DAO Pattern

Pada umumnya, DAO digunakan untuk aplikasi berbasis database, dimana aplikasi tersebut bertujuan memanipulasi data ke atau dari database.

Pengertian DAO

Data Access Object (DAO) adalah sebuah object yang menyediakan *interface* yang abstrak dimana ditujukan untuk aplikasi yang berhubungan dengan database.

DAO menyediakan beberapa operasi yang spesifik tanpa memaparkan rincian database, dan memberikan pemetaan pada aplikasi database. Pola desain ini sangat banyak diterapkan pada bahasa pemrograman Java.

DAO menawarkan perangkat tambahan yang baik dalam proses pengembangan perangkat lunak. Dengan memakai pola ini dapat memberikan ketahanan yang baik kepada aplikasi dan menambah fleksibelitas terhadap perubahan.

Keuntungan menggunakan DAO yang paling mendasar adalah relatif sederhana dan memisahkan proses yang terjadi pada aplikasi. Sehingga source code pada aplikasi menjadi lebih terstruktur dan mudah dipahami apabila terjadi kesalahan. Dimana perubahan yang terjadi pada persitence logic tidak akan mempengaruhi DAO yang terdapat pada client selama interface didefinisikan dengan benar. DAO dapat di implementasikan pada berbagai macam framework seperti hibernet, JPA, dll.

Keuntungan DAO

DAO memiliki beberapa keuntungan, antara lain adalah sebagai berikut :

- ✓ Modularitas program
  Memodulasi program menjadi kelas-kelas kecil dimana masing-masing kelas tersebut saling berinteraksi dengan kelas yang lainnya melalui Interface yang terdefinisi dengan benar.
- ✓ Penanganan kesalahan.
  Kesalahan di suatu modul program dapat dikoreksi tanpa perlu mempertimbangkan modul lainya, serta modul program dapat dipahami tanpa harus memahami keseluruhan.

- ✓ Pengorganisasian

  Pprogram merupakan kumpulan dari kelas yang terdefinisi dengan baik.
- ✓ Pengabstraksian.

  Masing-masing kelas berkorespondensi dengan satu abstraksi dan hanya satu-satunya.
- ✓ Penyederhanaan.

  Aplikasi yang kompleks dapat dibangun menjadi sederhana. Dengan mempartisi program menjadi komponen-komponen individual dapat mereduksi kompleksitas.

Konsep DAO Pada Java

DAO memiliki konsep tersendiri yang mana konsep tersebut sangat cocok digunakan pada bahasa pemrograman java. Hal ini dikarenakan Java sudah memberikan framework framework yang memperkuat dan melancarkan pola DAO.

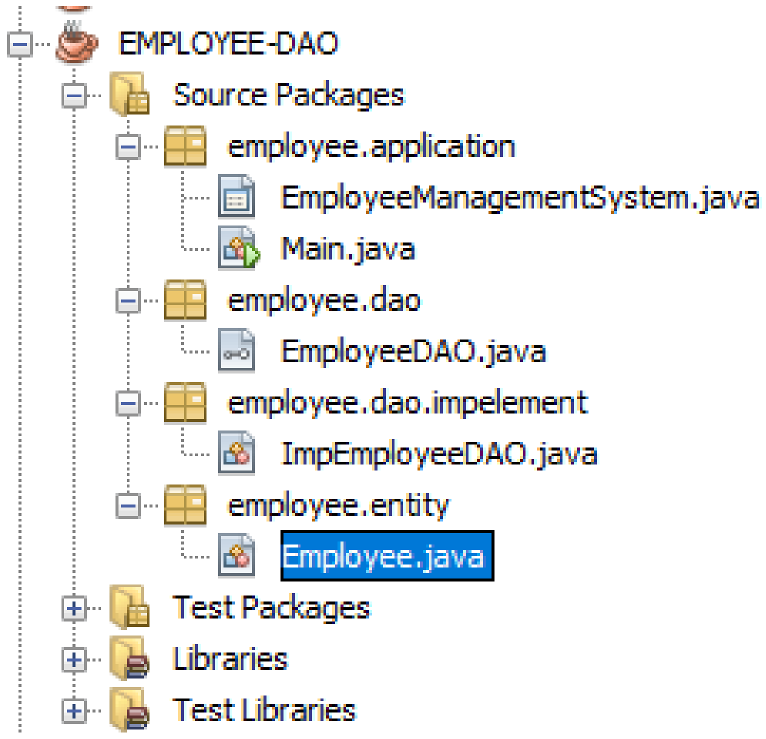
Berikut adalah konsep dasar DAO pada bahasa pemrograman Java :

a.	**Entity**	=	Class encapsulation yang berfungsi untuk menampung variabel-variabel (atribut kelas) yang nantinya dapat diakses oleh class lain.
b.	**Interface**	=	Prototype atau template atau abstraksi dari sebuah class, dalam hal ini bisa disebut class abstrak.
c.	**Implement**	=	Class implementasi atau pendeklarasian dari interface
d.	**Factory**	=	pemisah proses pembuatan sebuat object dari object lain.
e.	**View**	=	Antarmuka atau tampilan program (interface). Dalam hal ini adalah frame form.

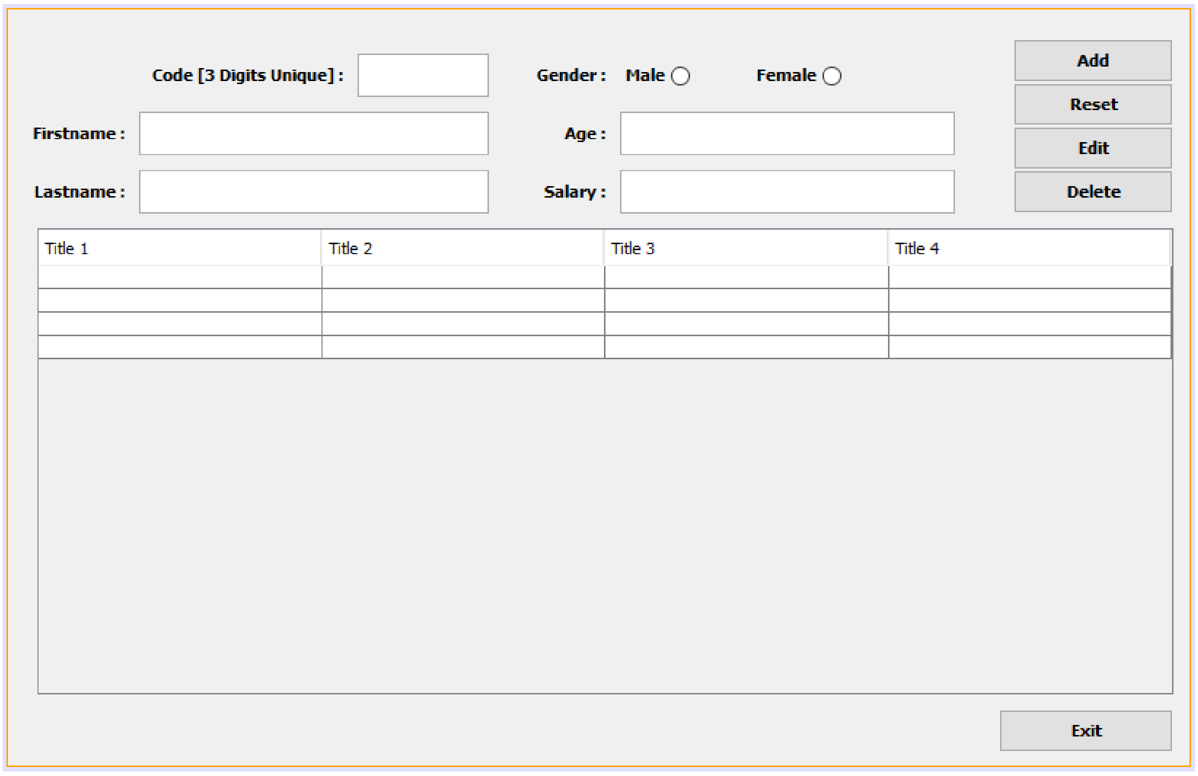
DAO merupakan OOP, sementara OOP sendiri bersifat melempar data dari satu class ke class lain. artinya kelima komponen tersebut nantinya saling berhubungan. Sehingga tidak akan jalan jika salah satunya saja terdapat error program ataupun error logic. Mungkin cukup menyusahkan. Tapi dengan begitu program akan lebih aman dan lebih teratur.

Contoh Implementasi DAO Pattern

A. Struktur Project Aplikasi Employee :



B. Antarmuka Grafis Aplikasi Employee :



C. Listing Program Aplikasi Employee :

Isi file : Employee.java

```
package employee.entity;

public class Employee {
    private String code;
    private String firstname;
    private String lastname;
    private String gender;
    private byte age;
    private int salary;

    public Employee() {
    }

    public void setCode(String code) {
        this.code = code;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }
```

```java
    public void setAge(byte age) {
        this.age = age;
    }

    public void setSalary(int salary) {
        this.salary = salary;
    }

    public String getCode() {
        return code;
    }

    public String getFirstname() {
        return firstname;
    }

    public String getLastname() {
        return lastname;
    }

    public String getGender() {
        return gender;
    }

    public byte getAge() {
        return age;
    }

    public int getSalary() {
        return salary;
    }

}
```

Isi file EmployeeDAO.java

```
package employee.dao;

import employee.entity.Employee;
import java.io.IOException;
import java.util.List;


public interface EmployeeDAO {
    public List<Employee> getAllEmployees() throws IOException;
    public Employee getEmployee(int row) throws IOException;
    public void insertEmployee(Employee e) throws IOException;
    public void deleteEmployee(Employee e) throws IOException;
    public void updateEmployee(int row, Employee e) throws IOException;

}
```

Isi file : ImplEmployeeDAO

```
package employee.dao.impelement;

import employee.dao.EmployeeDAO;
import employee.entity.Employee;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JOptionPane;


public class ImpEmployeeDAO implements EmployeeDAO{

    private List<Employee> employees;
    public ImpEmployeeDAO() {
        employees=new ArrayList<>();
    }


    @Override
    public List<Employee> getAllEmployees() throws IOException {
        return employees;
    }

    @Override
    public Employee getEmployee(int row) throws IOException {
        return employees.get(row);
    }

    @Override
    public void insertEmployee(Employee e) throws IOException {
        employees.add(e);
        JOptionPane.showMessageDialog(null, "Inserted Data Successfully", "Data Insert 
Information",JOptionPane.INFORMATION_MESSAGE);
    }

    @Override
    public void deleteEmployee(Employee e) throws IOException {
        employees.remove(e);
        JOptionPane.showMessageDialog(null, "Deleted Data Successfully", "Data Delete 
Information",JOptionPane.INFORMATION_MESSAGE);
    }

    @Override
    public void updateEmployee(int row, Employee e) throws IOException {
        Employee objEmployee=employees.get(row);
        objEmployee.setCode(e.getCode());
        objEmployee.setFirstname(e.getFirstname());
        objEmployee.setLastname(e.getLastname());
        objEmployee.setGender(e.getGender());
        objEmployee.setAge((byte) e.getAge());
        objEmployee.setSalary((int) e.getSalary());
        JOptionPane.showMessageDialog(null, "Updated Data Successfully", "Data Update
Information",JOptionPane.INFORMATION_MESSAGE);

    }
}
```

Isi file : EmployeeManagementSystem.java

```
package employee.application;

import employee.dao.EmployeeDAO;
import employee.dao.impelement.ImpEmployeeDAO;
import employee.entity.Employee;
import java.awt.event.ItemEvent;
import java.awt.event.KeyEvent;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import static javax.swing.JTable.AUTO_RESIZE_OFF;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableColumn;
import javax.swing.text.AttributeSet;
import javax.swing.text.BadLocationException;
import javax.swing.text.PlainDocument;
```

```java
public class EmployeeManagementSystem extends javax.swing.JFrame {

    private List<Employee> employees;
    private EmployeeDAO edao;
    private Employee e;
    private int row=-1;

    public EmployeeManagementSystem() {

        super("Employee Management System");
        initComponents();
        super.setResizable(false);
        super.setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE);
        showEmployeeData();
        resetForm();
        openedProtectForm(false);
        employees=new ArrayList<>();

    }


    private void resetForm(){

        String t="";
        txtCode.setText(t);
        txtCode.setDocument(new JTextFieldLimit(3));
        txtFirstname.setText(t);
        txtFirstname.setDocument(new JTextFieldLimit(30));
        txtLastname.setText(t);
        txtLastname.setDocument(new JTextFieldLimit(30));
        txtAge.setText(t);
        txtAge.setDocument(new JTextFieldLimit(3));
        txtSalary.setText(t);
        txtSalary.setDocument(new JTextFieldLimit(15));
        bgGender.clearSelection();
        btnAdd.setText("Add");
        btnUpdate.setText("Edit");
        btnDelete.setText("Delete");
        btnUpdate.setEnabled(false);
        btnDelete.setEnabled(false);
        btnAdd.setEnabled(true);

    }

    private void openedProtectForm(boolean response){

        txtCode.setEnabled(response);
        txtFirstname.setEnabled(response);
        txtLastname.setEnabled(response);
        txtAge.setEnabled(response);
        txtSalary.setEnabled(response);
        rbMale.setEnabled(response);
        rbFemale.setEnabled(response);

    }


    private void showEmployeeData(){

        try {
            Object columnNames[]=new Object[]{"#","Code","Firstname", "Lastname","Gender","Age","Salary"};
            DefaultTableModel dtm=new DefaultTableModel(null, columnNames){
                @Override
                public boolean isCellEditable(int row, int column) {
                    return false;
                }
            };
            tblEmployee.removeAll();
            tblEmployee.setModel(dtm);
            tblEmployee.setAutoResizeMode(AUTO_RESIZE_OFF);
            TableColumn tc;
            tc=tblEmployee.getColumnModel().getColumn(0);
            tc.setPreferredWidth(50);
            tc=tblEmployee.getColumnModel().getColumn(1);
            tc.setPreferredWidth(80);
            tc=tblEmployee.getColumnModel().getColumn(2);
            tc.setPreferredWidth(200);
            tc=tblEmployee.getColumnModel().getColumn(3);
            tc.setPreferredWidth(200);
            tc=tblEmployee.getColumnModel().getColumn(4);
            tc.setPreferredWidth(100);
            tc=tblEmployee.getColumnModel().getColumn(5);
            tc.setPreferredWidth(100);
            tc=tblEmployee.getColumnModel().getColumn(6);
            tc.setPreferredWidth(150);

            edao=new ImpEmployeeDAO();
            employees=edao.getAllEmployees();

            if(employees==null) return;
            int number=1;
            for(Employee employee:employees){
                dtm.addRow(new Object[]{number,employee.getCode(), employee.getFirstname(),
                                        employee.getLastname(),employee.getGender(),
                                        (int) employee.getAge(),(int) employee.getSalary()});

                number +=1;
            }
        } catch (IOException ex) {
            Logger.getLogger(EmployeeManagementSystem.class.getName()).log(Level.SEVERE, null, ex);
        }
```

```
    }


    private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {

        if(btnAdd.getText().equalsIgnoreCase("add")){
            btnAdd.setText("Save");
            openedProtectForm(true);
            txtCode.requestFocus();
        }
        else
        {
            e=new Employee();
            e.setCode(txtCode.getText());
            e.setFirstname(txtFirstname.getText().toUpperCase());
            e.setLastname(txtLastname.getText().toUpperCase());
            e.setGender((rbMale.isSelected()?"Male":"Female").toUpperCase());
            e.setAge(Byte.parseByte(txtAge.getText()));
            e.setSalary(Integer.parseInt(txtSalary.getText()));
            try {
                edao=new ImpEmployeeDAO();
                edao.insertEmployee(e);
                showEmployeeData();
            } catch (IOException ex) {
                Logger.getLogger(EmployeeManagementSystem.class.getName()).log(Level.SEVERE, null, ex);
            }

            btnAdd.setText("Add");
            resetForm();
            openedProtectForm(false);
        }

    }
```

```
private void btnResetActionPerformed(java.awt.event.ActionEvent evt) {

    resetForm(); openedProtectForm(false);
    tblEmployee.clearSelection();

}

private void txtFirstnameKeyReleased(java.awt.event.KeyEvent evt) {

    if(evt.getKeyCode()==KeyEvent.VK_ENTER){
        txtLastname.requestFocus();
    }

}

private void rbMaleItemStateChanged(java.awt.event.ItemEvent evt) {

    if(evt.getStateChange()==ItemEvent.SELECTED){
         txtAge.requestFocus();
    }

}

private void rbFemaleItemStateChanged(java.awt.event.ItemEvent evt) {

    if(evt.getStateChange()==ItemEvent.SELECTED){
         txtAge.requestFocus();
    }

}
```

```
private void txtAgeKeyReleased(java.awt.event.KeyEvent evt) {

    if(evt.getKeyCode()==KeyEvent.VK_ENTER){
        txtSalary.requestFocus();
    }

}

private void txtAgeKeyTyped(java.awt.event.KeyEvent evt) {

    if(!(Character.isDigit(evt.getKeyChar()))){
        evt.consume();
    }

}

private void txtSalaryKeyTyped(java.awt.event.KeyEvent evt) {

    if(!(Character.isDigit(evt.getKeyChar()))){
        evt.consume();
    }

}

private void txtLastnameKeyReleased(java.awt.event.KeyEvent evt) {

    if(evt.getKeyCode()==KeyEvent.VK_ENTER){
        rbMale.setSelected(true);
    }

}
```

```
    private void tblEmployeeMouseClicked(java.awt.event.MouseEvent evt) {

        row=tblEmployee.getSelectedRow();
        if(row>=0){
            txtCode.setText(tblEmployee.getValueAt(row, 1).toString());
            txtFirstname.setText(tblEmployee.getValueAt(row, 2).toString());
            txtLastname.setText(tblEmployee.getValueAt(row, 3).toString());
            if(tblEmployee.getValueAt(row, 4).toString().trim().equalsIgnoreCase("Male"))
                rbMale.setSelected(true);
            else
                rbFemale.setSelected(true);
            txtAge.setText(tblEmployee.getValueAt(row, 5).toString());
            txtSalary.setText(tblEmployee.getValueAt(row, 6).toString());

            btnUpdate.setEnabled(true);
            btnDelete.setEnabled(true);
            btnAdd.setEnabled(false);
        }

    }

    private void btnUpdateActionPerformed(java.awt.event.ActionEvent evt) {

        if(btnUpdate.getText().equalsIgnoreCase("edit")){
            btnUpdate.setText("Update");
            openedProtectForm(true);
        }
        else
        {
            try {
                edao=new ImpEmployeeDAO();
                e=edao.getEmployee(row);
                e.setCode(txtCode.getText());
                e.setFirstname(txtFirstname.getText().toUpperCase());
                e.setLastname(txtLastname.getText().toUpperCase());
                e.setGender((rbMale.isSelected()?"Male":"Female").toUpperCase());
                e.setAge(Byte.parseByte(txtAge.getText()));
                e.setSalary(Integer.parseInt(txtSalary.getText()));
                edao.updateEmployee(row, e);
                showEmployeeData();
            } catch (IOException ex) {
                Logger.getLogger(EmployeeManagementSystem.class.getName()).log(Level.SEVERE, null, ex);
            } finally {
                row=-1;
            }
            resetForm();
            openedProtectForm(false);
        }
    }

    private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt) {

            try {
                edao=new ImpEmployeeDAO();
                e=edao.getEmployee(row);
                edao.deleteEmployee(e);
                showEmployeeData();
            } catch (IOException ex) {
                Logger.getLogger(EmployeeManagementSystem.class.getName()).log(Level.SEVERE, null, ex);
            } finally {
                row=-1;
            }
            resetForm();
            openedProtectForm(false);

    }
```

```
private void txtCodeKeyReleased(java.awt.event.KeyEvent evt) {

    if(evt.getKeyCode()==KeyEvent.VK_ENTER){
        txtFirstname.requestFocus();
    }

}

private void txtCodeKeyTyped(java.awt.event.KeyEvent evt) {

    if(!(Character.isDigit(evt.getKeyChar()))){
        evt.consume();
    }

}

private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {

    System.exit(0);

}

//inner class
class JTextFieldLimit extends PlainDocument {

    private int limit;
    private boolean toUppercase = false;

    JTextFieldLimit(int limit) {
        super();
        this.limit = limit;
    }
```

```java
        JTextFieldLimit(int limit, boolean upper) {
            super();
            this.limit = limit;
            toUppercase = upper;
        }

        @Override
        public void insertString(int offset, String  str, AttributeSet attr){
            if (str == null) return;

            if ((getLength() + str.length()) <= limit) {
                try {
                    if (toUppercase) str = str.toUpperCase();
                    super.insertString(offset, str, attr);
                } catch (BadLocationException ex) {
                    Logger.getLogger(EmployeeManagementSystem.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
        }
    }


    // Variables declaration - do not modify
    private javax.swing.ButtonGroup bgGender;
    private javax.swing.JButton btnAdd;
    private javax.swing.JButton btnDelete;
    private javax.swing.JButton btnExit;
    private javax.swing.JButton btnReset;
    private javax.swing.JButton btnUpdate;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JRadioButton rbFemale;
    private javax.swing.JRadioButton rbMale;
    private javax.swing.JTable tblEmployee;
    private javax.swing.JTextField txtAge;
    private javax.swing.JTextField txtCode;
    private javax.swing.JTextField txtFirstname;
    private javax.swing.JTextField txtLastname;
    private javax.swing.JTextField txtSalary;
    // End of variables declaration
}
```

Isi file Main.java

```
package employee.application;

public class Main {
    public static void main(String[] args) {
        new EmployeeManagementSystem().setVisible(true);
    }
}
```