IlmuKomputer.Com
Ikatlah ilmu dengan menuliskannya

ADO.NET Tutorial

Alex Budiyanto
alex@alexbudiyanto.web.id
http://alexbudiyanto.web.id/

Lisensi Dokumen:
Copyright © 2003-2007 IlmuKomputer.Com
Seluruh dokumen di IlmuKomputer.Com dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari IlmuKomputer.Com.

Materi ini saya bawakan dalam workshop Building Desktop Application pada tanggal 10 Maret 2007 di Teknik Informatika – Universitas Atma Jaya Yogyakarta. Workshop tersebut, dibagi dalam empat sesi, dan salah satu sesi tersebut adalah ADO.NET, yang saya bawakan sendiri. Dalam workshop mengenai ADO.NET ini, peserta diharapkan mengerti tentang arsitektur umum ADO.NET, menulis stored procedure, dan memanggil stored procedure tersebut dari aplikasi. Selain itu, dalam tutorial ini, juga dikenalkan bagaimana memanipulasi data dan menampilkan data, menggunakan control DataGridView.

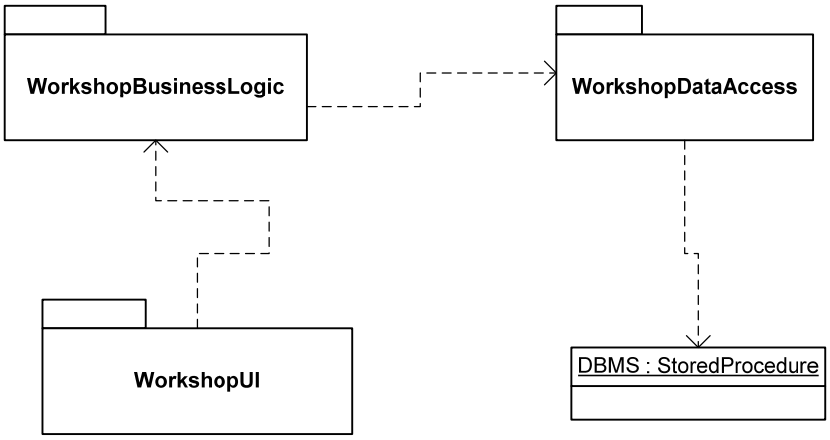
Tujuan Hands On Lab

Dalam Hands on lab kali ini kita akan belajar tentang ADO.NET yang akan langsung dipraktekan penggunaannya dalam pembuatan sebuah aplikasi kecil. Dalam pembuatan aplikasi ini kita akan belajar tentang :

- Arsitektur umum aplikasi berbasis teknologi Windows Forms.
- Membuat Stored Procedure yang akan ditanam pada sebuah DBMS
- Mengatahui berbagai method yang dimiliki oleh ADO.NET seperti :
    - ExecuteNonQuery() method
    - ExecuteReader() method
    - ExecuteScalar() method
    - SQL Transaction *

Arsitektur Aplikasi Hands On Lab

Dalam HOL kali ini kita akan membuat sebuah aplikasi dengan arsitekur seperti gambar dibawah ini :

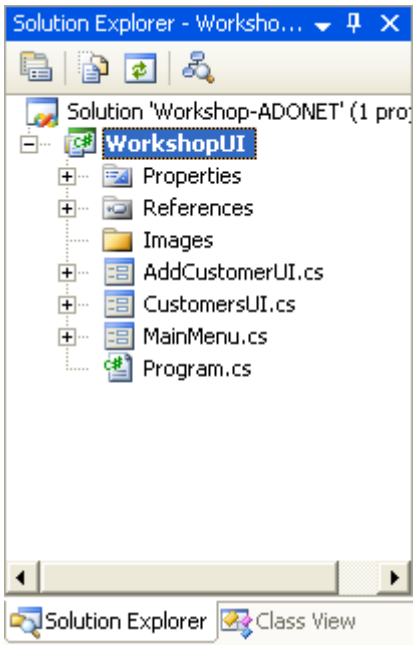


Gambar 1. Arsitektur Aplikasi

Dari gambar tersebut terlihat beberapa package yang nantinya akan diimplementasikan dalam bentuk library. Package WorkshopDataAccess akan berinteraksi langsung dengan DBMS lewat pemanggilan Stored Procedure, kemudian package WorkshopBusinessLogic akan berinteraksi dengan package WorkshopDataAccess dengan memanggil method-method yang dimiliki oleh kelas yang terdapat dalam package tersebut, dan akhirnya package WorkshopBusinessLogic akan dikonsumsi oleh package WorkshopUI.

Langkah Pembuatan Aplikasi

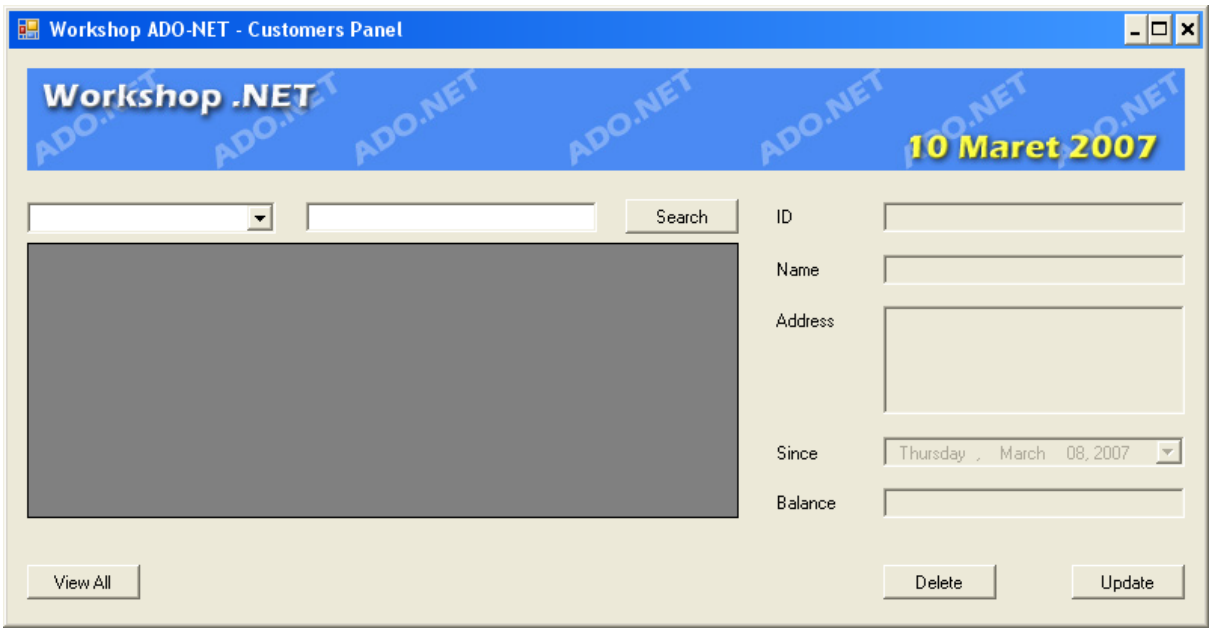
- Berdoa terlebih dahulu ;-)
- Buka Solution *Workshop-ADONET* dalam folder Step 1 yang terdapat dalam HOL Kit.
- Dalam solution tersebut sudah terdapat sebuah project dengan nama *WorkshopUI* yang bisa dilihat dalam jendela *Solution Explorer* seperti pada gambar berikut ini :



Gambar 2. Solution Explorer

- Coba jalankan project tersebut dengan menekan tombol F5 atau dari menu Debug → Start Debugging.
- Setelah dijalankan terlihat sebuah jendela dengan sebuah menu Task dimana dalam menu tersebut terdapat dua buah menu yaitu *Add New Customer* dan *Customers Panel.*
- Menu *Add New Customer* apabila diklik akan keluar tampilan untuk menambahkan Customer baru, sedangkan jika yang diklik adalah menu *Cutomers Panel* maka akan keluar tampilan pengelolaan Customer. Agar lebih jelasnya perhatikan gambar dibawah ini :

Gambar 3. Tampilan Jendela Insert New Customer



Gambar 4. Tampilan Jendela Customers Panel

- Sekarang, sudah terbayangkan kita mau buat aplikasi apa? Jika sudah sekarang kita mulai kerja yuk ;-)

Pembuatan Stored Procedure

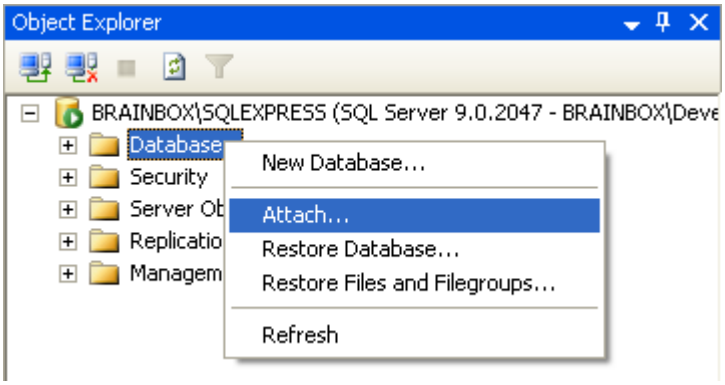
Kita tinggalkan semetera Visual Studio 2005. Dalam langkah ini kita akan membuat beberapa stored procedure yang akan dipakai dalam aplikasi yang akan kita buat. Karena jatah sesi ini "hanya" dua jam, maka kita akan memakai database yang sudah disiapkan dalam Worskhop Kit.

Yang harus kita lakukan pertama kali adalah "menanam" database tersebut dalam DBMS SQL Server 2005. Berikut ini adalah langkah-langkah-nya :

1. Buka SQL Server 2005 Express Edition dari Windows Start → All Programs → Microsoft SQL Server 2005 → SQL Server Management Studio Express.
2. Akan keluar jendela seperti gambar dibawah ini :

Gambar 5. Tampilan Jendela SQL Server Login

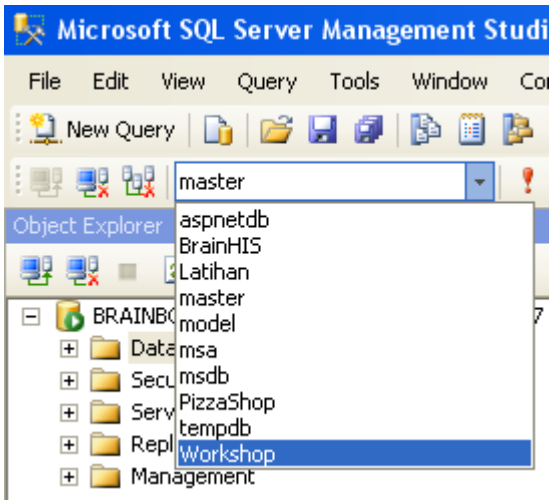
3. Sesuaikan isian *Server name* dengan *namahost\SQLEXPRESS*. Dimana namahost merupakan *hostname* dari komputer yang kita pakai. Setelah itu pastikan kita memakai *Authentication* berupa *Windows Authentication.* Setelah kita yakin, kita sudah benar melakukan langkah-langkah tersebut maka kita pilih tombol *Connect.*
4. Setelah kita berhasil login, kemudian klik kanan pada *Object Explorer* kemudian pilih menu *Attach*. Untuk lebih jelasnya lihat gambar dibawah ini :



Gambar 6. Tampilan Menu *Attach* Database

5. Pada jendela *Attach Databases* klik tombol *Add* yang akan menampilkan jendela *Locate Database Files.* Dalam jendela ini pilih File *Workshop.mdf* yang ada dalam HOL Kit. Setelah file ditemukan tekan tombol OK dan tekan tombol OK sekali lagi.
6. Sampai kita sudah berhasil "menanamkan" database kedalam sebuah DBMS. Untuk selanjutnya kita akan mulai untuk menulis *Stored Procedure.*

7. Dalam jendela *Microsoft SQL Server Management Studio Express* klik icon *New Query* yang terdapat persis dibawah menu *File*. Setelah itu pilih database *Workshop* pada comboBox *Available Databases*. Untuk lebih jelasnya kita lihat gambar dibawah ini :



Gambar 7. Pemilihan Database *Workshop*

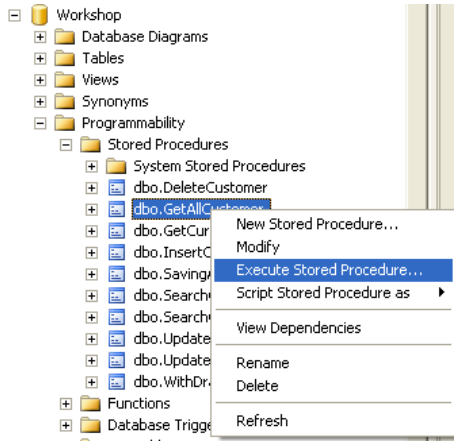
8. Dalam jendela Query kita tuliskan beberapa *Stored Procedure* seperti dibawah ini :

```
CREATE PROCEDURE [dbo].[InsertCustomer]
	@cName varchar(50),
	@cAddress varchar(100),
	@cBalance int,
	@cID int OUTPUT
AS
	SELECT @cID = MAX(cID)+1
		FROM Example
	INSERT INTO Example(cName, cAddress, cSince,cBalance)
	VALUES(@cName,@cAddress,GETDATE(),@cBalance)
GO
CREATE PROCEDURE [dbo].[GetAllCustomer]
AS
	SELECT cID, cName, cAddress, cSince, cBalance
	FROM Example
	WHERE cIsActive = 1

GO

CREATE PROCEDURE [dbo].[UpdateCustomer]
	@cName varchar(50),
	@cAddress varchar(100),
	@cID int
AS
	UPDATE Example
	SET
		cName=@cName,
		cAddress= @cAddress
	WHERE cID = @cID

GO

CREATE PROCEDURE [dbo].[DeleteCustomer]
	@cID int
AS
	UPDATE Example
	SET
		cIsActive = 0
	WHERE cID=@cID
```

9. Setelah kode diatas selesai diketikan, kita bisa menjalankan query tersebut dengan tombol F5 atau dari menu Query → Execute.

SARAN sebaiknya anda mengetikan masing-masing bagian stored procedure tersebut satu-satu. Setelah anda mengetikan satu bagian (misalnya procedure *InsertCustomer*) langsung eksekusi query tersebut. Hal ini akan meminimalisasi jumlah kesalahan yang mungkin akan terjadi ☺

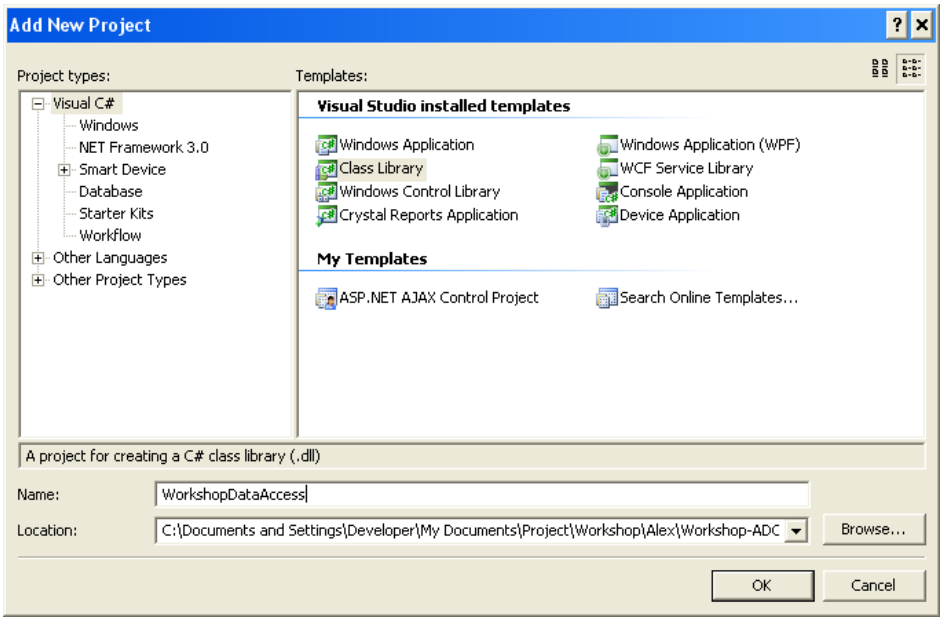
10. Ada baiknya anda mencoba masing-masing *Stored Procedure* yang sudah anda buat, untuk memastikan bahwa semua akan berjalan sesuai dengan apa yang kita inginkan. Untuk mencoba menjalankan *Stored Procedure* yang sudah kita buat, dalam *Object Explorer* kita *expand* database workshop, kemudian kita *expand* lagi pada *tree programmability*, lalu kita *expand* lagi pada *tree Stored Procedure*. Disana akan terlihat beberapa *Stored Procedure* yang sudah kita buat sebelumnya. Untuk menjalankan tinggal klik kanan nama *Stored Procedure* tersebut kemudian kita pilih menu *Execute Stored Procedure*. Untuk lebih jelasnya kita lihat gambar dibawah ini :



Gambar 8. Cara Menjalankan *Stored Procedure*

Desain dan Pemrograman DataAccess Layer

1. Kita kembali pada Visual Studio 2005 yang sudah kita buka sebelumnya.
2. Kita tambahkan project baru dengan cara mengklik kanan *solution Worskhop-ADONET* dalam *solution explorer*. Setelah itu kita pilih menu *Add* → *New Project*, dimana kemudian akan ditampilkan jendela *Add New Project*. Pada jendela ini kita pilih tipe project *Class Library* pada bagian *Visual Studio installed templates*, kemudian kita isikan "WorkshopDataAccess" pada bagian nama project. Untuk lebih jelasnya kita lihat gambar dibawah ini :



Gambar 9. Menambahkan Project Baru

3. Setelah kita yakin apa yang telah kita lakukan sudah benar, maka kita tekan tombol OK.
4. Dalam *Solution Explorer* akan muncul sebuah project baru dengan nama *WorkshopDataAccess*. Dalam project baru tersebut kita klik kanan object *Class1.cs* kemudian kita pilih menu *Rename*. Dalam kotak isian rename kita isikan nama *DataAccess.cs*.
5. Dalam jendela *DataAccess.cs* kita tambahkan direktif berikut :

```
using System.Data;
using System.Data.SqlClient;
```

6. Masih dalam jendela *DataAccess.cs* kita ketikan kode program dibawah ini didalam kelas DataAccess :

```
private string connectionString;
private SqlConnection conn;
public DataAccess()
{
      connectionString = @"Data Source=.\SQLExpress;Initial
Catalog=Workshop;Integrated Security=True";
      conn = new SqlConnection(connectionString);
}
```

```
public int InsertNewCustomer(string _name, string _address)
{
      try
      {
            conn.Open();
            SqlCommand cmd = new SqlCommand("InsertCustomer",
            conn);
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.Add(new SqlParameter("@cName",
            SqlDbType.VarChar));
            cmd.Parameters["@cName"].Value = _name;
            cmd.Parameters.Add(new SqlParameter("@cAddress",
            SqlDbType.VarChar));
            cmd.Parameters["@cAddress"].Value = _address;
            cmd.Parameters.Add(new SqlParameter("@cID",
            SqlDbType.Int, 0, ParameterDirection.Output, false, 0,
            0, "cID", DataRowVersion.Default, null));
            cmd.UpdatedRowSource =
            UpdateRowSource.OutputParameters;
            cmd.ExecuteNonQuery();
            return (int)cmd.Parameters["@cID"].Value;
      }
      catch (Exception)
      {
            return 0;
      }
      finally
      {
            conn.Close();
      }
}
```

7. Kita tambahkan lagi sebuah method yang akan digunakan untuk merubah data *Customer*, seperti pada kode dibawah ini :

```
public int UpdateCustomer(int _id, string _name, string _address)
{
      try
      {
            conn.Open();
            SqlCommand cmd = new SqlCommand("UpdateCustomer",
            conn);
            cmd.Parameters.Add(new SqlParameter("@cName",
            SqlDbType.VarChar));
            cmd.Parameters["@cName"].Value = _name;
            cmd.Parameters.Add(new SqlParameter("@cAddress",
            SqlDbType.VarChar));
            cmd.Parameters["@cAddress"].Value = _address;
            cmd.Parameters.Add(new SqlParameter("@cID",
            SqlDbType.Int));
            cmd.Parameters["@cID"].Value = _id;
            return cmd.ExecuteNonQuery();
      }
      catch (Exception ex)
      {
            return 0;
      }
      finally
      {
            conn.Close();
      }
}
```

8. Kita tambahkan lagi method yang akan digunakan untuk menghapus data *Customer*.

```
public int DeleteCustomer(int _id)
{
      try
      {
            conn.Open();
            SqlCommand cmd = new SqlCommand("DeleteCustomer",
            conn);
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.Add(new SqlParameter("@cID",
            SqlDbType.Int));
            cmd.Parameters["@cID"].Value = _id;
            return cmd.ExecuteNonQuery();
      }
      catch (Exception ex)
      {
            return 0;
      }
      finally
      {
            conn.Close();
      }
}
```

9. Kita tambahkan lagi method yang akan digunakan untuk mengambil semua data *Customers*.

```
public SqlDataReader GetAllCustomer()
{
      try
      {
            conn.Open();
            SqlCommand cmd = new SqlCommand("GetAllCustomer",
            conn);
            cmd.CommandType = CommandType.StoredProcedure;
            SqlDataReader reader = cmd.ExecuteReader();
            return reader;
      }
      catch (Exception)
      {
            return null;
      }
}
```

10. Jika semua kode program diatas sudah selesai kita ketikan, maka selanjutnya kita coba build project tersebut. Untuk melakukan hal tersebut, caranya adalah dengan mengklik kanan nama project pada *solution explorer* kemudian kita pilih menu build. Jika hasil build ini sukses (tidak ada ERROR) maka program kita sudah benar secara *syntax*, tapi jika hasil build tersebut belum sukses, maka kita perlu untuk memperbaiki kesalahan *syntax* terlebih dahulu.

Desain dan Pemrograman Business Logic

1. Business logic ini akan kita gunakan untuk menghubungkan antara Data Access Layer dan antarmuka. Kita akan mengimplementasikan *business logic ini* dalam sebuah project class library. Cara pembuatan project ini hampir sama dengan cara pembuatan project sebelumnya, hanya saja kita ganti nama project menjadi *WorkshopBusinessLogic.*
2. Dalam project *WorkshopBusinessLogic* tersebut kita ganti nama object *Class1.cs* menjadi *Customer.cs.*
3. Seperti pada gambar 1 diatas, Library *WorkshopBusinessLogic* ini akan memakai library dari *WorkshopDataAccess.* Maka agar kelas ataupun method yang telah didefinisikan dalam library *WorkshopDataAccess* dapat dipakai, maka kita harus menambahkan reference terlebih dahulu. Adapun cara untuk menamahkan reference adalah dengan mengklik kanan nama project *WorkshopBusinessLogic* kemudian kita klik menu *Add Reference.*
4. Pada jendela *Add Reference* kita pilih tab *Projects,* kemudian kita pilih nama project *WorkshopDataAccess* dan kita tekan tombol OK.
5. Setelah *reference* berhasil ditambahkan, maka kita tambahkan beberapa direktif seperti dibawah ini dalam jendela *BusinessLogic.cs :*

```
using System.Data.SqlClient;
using WorkshopDataAccess;
```

6. Selanjutnya kita ketikan kode dibawah ini didalam kelas *Customer*:

```
private int iD;
private string name;
private string address;
private DateTime since;
private int balance;

public int ID
{
      get { return iD; }
}
public string Name
{
      get { return name; }
     set { name = value; }
}
public string Address
{
      get { return address; }
     set { address = value; }
}
public DateTime Since
{
      get { return since; }
}
public int Balance
{
      get { return balance; }
}
```

7. Kita tambahkan lagi tiga buah kunstruktor seperti kode dibawah ini :

```
public Customer(string _name, string _address, DateTime _since,
int _balance)
{
	this.name = _name;
	this.address = _address;
	this.since = _since;
	this.balance = _balance;
}
public Customer(int _id, string _name, string _address, DateTime
_since, int _balance)
{
	this.iD = _id;
	this.name = _name;
	this.address = _address;
	this.since = _since;
	this.balance = _balance;
}
public Customer(int _id)
{
	this.iD = _id;
}
```

8. Kemudian kita tambahkan beberapa method, seperti pada kode dibawah ini :

```
public int Add()
{
	return new DataAccess().InsertNewCustomer(this.name,
	this.address, this.balance);
}
public int Update()
{
	return new DataAccess().UpdateCustomer(this.iD, this.name,
	this.address);
}
public int Delete()
{
	return new DataAccess().DeleteCustomer(this.iD);
}
```

9. Kita tambahkan lagi method *GetAllCustomers* :

```
public static List<Customer> GetAllCustomers()
{
	SqlDataReader reader = new DataAccess().GetAllCustomer();
	List<Customer> listCustomer = new List<Customer>();
	while (reader.Read())
	{
		try
		{

			Customer customer =
			new Customer(Convert.ToInt32(reader["cID"].ToString()),
			reader["cName"].ToString(),
			reader["cAddress"].ToString(),
			Convert.ToDateTime(reader["cSince"].ToString()),
			Convert.ToInt32(reader["cBalance"].ToString()));
			listCustomer.Add(customer);
		}
		catch (Exception ex)
		 {
		 }
	}
	reader.Close();
	return listCustomer;
}
```

10. Kita tambahkan method ToString yang akan mengoveride kelas object menjadi seperti dibawah ini :

```
public override string ToString()
{
      return "ID\t:" + this.iD + "\n" +
            "Name\t:" + this.name + "\n" +
            "Address\t:" + this.address + "\n" +
            "Balance\t:" + this.balance;
}
```

Setelah semua kode program diatas selesai dituliskan, maka kita build terlebih dahulu project tersebut. Jika build sudah sukses maka sampai disini kita sudah selesai membuat *Business Logic layer* dari aplikasi kita.

Desain dan Pemograman User Interface

1. Sampai disini, kita akan mulai untuk menghubungkan antara *User Interface* dan *Business Logic*. Untuk melakukan hal tersebut, kita kembali ke project *WorkshopUI.* Dalam project tersebut kita tambahkan *Reference* yang caranya sama dengan langkah sebelumnya, kita pilih *WorkshopBusinessLogic* sebagai *reference.*
2. *Double object* CustomersUI.cs dalam project *WorkshopUI* yang terdapat dalam *Solution Explorer.*
3. Klik dua kali pada button dengan text *Display All (buttonDisplayAll)*. Kemudian tambahkan kode, sehingga event *Click* dari button tersebut menjadi seperti dabawah ini :

```
private void buttonDisplayAll_Click(object sender, EventArgs e)
{
      dataGridViewCustomers.DataSource =
      Customer.GetAllCustomers();
      dataGridViewCustomers.Refresh();
}
```

4. Kemudian kita kembali ke *Design View* dari object *CustomersUI*, lalu kita pilih object *dataGridViewCustomers,* kemudian dalam jendela *Properties*, kita pilih event *CellClick* dan klik dua kali pada nama event tersebut.
5. Kita tambahkan kode pada event *CellClick* dari object *dataGridViewCustomers*, sehingga kodenya menjadi seperti dibawah ini :

```
private void dataGridViewCustomers_CellClick(object sender,
DataGridViewCellEventArgs e)
{
        try
       {
                textBoxID.Text = dataGridViewCustomers["id",
                e.RowIndex].Value.ToString();
                textBoxName.Text = dataGridViewCustomers["name",
                e.RowIndex].Value.ToString();
                textBoxAddress.Text = dataGridViewCustomers["address",
                e.RowIndex].Value.ToString();
                dateTimePickerSince.Value =
                Convert.ToDateTime(dataGridViewCustomers["since",
                e.RowIndex].Value.ToString());
                textBoxBalance.Text = dataGridViewCustomers["balance",
                e.RowIndex].Value.ToString();
        }
       catch (Exception)
       {
                //TO:DO
       }
}
```

6. Kita kembali lagi ke *CustumersUI.cs Design View*, kemudian kita klik dua kali pada *button Update (buttonUpdate)*, lalu kita tambahkan kode sehingga menjadi seperti dibawah ini :

```
private void buttonUpdate_Click(object sender, EventArgs e)
{
       Customer customer = new
       Customer(Convert.ToInt32(textBoxID.Text));
       customer.Name = textBoxName.Text;
       customer.Address = textBoxAddress.Text;
       if (customer.Update() == 1)
              MessageBox.Show("Customer Information :\n" +
              customer.ToString()+
              "\nWas Changed", "Information", MessageBoxButtons.OK,
              MessageBoxIcon.Information);
       else
              MessageBox.Show("Customer Information :\n = " +
              customer.ToString() +
              "\nCan't Be Update", "Information",
              MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

7. Dengan cara yang hampir sama dengan diatas, kita klik dua kali pada tombol *Delete (buttonDelete)*, lalu kita tambahkan kode sehingga menjadi seperti dibawah ini :

```
private void buttonDelete_Click(object sender, EventArgs e)
{
       Customer customer = new Customer(Convert.ToInt32(textBoxID.Text));
      customer.Name = textBoxName.Text;
      customer.Address = textBoxAddress.Text;
      if (customer.Delete() == 1)
              MessageBox.Show("Customer Information :\n" +
              customer.ToString()+
              "\nWas Deleted", "Information", MessageBoxButtons.OK,
              MessageBoxIcon.Information);
       else
              MessageBox.Show("Customer Information :\n = " +
              customer.ToString() +
              "\nCan't Be Deleted", "Information", MessageBoxButtons.OK,
              MessageBoxIcon.Information);
}
```

8. Selanjutnya, dalam *Solution explorer*, dalam project *WorkshopUI* kita klik dua kali pada *AddCustomerUI.cs*, dalam *Design View* kita klik dua kali pada tombol *Insert (buttonInsert)*, lalu tambahkan kode sehingga menjadi seperti dibawah ini :

```
private void buttonInsert_Click(object sender, EventArgs e)
{
        Customer customer = new Customer();
        customer.Name = textBoxName.Text;
        customer.Address = textBoxAddress.Text;
        customer.Balance = textBoxBalance.Text;
        customer.ID = customer.Add();
        if (customer.ID != 0)
                MessageBox.Show(customer.ToString());
        else
                MessageBox.Show("Gagal");
}
```

9. Sampai disini, kita coba build project kita, jika build berhasil dengan sukses, maka kita coba dahulu untuk menjalankan aplikasi yang sudah kita buat. Untuk mencoba menjalankan aplikasi ini, kita tekan F5 atau dari menu Debug → Start Debugging.
10. Setelah dijalankan terlihat pada jendela *Customers Panel* tampilan data tidak urut, dimana kolom *ID* tidak terletak pada kolom pertama. Untuk membuat kolom tersebut urut, maka kita perlu untuk membuat *custom* kolom. Untuk melakukan hal tersebut, kita keluar terlebih dahulu dari mode *debugging.*
11. Pada *Design View* pada object *CustomersUI.cs* kita pilih object *dataGridViewCustomers*, lalu dalam jendela *Properties*, kita pilih properti *Columns*, lalu kita klik tombol kecil disebelah tulisan (*Collection).*
12. Akan tampil jendela, *Edit Columns*, kita tekan tombol *Add*, setelah tombol *Add* tersebut ditekan akan tampil jendela *Add Column.*
13. Dalam jendela ini, kita tambahkan kolom sebagai berikut ini :

Name	Type	Header Text
ID	DataGridViewTextBoxColumn	ID
Name	DataGridViewTextBoxColumn	Name
Address	DataGridViewTextBoxColumn	Address
Since	DataGridViewTextBoxColumn	Since
Balance	DataGridViewTextBoxColumn	Balance

14. Setelah semua kolom berhasil ditambahkan kita tekan tombol Close pada jendela *Add Column.*
15. Selanjutnya kita akan mengatur tampilan dari masing-masing kolom dan mengatur juga asal data dari masing-masing kolom. Untuk melakukan hal tersebut, masih dalam

jendela *Edit Columns*, kita atur properti dari masing-masing kolom sesuai dengan nilai dibawah ini :

Column	DataPropertyName	Width
ID	Id	30
Name	name	100
Address	address	130
Since	since	78
Balance	balance	91

16. Jika sudah selesai, coba jalankan aplikasi yang telah kita buat tersebut.

Sampai disini dulu Tutorial kita hari ini, dalam tutorial ini kita sudah belajar, bagaimana membuat stored procedure, memakai beberapa pustaka dari ADO.NET dan menampilkan data dalam sebuah control DataGridView, semoga tulisan ini bisa bermanfaat bagi kita semua, dan sampai ketemu di tutorial yang akan datang.

Alex Budiyanto, MCTS Menamatkan SMU di SMU Tiga Maret Yogyakarta pada tahun 2003 sebagai lulusan terbaik. Saat ini masih tercatat sebagai mahasiswa aktif di Teknik Informatika - Universitas Atma Jaya Yogyakarta. Aktif di berbagai komunitas seperti IlmuKomputer.Com, KPLI-Jogja, INDC dan berbagai komunitas IT lainnya. Pernah menjabat sebagai koordinator Kelompok Study Linux - Universitas Atma Jaya Yogyakarta dari tahun 2004 - 2006. Berpengalaman sebagai System Administrator Untuk Mail Server, Web Server, dan DNS Server di Fakultas Teknologi Industri - Universitas Atma Jaya Yogyakarta sejak tahun 2004. Sejak tahun 2005 dipercaya menjadi Asisten Dosen untuk mata kuliah Praktikum Sistem Operasi dan Jaringan di Universitas yang sama.

MCTS merupakan sertifikasi internasional pertama yang didapatkan dari Microsoft. Terhitung sejak January 2007 menjabat sebagai Microsoft Students Ambasador untuk Universitas Atma Jaya Yogyakarta. Sangat tertarik pada bidang Software Enginering, Object Oriented Programming, Rational Database dan Networking. Mendalami bahasa pemrograman C/C++, C# dan Java. Selain itu mendalami juga database seperti : SQL-Server dan Oracle. Saat ini menjadi Trainer ASP.NET dan Linux di Brainmatics.Com maupun IlmuKomputer.Com.

Informasi lebih lanjut bisa didapatkan di :

- Friendster : http://friendster.com/abudiyanto
- Flickr : http://flickr.com/photos/alexbudiyanto/
- Email : alex[dot]budiyanto[at]gmail.com
- YM! : alex.budiyanto