

Setelah melaksanakan praktikum ini diharapkan mahasiswa dapat :

- ✓ Menggunakan database SQLite sebagai sumber data
- ✓ Menggunakan ADO.NET sebagai teknologi akses data
- ✓ Menggunakan blok Try-Catch-Finally untuk penanganan error

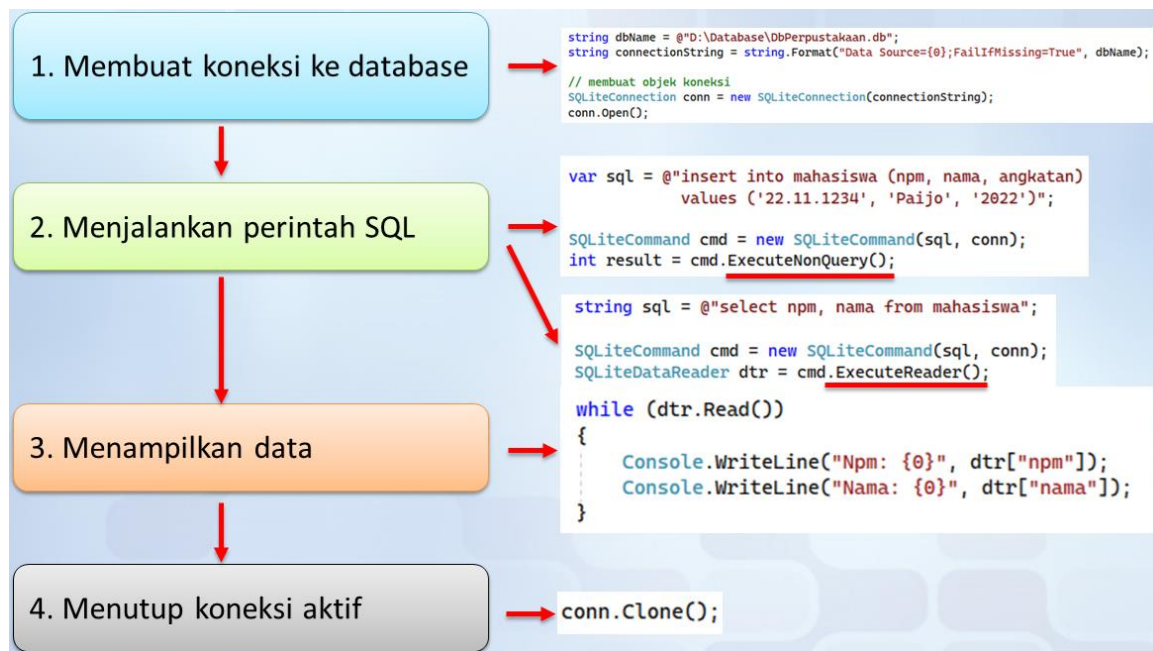
Teori Singkat

Pada praktikum kali ini, kita akan mempelajari cara menggunakan teknologi akses data ADO.NET untuk mengakses berbagai sumber data seperti file text, excel, dan sumber data lainnya seperti database.

Salah satu komponen ADO.NET yaitu *.NET Framework Data Provider* digunakan untuk melakukan koneksi ke database, mengeksekusi perintah SQL seperti INSERT, UPDATE, DELETE dan SELECT. Ada 3 class utama yang digunakan *.NET Framework Data Provider* untuk berinteraksi dengan database yaitu:

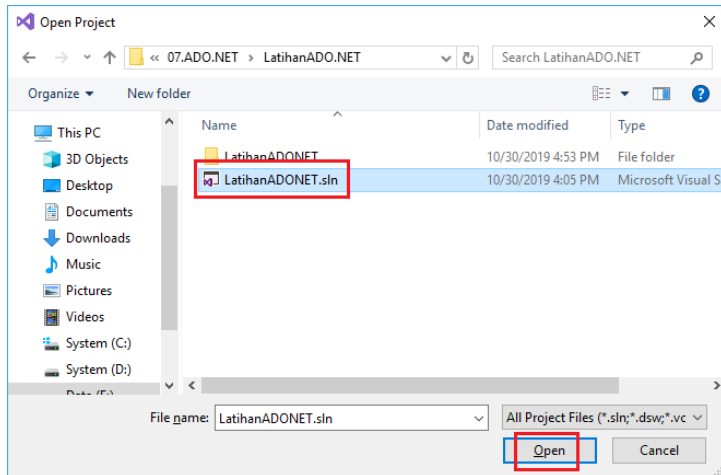
- ✓ Class Connection digunakan untuk membuat koneksi ke database.
- ✓ Class Command digunakan untuk menjalankan perintah SQL.
- ✓ Class DataReader digunakan untuk menampung hasil perintah SELECT.

Berikut adalah urutan langkah-langkah untuk mengakses database menggunakan *.NET Framework Data Provider*.

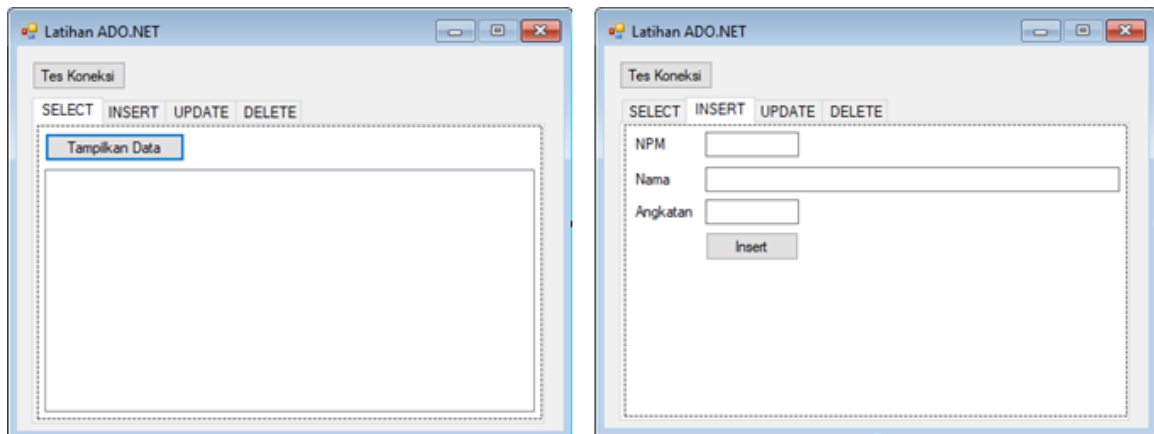


Persiapan Sebelum Praktikum

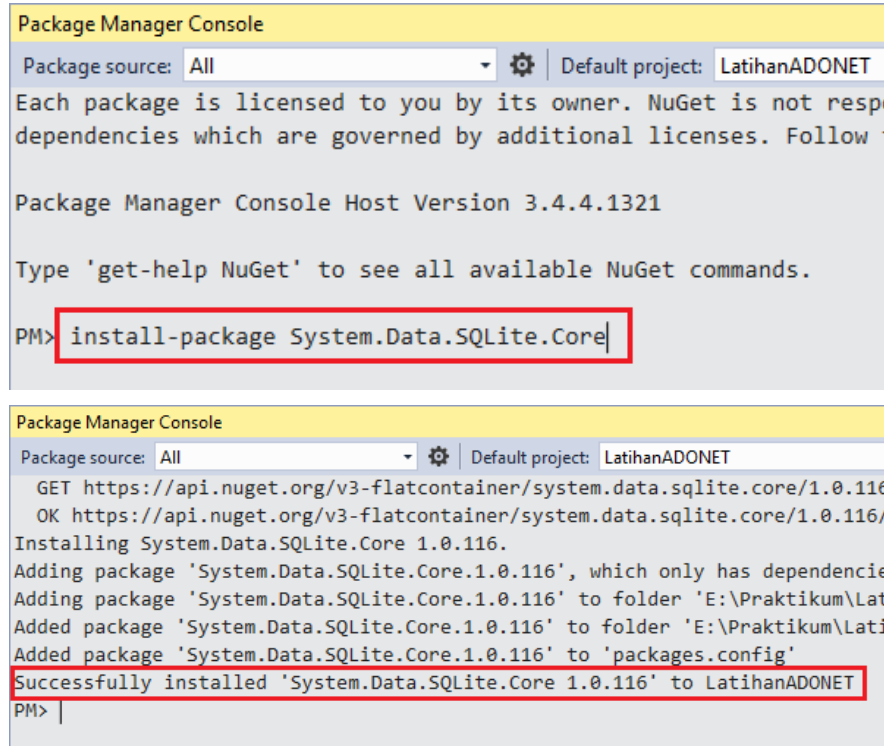
1. Awali selalu pekerjaan dgn doa, mudah-mudahan diberi kemudahan dan dapat memberikan manfaat
2. Jalankan aplikasi Microsoft Visual Studio .NET
3. Buka file solution LatihanADONET yang sudah disertakan pada modul praktikum. Melalui menu File -> Open -> Project/Solution ...



Pada solution ini sudah disertakan satu buah project berbasis Windows Form dengan nama LatihanADONET. Di project ini juga sudah dipersiapkan user interfacenya untuk keperluan uji coba teknologi ADO.NET.



4. Menginstall .NET Framework Data Provider for SQLite
 - ✓ Klik menu tools -> NuGet Package Manager -> Package Manager Console
 - ✓ Kemudian ketik perintah *install-package System.Data.SQLite.Core* pada console NuGet Package Manager



```
Package Manager Console
Package source: All | Default project: LatihanADONET
Each package is licensed to you by its owner. NuGet is not responsible for the
dependencies which are governed by additional licenses. Follow the license
agreements for each package you install.

Package Manager Console Host Version 3.4.4.1321

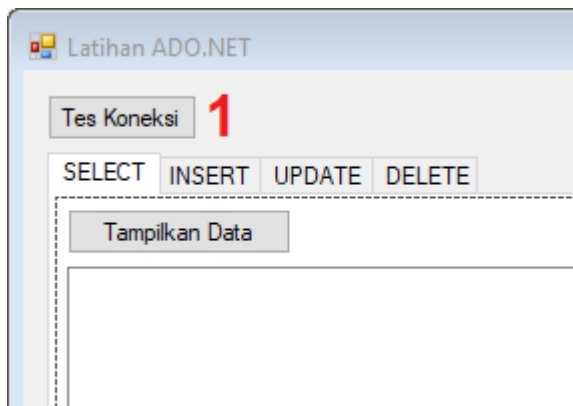
Type 'get-help NuGet' to see all available NuGet commands.

PM> install-package System.Data.SQLite.Core

Package Manager Console
Package source: All | Default project: LatihanADONET
GET https://api.nuget.org/v3-flatcontainer/system.data.sqlite.core/1.0.116/
OK https://api.nuget.org/v3-flatcontainer/system.data.sqlite.core/1.0.116/
Installing System.Data.SQLite.Core 1.0.116.
Adding package 'System.Data.SQLite.Core.1.0.116', which only has dependencies
Adding package 'System.Data.SQLite.Core.1.0.116' to folder 'E:\Praktikum\LatihanADONET'
Added package 'System.Data.SQLite.Core.1.0.116' to folder 'E:\Praktikum\LatihanADONET'
Added package 'System.Data.SQLite.Core.1.0.116' to 'packages.config'
Successfully installed 'System.Data.SQLite.Core 1.0.116' to LatihanADONET
PM>
```

Latihan 7.1 (Tes Koneksi ke Database)

1. Desain User Interface



2. Pengaturan Properties

No	Komponen	Property	Nilai/Value
1	Button	Name Text	btnTesKoneksi Tes Koneksi

3. Kode Program

Aktifkan editor code, kemudian di bagian deklarasi namespace tambahkan pemanggilan namespace *System.Data.SQLite*. Namespace *System.Data.SQLite* perlu kita tambahkan agar kita bisa mengakses semua class dari *.NET Framework Data Provider for SQLite* yang diperlukan untuk mengakses database SQLite.

```
3 | using System.Windows.Forms;
4 |
5 | using System.Data.SQLite;
6 |
7 | namespace LatihanADONET
8 | {
9 |     public partial class Form1 : Form
10 |    {
11 |        // constructor
12 |        public Form1()
13 |        {
14 |            InitializeComponent();
```

Setelah itu kembali ke desain form, kemudian klik ganda tombol *Tes Koneksi* untuk mengaktifkan event *btnTesKoneksi_Click*.

```
private void btnTesKoneksi_Click(object sender, EventArgs e)
{
    ...
}
```

Kemudian lengkapi kodenya seperti berikut:

```
private void btnTesKoneksi_Click(object sender, EventArgs e)
{
    // membuat objek connection
    SQLiteConnection conn = GetOpenConnection();

    // cek status koneksi
    if (conn.State == ConnectionState.Open) // koneksi berhasil
    {
        MessageBox.Show("Koneksi ke database berhasil !", "Informasi",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    }
    else
        MessageBox.Show("Koneksi ke database gagal !!!", "Informasi",
            MessageBoxButtons.OK,
            MessageBoxIcon.Exclamation);

    conn.Dispose(); // tutup dan hapus objek connection dari memory
}
```

Pada kode di atas pada saat pembuatan objek *Connection*, kita memanggil sebuah method dengan nama *GetOpenConnection*. Method inilah yang bertugas untuk membuat objek *Connection* dan melakukan koneksi ke database. Adapun kode untuk method *GetOpenConnection* seperti berikut:

```
private SQLiteConnection GetOpenConnection()
{
    SQLiteConnection conn = null; // deklarasi objek connection

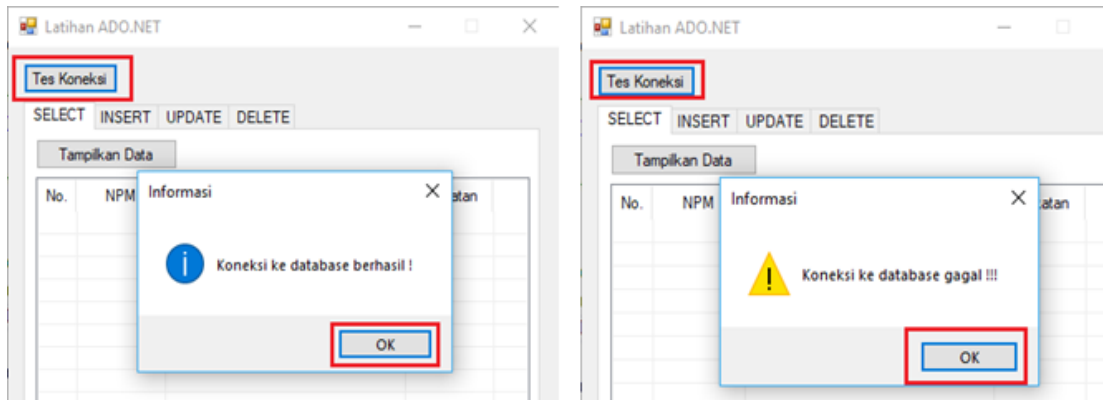
    try // penggunaan blok try-catch untuk penanganan error
    {
        // atur ulang lokasi database yang disesuaikan dengan
        // lokasi database perpustakaan Anda
        string dbName = @"D:\Database\DbPerpustakaan.db";

        // deklarasi variabel connectionString, ref:
        https://www.connectionstrings.com/
        string connectionString = string.Format("Data
        Source={0};FailIfMissing=True", dbName);

        conn = new SQLiteConnection(connectionString); // buat objek
        connection
        conn.Open(); // buka koneksi ke database
    }
    // jika terjadi error di blok try, akan ditangani langsung oleh blok
    catch
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message, "Error",
        MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }

    return conn;
}
```

4. Output



Setelah itu jalankan aplikasi dengan menekan tombol F5 (Start Debugging) kemudian klik tombol *Tes Koneksi*. Jika koneksi ke **database gagal**, silahkan diperbaiki dan jangan lanjut dulu ke latihan berikutnya.

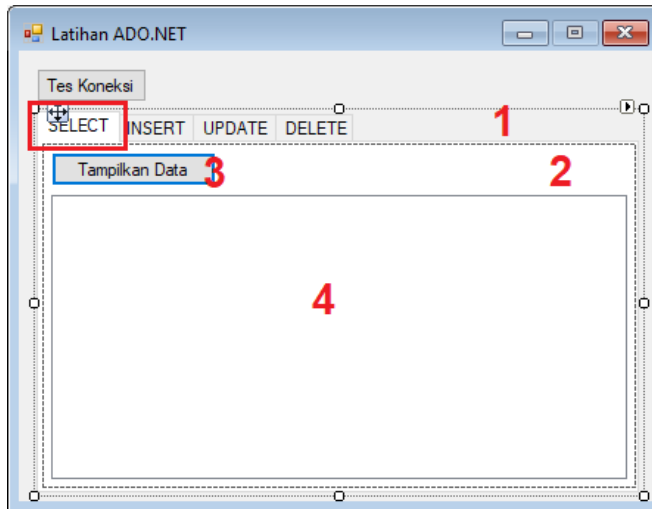
Latihan 7.2 (Menampilkan Data Mahasiswa)

Setelah tes koneksi ke database berhasil, latihan berikutnya adalah kita akan menampilkan data mahasiswa dengan format tabel menggunakan komponen ListView. Adapun perintah *SQL* yang digunakan untuk mengambil/membaca data dari sebuah tabel adalah *SELECT*. Berikut adalah contoh penggunaan perintah *SELECT* untuk menampilkan semua data mahasiswa yang diambil dari tabel mahasiswa.

```
select npm, nama, angkatan  
from mahasiswa  
order by nama
```

Nah untuk menjalankan perintah *SELECT* di atas kita membutuhkan objek dari class *Command*.

1. Desain User Interface



2. Pengaturan Properties

No	Komponen	Property	Nilai/Value
1	TabControl	-	-
2	TabPage1	Text	SELECT
3	Button	Name	btnTampilkanData
		Text	Tampilkan Data
4	ListView	Name	lvwMahasiswa

3. Kode Program

Aktifkan code editor kemudian tambahkan method *InisialisasiListView* yang digunakan untuk memformat tampilan objek ListView

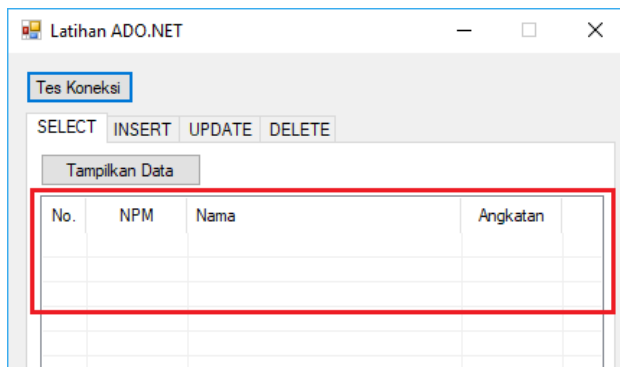
```
// atur format listview
private void InisialisasiListView()
{
    lvwMahasiswa.View = View.Details;
    lvwMahasiswa.FullRowSelect = true;
    lvwMahasiswa.GridLines = true;

    lvwMahasiswa.Columns.Add("No.", 30, HorizontalAlignment.Center);
    lvwMahasiswa.Columns.Add("NPM", 70, HorizontalAlignment.Center);
    lvwMahasiswa.Columns.Add("Nama", 190, HorizontalAlignment.Left);
    lvwMahasiswa.Columns.Add("Angkatan", 70, HorizontalAlignment.Center);
}
```

Setelah itu panggil method *InisialisasiListView* dari dalam constructor.

```
public partial class Form1 : Form
{
    // constructor
    public Form1()
    {
        InitializeComponent();
        InisialisasiListView();
    }
}
```

Sampai tahap ini Anda bisa mencoba untuk menjalankan aplikasi, dan cek apakah tampilannya sudah seperti gambar berikut?



Setelah itu kembali lagi ke desain form, kemudian klik ganda tombol *Tampilkan Data* untuk mengaktifkan event *btnTampilkanData_Click*.

```
private void btnTampilkanData_Click(object sender, EventArgs e)
{
    ...
}
```

Kemudian lengkapi kodenya seperti berikut:

```
private void btnTampilkanData_Click(object sender, EventArgs e)
{
    lvwMahasiswa.Items.Clear();

    // membuat objek Connection, sekaligus buka koneksi ke database
    SQLiteConnection conn = GetOpenConnection();

    // deklarasi variabel sql untuk menampung perintah SELECT
    string sql = @"select npm, nama, angkatan
                  from mahasiswa
                  order by nama";

    // membuat objek Command untuk mengeksekusi perintah SQL
    SQLiteCommand cmd = new SQLiteCommand(sql, conn);

    // membuat objek DataReader untuk menampung hasil perintah SELECT
    SQLiteDataReader dtr = cmd.ExecuteReader(); // eksekusi perintah SELECT

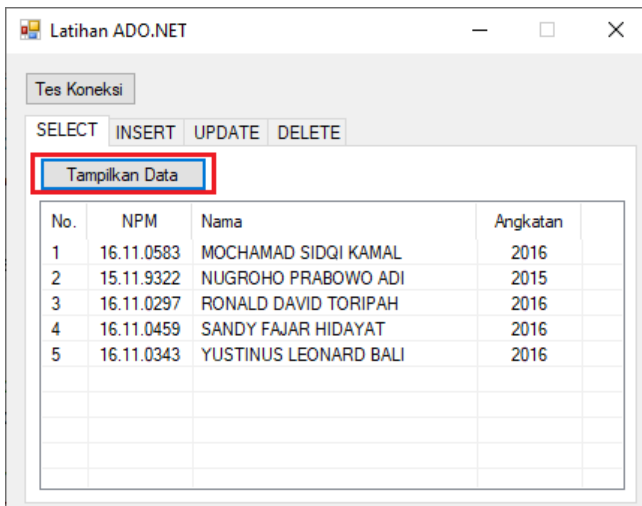
    while (dtr.Read()) // gunakan perulangan utk menampilkan data ke
listview
    {
        var noUrut = lvwMahasiswa.Items.Count + 1;

        var item = new ListViewItem(noUrut.ToString());
        item.SubItems.Add(dtr["npm"].ToString());
        item.SubItems.Add(dtr["nama"].ToString());
        item.SubItems.Add(dtr["angkatan"].ToString());

        lvwMahasiswa.Items.Add(item);
    }

    // setelah selesai digunakan,
    // segera hapus objek datareader, command dan connection dari memory
    dtr.Dispose();
    cmd.Dispose();
    conn.Dispose();
}
```

4. Output



No.	NPM	Nama	Angkatan	
1	16.11.0583	MOCHAMAD SIDQI KAMAL	2016	
2	15.11.9322	NUGROHO PRABOWO ADI	2015	
3	16.11.0297	RONALD DAVID TORIPAH	2016	
4	16.11.0459	SANDY FAJAR HIDAYAT	2016	
5	16.11.0343	YUSTINUS LEONARD BALI	2016	

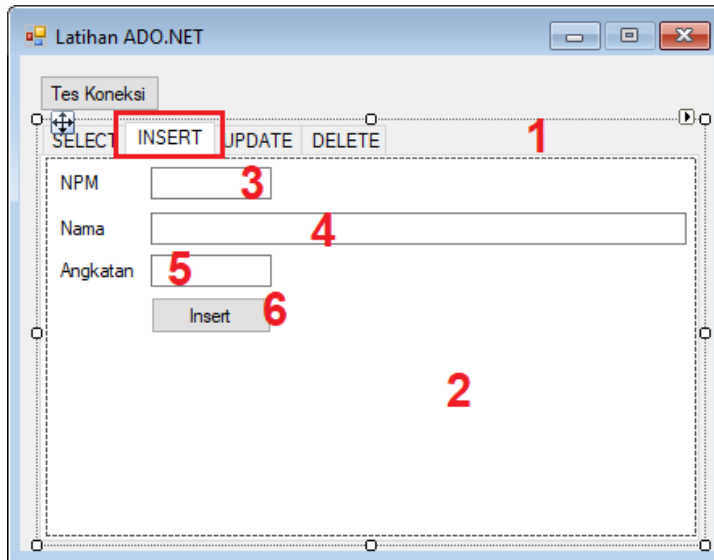
Latihan 7.3 (Menambahkan Data Mahasiswa)

Latihan berikutnya adalah kita akan membuat form input untuk menambahkan data mahasiswa. Adapun perintah *SQL* yang digunakan untuk menambahkan data ke dalam sebuah tabel adalah *INSERT*. Contoh penggunaan perintah *INSERT* seperti berikut.

```
insert into mahasiswa (npm, nama, angkatan)
values ('18.11.1234', 'Paijo', '2018')
```

Untuk menjalankan perintah *INSERT* di atas kita juga membutuhkan objek dari class *Command*.

1. Desain User Interface



2. Pengaturan Properties

No	Komponen	Property	Nilai/Value
1	tabControl	-	-
2	tabPage2	Text	Insert
3	TextBox	Name	txtNpmInsert
4	TextBox	Name	txtNamaInsert
5	TextBox	Name	txtAngkatanInsert
6	Button	Name Text	btnInsert Insert

3. Kode Program

Klik ganda tombol *Insert* untuk mengaktifkan event *btnInsert_Click*.

```
private void btnInsert_Click(object sender, EventArgs e)
{
    ...
}
```

Kemudian lengkapi kodenya seperti berikut:

```
private void btnInsert_Click(object sender, EventArgs e)
{
    var result = 0;

    // validasi npm harus diisi
    if (string.IsNullOrEmpty(txtNpmInsert.Text))
    {
        MessageBox.Show("NPM harus diisi !!!", "Informasi",
        MessageBoxButtons.OK,
        MessageBoxIcon.Exclamation);

        txtNpmInsert.Focus();
        return;
    }

    // validasi nama harus diisi
    if (string.IsNullOrEmpty(txtNamaInsert.Text))
    {
        MessageBox.Show("Nama harus diisi !!!", "Informasi",
        MessageBoxButtons.OK,
        MessageBoxIcon.Exclamation);

        txtNamaInsert.Focus();
        return;
    }

    // membuat objek Connection, sekaligus buka koneksi ke database
    SQLiteConnection conn = GetOpenConnection();

    // deklarasi variabel sql untuk menampung perintah INSERT
    var sql = @"insert into mahasiswa (npm, nama, angkatan)
    values (@npm, @nama, @angkatan)";

    // membuat objek Command untuk mengeksekusi perintah SQL
    SQLiteCommand cmd = new SQLiteCommand(sql, conn);
```

Lanjutan kode di halaman berikutnya.

```
try
{
    // set parameter untuk nama, angkatan dan npm
    cmd.Parameters.AddWithValue("@npm", txtNpmInsert.Text);
    cmd.Parameters.AddWithValue("@nama", txtNamaInsert.Text);
    cmd.Parameters.AddWithValue("@angkatan",
txtAngkatanInsert.Text);

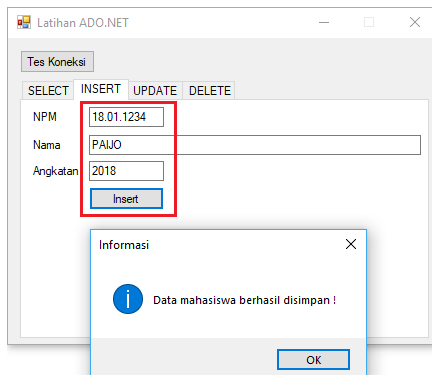
    result = cmd.ExecuteNonQuery(); // eksekusi perintah INSERT
}
catch (Exception ex)
{
    MessageBox.Show("Error: " + ex.Message, "Error",
MessageBoxButtons.OK,
MessageBoxIcon.Error);
}
finally
{
    cmd.Dispose();
}

if (result > 0)
{
    MessageBox.Show("Data mahasiswa berhasil disimpan !",
"Informasi", MessageBoxButtons.OK,
MessageBoxIcon.Information);

    // reset form
    txtNpmInsert.Clear();
    txtNamaInsert.Clear();
    txtAngkatanInsert.Clear();
    txtNpmInsert.Focus();
}
else
    MessageBox.Show("Data mahasiswa gagal disimpan !!!",
"Informasi", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);

// setelah selesai digunakan,
// segera hapus objek connection dari memory
conn.Dispose();
}
```

4. Output



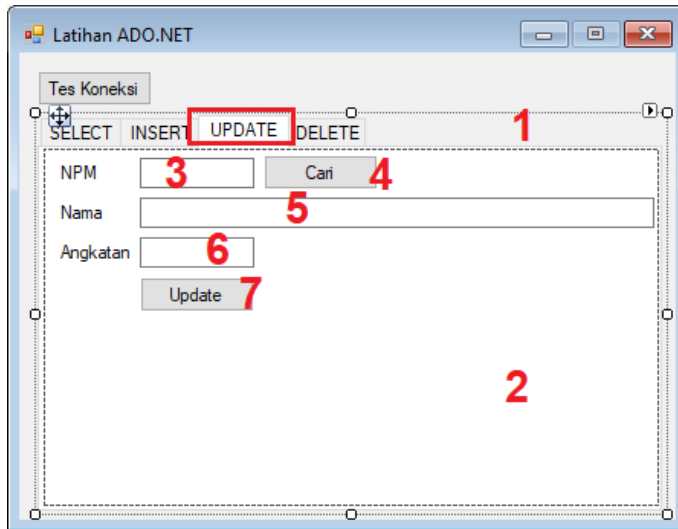
Latihan 7.4 (Mengupdate Data Mahasiswa)

Latihan berikutnya adalah kita akan membuat form input untuk mengupdate data mahasiswa. Adapun perintah *SQL* yang digunakan untuk menambahkan data ke dalam sebuah tabel adalah *UPDATE*. Contoh penggunaan perintah *UPDATE* seperti berikut.

```
update mahasiswa set nama = 'Jhono', angkatan = '2018'
where npm = '18.11.1234'
```

Untuk menjalankan perintah *UPDATE* di atas kita juga membutuhkan objek dari class *Command*.

1. Desain User Interface



2. Pengaturan Properties

No	Komponen	Property	Nilai/Value
1	tabControl1	TabPage	-
2	tabPage3	Text	Update
3	textBox	Name	txtNpmUpdate
4	button	Name Text	btnCariUpdate Cari
5	textBox	Name	txtNamaUpdate
6	textBox	Name	txtAngkatanUpdate
7	button	Name Text	btnUpdate Update

3. Kode Program

Pada latihan ini kita akan menambahkan 2 blok kode yaitu blok kode untuk pencarian data mahasiswa dengan menggunakan perintah *SELECT* seperti berikut:

```
select npm, nama, angkatan
from mahasiswa
where npm = '18.01.1234'
```

dan blok kode untuk mengupdate data mahasiswa

```
update mahasiswa set nama = 'Jhono', angkatan = '2018'  
where npm = '18.01.1234'
```

Yang pertama kita akan melengkapi kode untuk pencarian data mahasiswa. Caranya dengan mengklik ganda tombol Cari (tab Update).

```
private void btnCariUpdate_Click(object sender, EventArgs e)  
{  
    ...  
}
```

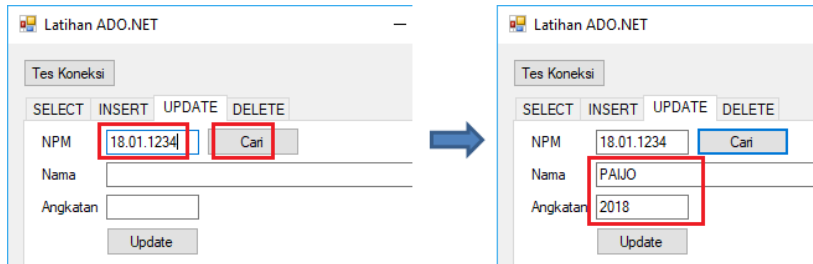
Kemudian lengkapi kodenya seperti berikut:

```
private void btnCariUpdate_Click(object sender, EventArgs e)  
{  
    // validasi npm harus diisi  
    if (string.IsNullOrEmpty(txtNpmUpdate.Text))  
    {  
        MessageBox.Show("NPM harus !!!", "Informasi", MessageBoxButtons.OK,  
            MessageBoxIcon.Exclamation);  
  
        txtNpmUpdate.Focus();  
        return;  
    }  
  
    // membuat objek Connection, sekaligus buka koneksi ke database  
    SQLiteConnection conn = GetOpenConnection();  
  
    // deklarasi variabel sql untuk menampung perintah SELECT  
    string sql = @"select npm, nama, angkatan  
        from mahasiswa  
        where npm = @npm";  
  
    // membuat objek Command untuk mengeksekusi perintah SQL  
    SQLiteCommand cmd = new SQLiteCommand(sql, conn);  
    cmd.Parameters.AddWithValue("@npm", txtNpmUpdate.Text);  
  
    // membuat objek DataReader untuk menampung hasil perintah SELECT  
    SQLiteDataReader dtr = cmd.ExecuteReader(); // eksekusi perintah SELECT  
  
    if (dtr.Read()) // data ditemukan  
    {  
        // tampilkan nilainya ke textbox  
        txtNpmUpdate.Text = dtr["npm"].ToString();  
        txtNamaUpdate.Text = dtr["nama"].ToString();  
        txtAngkatanUpdate.Text = dtr["angkatan"].ToString();  
    }  
    else  
        MessageBox.Show("Data mahasiswa tidak ditemukan !", "Informasi",  
            MessageBoxButtons.OK,  
            MessageBoxIcon.Information);  
}
```

Lanjutkan kode di halaman berikutnya.

```
// setelah selesai digunakan,  
// segera hapus objek datareader, command dan connection dari memory  
dtr.Dispose();  
cmd.Dispose();  
conn.Dispose();  
}
```

Sampai di sini kita bisa mencoba apakah tombol pencariannya berfungsi dengan baik atau tidak dengan cara menginputkan salah satu NPM yang ada kemudian klik tombol Cari.



Dan jika pencarian data berhasil menampilkan informasi nama dan angkatan, Anda baru boleh melanjutkan ke langkah berikutnya.

Setelah pencarian data berhasil, kita akan menambahkan kode untuk Update, caranya sama yaitu dengan mengklik ganda tombol Update.

```
private void btnUpdate_Click(object sender, EventArgs e)  
{  
    ...  
}
```

Kemudian lengkapi kodenya seperti berikut:

```
private void btnUpdate_Click(object sender, EventArgs e)  
{  
    var result = 0;  
  
    // validasi npm harus diisi  
    if (string.IsNullOrEmpty(txtNpmUpdate.Text))  
    {  
        MessageBox.Show("NPM harus !!!", "Informasi", MessageBoxButtons.OK,  
            MessageBoxIcon.Exclamation);  
  
        txtNpmUpdate.Focus();  
        return;  
    }  
  
    // validasi nama harus diisi  
    if (string.IsNullOrEmpty(txtNamaUpdate.Text))  
    {  
        MessageBox.Show("Nama harus !!!", "Informasi", MessageBoxButtons.OK,  
            MessageBoxIcon.Exclamation);  
  
        txtNamaUpdate.Focus();  
        return;  
    }  
}
```

```
// membuat objek Connection, sekaligus buka koneksi ke database
SQLiteConnection conn = GetOpenConnection();

// deklarasi variabel sql untuk menampung perintah UPDATE
string sql = @"update mahasiswa set nama = @nama, angkatan = @angkatan
              where npm = @npm";

// membuat objek Command untuk mengeksekusi perintah SQL
SQLiteCommand cmd = new SQLiteCommand(sql, conn);

try
{
    // set parameter untuk nama, angkatan dan npm
    cmd.Parameters.AddWithValue("@nama", txtNamaUpdate.Text);
    cmd.Parameters.AddWithValue("@angkatan", txtAngkatanUpdate.Text);
    cmd.Parameters.AddWithValue("@npm", txtNpmUpdate.Text);

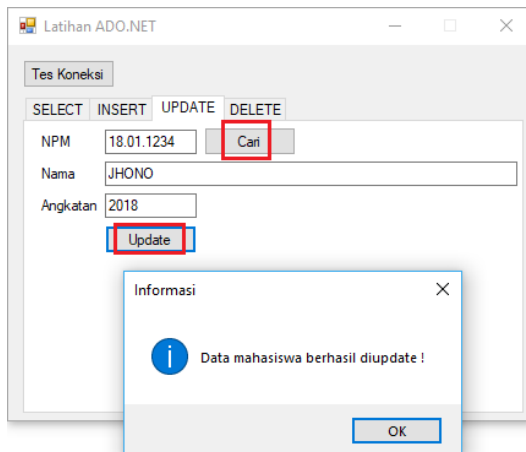
    result = cmd.ExecuteNonQuery();
}
catch (Exception ex)
{
    MessageBox.Show("Error: " + ex.Message, "Error", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
}
finally
{
    cmd.Dispose();
}

if (result > 0)
{
    MessageBox.Show("Data mahasiswa berhasil diupdate !", "Informasi",
        MessageBoxButtons.OK,
        MessageBoxIcon.Information);

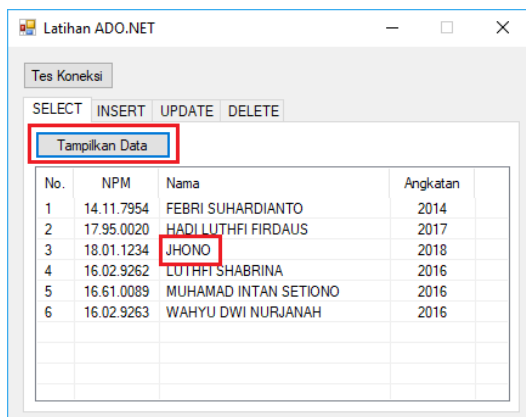
    // reset form
    txtNpmUpdate.Clear();
    txtNamaUpdate.Clear();
    txtAngkatanUpdate.Clear();
    txtNpmUpdate.Focus();
}
else
{
    MessageBox.Show("Data mahasiswa gagal diupdate !!!", "Informasi",
        MessageBoxButtons.OK,
        MessageBoxIcon.Exclamation);

    // setelah selesai digunakan,
    // segera hapus objek connection dari memory
    conn.Dispose();
}
```

4. Output

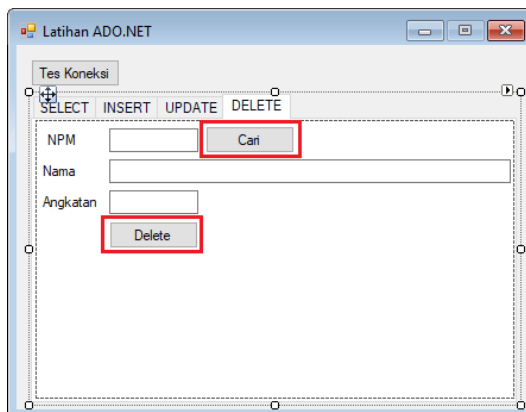


Setelah update berhasil, silahkan cek di tab SELECT, apakah nama PAIJO sudah berubah menjadi JHONO.



Tugas 7.1

- ✓ Lengkapi kode untuk pencarian dan hapus data mahasiswa



✓ Output

The screenshot shows the 'Latihan ADO.NET' application window. The 'Tes Koneksi' button is at the top. Below it are tabs for 'SELECT', 'INSERT', 'UPDATE', and 'DELETE'. The 'Tampilkan Data' button is highlighted with a red box. Below the button is a table with the following data:

No.	NPM	Nama	Angkatan
1	14.11.7954	FEBRI SUHARDIANTO	2014
2	17.95.0020	HADI LUTHFI FIRDAUS	2017
3	18.01.1234	JHONO	2018
4	16.02.9262	LUTHFI SHABRINA	2016
5	16.61.0089	MUHAMAD INTAN SETIONO	2016
6	16.02.9263	WAHYU DWI NURJANAH	2016

The screenshot shows the 'Latihan ADO.NET' application window. The 'Tes Koneksi' button is at the top. Below it are tabs for 'SELECT', 'INSERT', 'UPDATE', and 'DELETE'. The 'Cari' button is highlighted with a red box. Below the 'Cari' button are input fields for 'NPM' (18.01.1234), 'Nama' (JHONO), and 'Angkatan' (2018). The 'Delete' button is highlighted with a red box. A 'Konfirmasi' dialog box is shown with a yellow warning icon and the text 'Apakah data mahasiswa ingin dihapus?'. The 'Yes' button is highlighted with a red box.

The screenshot shows the 'Latihan ADO.NET' application window. The 'Tes Koneksi' button is at the top. Below it are tabs for 'SELECT', 'INSERT', 'UPDATE', and 'DELETE'. The 'Tampilkan Data' button is highlighted with a red box. Below the button is a table with the following data:

No.	NPM	Nama	Angkatan
1	14.11.7954	FEBRI SUHARDIANTO	2014
2	17.95.0020	HADI LUTHFI FIRDAUS	2017
3	16.02.9262	LUTHFI SHABRINA	2016
4	16.61.0089	MUHAMAD INTAN SETIONO	2016
5	16.02.9263	WAHYU DWI NURJANAH	2016

Data JHONO sudah di hapus

Selesai ☺

Kamarudin, M.Kom

<http://coding4ever.net/>
<https://github.com/rudi-krsoftware/open-retail>