        return sqlSessionFactory.openSession(true);
    }
}
```

Penjelasan

code ini mendefinisikan sebuah kelas Java bernama MyBatisUtil, yang berada didalam paket model. Kelas ini berfungsi sebahagai utilitas untuk mengatur dan menyediakan koneksi ke database menggunakan MyBatis. Bertujuan untuk menyederhanakan pembuatan SqlSession yang di gunakan untuk berinteraksi dengan database

UserView.java

```
package view;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionListener;
/**
 *
```

UserView.java

```
 * @author Diaza
 */
public class UserView extends JFrame {
    private JTextField txtName = new JTextField(20);
    private JTextField txtEmail = new JTextField(20);
    private JButton btnAdd = new JButton("Add User");
    private JButton btnRefresh = new JButton("Refresh");
    private JButton btnExport = new JButton("Export");
    private JList<String> userList = new JList<>();
    private DefaultListModel<String> listModel = new
DefaultListModel<>();

    public UserView() {
        setTitle("User Management");
        setSize (400, 300);
        setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);

        JPanel panel = new JPanel (new GridLayout (5, 1));
        panel.add(new JLabel ("Name:"));
        panel.add(txtName);
        panel.add(new JLabel("Email:"));
        panel.add(txtEmail);

        JPanel buttonPanel = new JPanel();
        buttonPanel.add(btnAdd);
        buttonPanel.add(btnRefresh);
        buttonPanel.add(btnExport);
        panel.add(buttonPanel);

        userList.setModel(listModel);
        add(panel, BorderLayout.NORTH);
        add(new JScrollPane (userList), BorderLayout.CENTER);
    }

    public String getNameInput () {
        return txtName.getText();
    }

    public String getEmailInput() {
        return txtEmail.getText();
```

UserView.java

```
    }

    public void setUserList (String[] users) {
        listModel.clear();
        for (String user : users) {
                listModel.addElement(user);
        }
    }

    public void addAddUserListener (ActionListener listener) {
        btnAdd.addActionListener(listener);
    }

    public void addRefreshListener (ActionListener listener) {
        btnRefresh.addActionListener(listener);
    }

    public void addExportListener(ActionListener listener){
        btnExport.addActionListener(listener);
    }
}
```

Penjelasan

UserView adalah kelas yang berperan sebagai antarmuka pengguna dalam aplikasi berbasis GUI. Kelas ini:

1. Menyediakan Input: Memungkinkan pengguna untuk memasukkan nama dan email.
2. Menampilkan Data: Menampilkan daftar pengguna dalam bentuk list (JList).
3. Interaksi Pengguna: Menyediakan tombol untuk menambahkan pengguna, menyegarkan daftar, dan mengekspor data, dengan dukungan ActionListener untuk menghubungkannya ke controller dalam pola MVC.
4. Komponen yang Modular: Dirancang agar dapat digunakan ulang dalam aplikasi dengan cara mudah diintegrasikan dengan logika bisnis.

UserController.java

```
package controller;

import model.*;
import view.UserView;
```

UserController.java

```
import view.UserPdf;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
/**
 *
 * @author Diaza
 */
public class UserController {
    private UserView view;
    private UserMapper mapper;
    private UserPdf pdf;

    public UserController(UserView view, UserMapper mapper, UserPdf
pdf) {
        this.view = view;
        this.mapper = mapper;
        this.pdf = pdf;

        this.view.addAddUserListener(new AddUserListener());
        this.view.addRefreshListener(new RefreshListener());
        this.view.addExportListener(new ExportListener());
    }

    class AddUserListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            String name = view.getNameInput();
            String email = view.getEmailInput();
            if (!name.isEmpty() && !email.isEmpty()) {
                User user = new User();
                user.setName(name);
                user.setEmail(email);
                mapper.insertUser(user);
                JOptionPane.showMessageDialog(view, "User added
successfully!");
```

UserController.java

```
            } else {
                JOptionPane.showMessageDialog(view, "please fill in
all fields.");
            }
        }
    }

    class RefreshListener implements ActionListener {
        @Override
        public void actionPerformed (ActionEvent e) {
            List<User> users = mapper.getAllUsers();
            String[] userArray = users.stream()
                                    .map(u -> u.getName()+ " (" +
u.getEmail() + ")")
                                    .toArray(String[]::new);
            view.setUserList(userArray);
        }
    }

    class ExportListener implements ActionListener{
        @Override
        public void actionPerformed(ActionEvent e){
            List<User> users = mapper.getAllUsers();
            pdf.exportPdf(users);
        }
    }
}
```

Penjelasan

User Controller ini:

1. Berfungsi sebagai penghubung antara View (antarmuka pengguna) dan Model (database).
2. Menangani logika aplikasi terkait penambahan pengguna, penyegaran data, dan ekspor PDF.
3. Menggunakan pola desain berbasis event listener untuk merespons aksi pengguna dari View.
4. Memanfaatkan MyBatis (UserMapper) untuk operasi basis data dan sebuah utilitas (UserPdf) untuk ekspor data ke format PDF.

Penjelasan Kode

Jika **berkas memiliki output**, gunakan tabel ini.

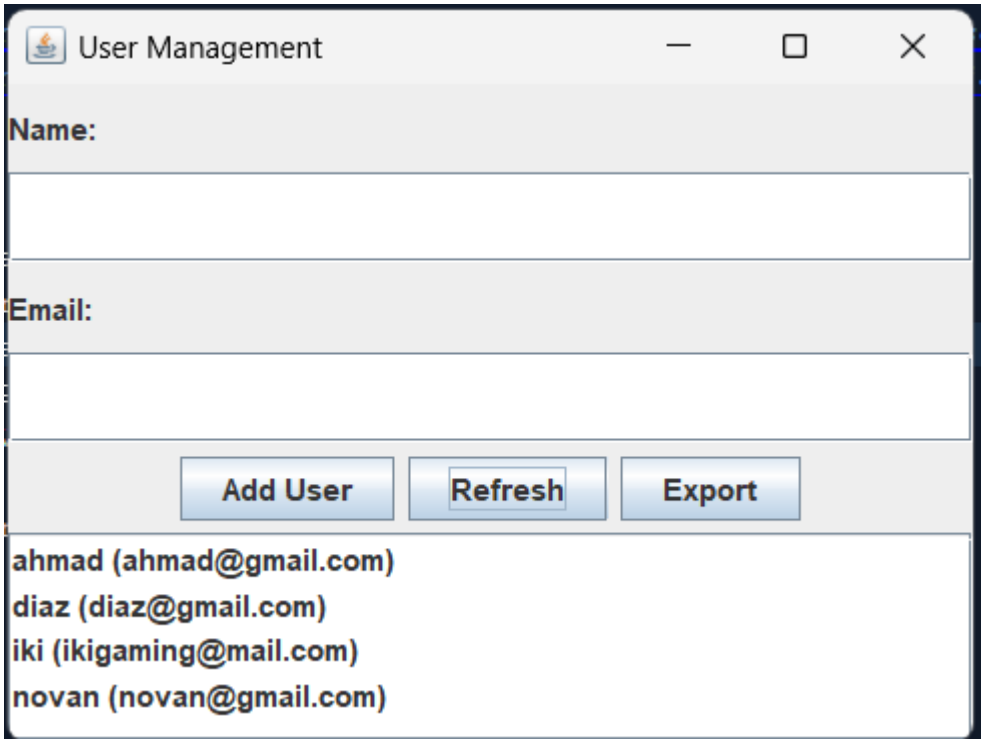
Main.java

```
package main;

import model.MyBatisUtil;
import model.UserMapper;
import org.apache.ibatis.session.SqlSession;
import view.UserPdf;
import view.UserView;

import controller.UserController;

/**
 *
 * @author Diaza
 */
public class Main {
    public static void main(String[] args) {
        SqlSession session = MyBatisUtil.getSqlSession();
        UserMapper mapper = session.getMapper(UserMapper.class);
        UserPdf pdf = new UserPdf();

        UserView view = new UserView();
        new UserController(view, mapper, pdf);

        view.setVisible(true);
    }
}
```

Penjelasan

Kode ini adalah penghubung yang mengatur komponen-komponen aplikasi sebelum dijalankan. Semua komponen (View, Model, Controller) diinisialisasi dan dikaitkan di sini, sehingga aplikasi dapat bekerja sebagai sebuah sistem yang terintegrasi.

Output



PDF Reporting:

No	Name	Email
1	ahmad	ahmad@gmail.com
2	diaz	diaz@gmail.com
3	iki	ikigaming@mail.com
4	novan	novan@gmail.com