IlmuKomputer.Com
Ikatlah ilmu dengan menuliskannya

Asynchronous Programming (Async dan Await) pada C# 6.0

Junindar, ST, MCPD, MOS, MCT, MVP .NET
junindar@gmail.com

http://junindar.blogspot.com

Abstrak

Didalam pemograman dikenal dengan dua buah teknik dalam membangun aplikasi yaitu *Synchronous* dan *Asynchronous*. Kedua fungsi ini harus digunakan sebaik-baik mungkin untuk mendapatkan hasil yang maksimal. *Synchronous* adalah dimana sebuah proses akan dilanjutkan ke proses selanjutnya setelah proses sebelumnya selesai. Sedangkan *Asynchronous* dimana proses selanjutnya dapat langsug dikerjakan tanpa harus menunggu proses sebelumnya selesai.

Pendahuluan

Didalam pemograman dikenal dengan dua buah teknik dalam membangun aplikasi yaitu *Synchronous* dan *Asynchronous*. Kedua fungsi ini harus digunakan sebaik-baik mungkin untuk mendapatkan hasil yang maksimal. *Synchronous* adalah dimana sebuah proses akan dilanjutkan ke proses selanjutnya setelah proses sebelumnya selesai. Sedangkan *Asynchronous* dimana proses selanjutnya dapat langsug dikerjakan tanpa harus menunggu proses sebelumnya selesai.

Didalam artikel ini akan dijelaskan bagaimana menggunakan *Asynchronous* dalam pemograman dengan C# sebagai bahasa pemograman nya. Pada C# 6.0 terdapat fungsi *async* dan *await* sehingga memudahkan kita dalam membuat pemograman *Asynchronous.*

Isi

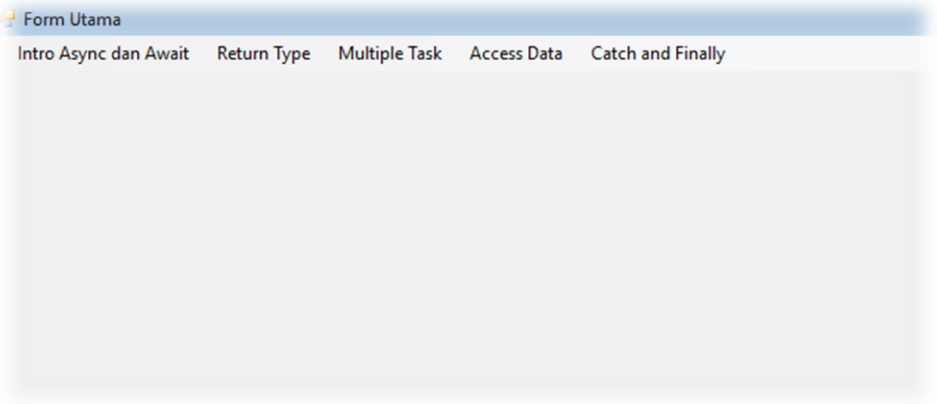
Untuk memudahkan memahami isi dari artikel ini, kita akan membuat sebuah project latihan, dimana kita akan menggunakan *async* dan *await* . Ikuti langkah-langkah dibawah ini.

1. Buat sebuah project dengan nama “Latihan Asynchronous”.
2. Ganti properties pada Form1 menjadi seperti berikut.

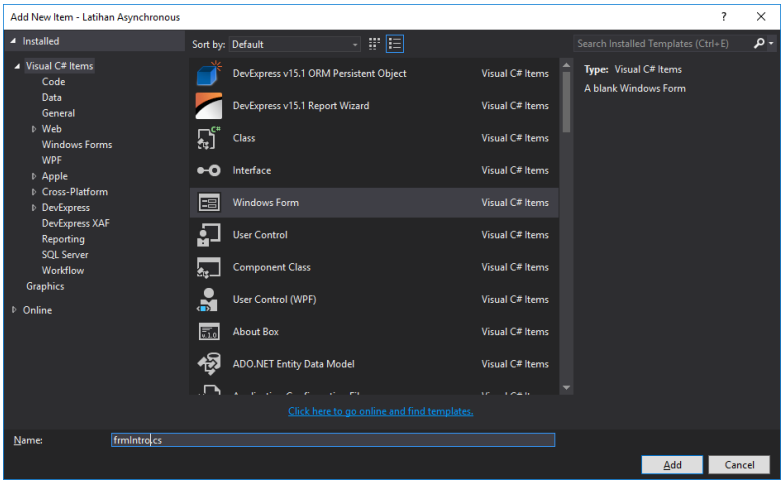
   Name = frmUtama

   Text = Form Utama

   StartPosition = CenterScreen
3. Tambahkan Menustrip pada form, lalu tambahkan 5 (lima) menu item seperti berikut.



4. Tambah sebuah form dengan nama “frmIntro”.



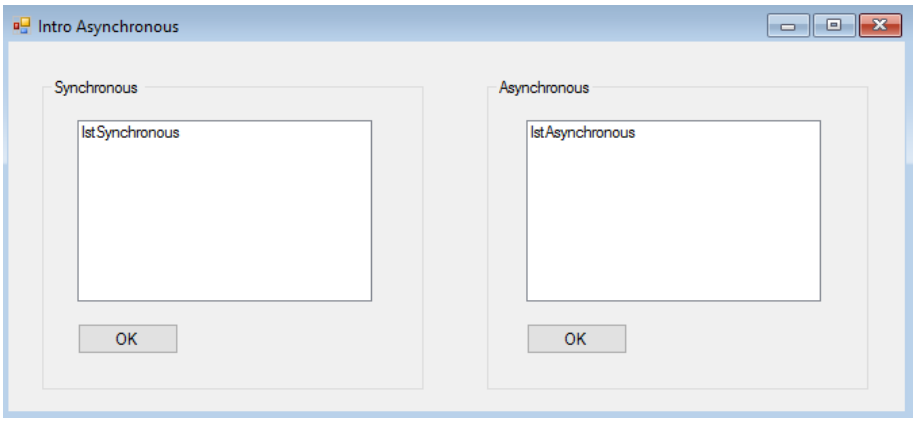
5. Tambahkan beberapa control pada frmIntro, sebelumnya ganti text form menjadi **Intro Asynchronous.**

   - 2 buah ListBox dengan masing-masing properties name sebagai berikut.

lstSynchronous dan **lstAsynchronous**

- 2 buah GroupBox dengan masing-masing text **Synchronous** dan **Asynchronous**
- 2 buah button dengan nama **btnSynchronous**dan text nya **OK.** Dan btnAsynchronous dan text nya juga OK.

Susun control-control tersebut seperti gambar dibawah.



Pada latihan ini kita akan mengggunakan *Task*. *Task* adalah kelas yang digunakan untuk melakukan pekerjaan *asynchronous*. Hampir sama dengan *Thread*, tetapi pengunaan *Task* sangat mudah jika dibandingkan dengan *Thread*. Dan jika *Task* berjalan bersamaan disebut *Task Pararelism*. Buka jendela Code untuk form diatas. Dan Tambahkan sebuah task seperti dibawah.

```
private  Task LongProcessing()
       {
          return Task.Run(() =>
          {
             System.Threading.Thread.Sleep(5000);
        });
```

Pada task diatas dapat dilihat, dimana kita menggunakan Thread.Sleep selama 5 detik. Proses 5 detik ini kita asumsikan sebagai sebuah proses yang memakan waktu lama. Dan selanjutnya adalah membuat sebuah asynchronous method untuk memanggil Task diatas. Ditambah dengan sintaks untuk menambahkan item pada ListBox. Sintaks lengkapnya dapat dilihat dibawah ini.

```
private async void AsynchronousProcess()
{
        await LongProcessing();
        lstAsynchronous.Items.Add("Asynchronous Process");
 }
```

Perhatikan sintaks diatas, kita menggunakan *await* untuk memanggil Task yang telah dibuat diatas. Keyword await digunakan untuk "menunggu" proses selesai. Berbeda dengan method biasa, dengan menggunakan async-await maka aplikasi yang dibuat tidak akan *blocking* selama proses berjalan.

Dan untuk membandingkannya, kita akan membuat method biasa (*synchronous*) yang fungsi nya sama seperti method *asynchronous* diatas.

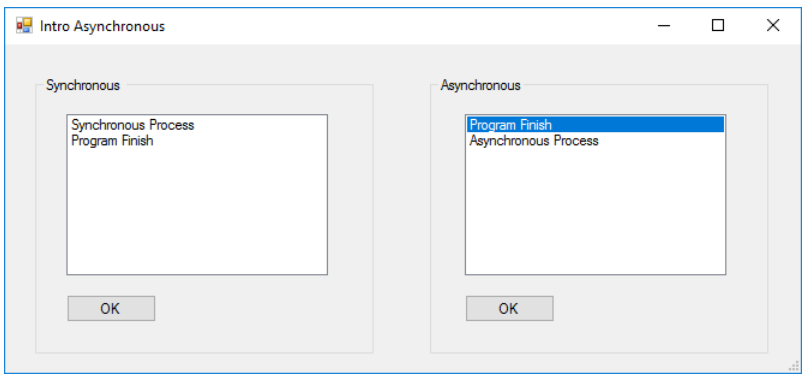
```
 private void SynchronousProcess()
{
        System.Threading.Thread.Sleep(5000);
        lstSynchronous.Items.Add("Synchronous Process");
 }
```

Klik ganda btnSynchronous dan ketikkan sintaks dibawah.

```
SynchronousProcess();
lstSynchronous.Items.Add("Program Finish");
```

Selanjutnya klik ganda btnAsynchronous dan ketikkan sintaks dibawah.

```
AsynchronousProcess();
lstAsynchronous.Items.Add("Program Finish");
```



Jalankan program dan klik kedua button pada form. Untuk button Synchronous setelah kita klik, maka form menjadi tidak responsive (*freezing*) sampai prosesnya selesai dan eksekusi proses nya pun berurutan (menunggu proses sebelumnya selesai). Sedangkan untuk button *Asynchronous* pada saat aplikasi sedang menjalankan proses, maka antara muka pun tidak akan terganggu. Selanjutnya eksekusi proses tidak perlu menunggu proses sebelumnya selesai. Sehingga proses dapat berjalan secara parallel dan akan membuat aplikasi menjadi lebih cepat.

Pada latihan kedua ini, akan dijelaskan Return Type dari Asynchronous. Seperti method biasa, pada Asynchronous juga bisa memiliki return (nilai kembalian) sesuai dengan tipe data yang telah di set.

Tambahkan sebuah form selanjutnya tambahkan control-control seperti pada latihan sebelumnya. Pada jendela Code buat sebuah static Task seperti dibawah.

```
private static Task<string> LongProcessing()
    {
        return Task.Run(() =>
        {
            System.Threading.Thread.Sleep(5000);
            return "Asynchronous Process";
        });
    }
```

Dapat dilihat pada sintaks diatas, Return Type dari Task yang dibuat adalah “String”. Sedangkan untuk mendapat nilai dari task tersebut, ketikkan sintaks dari Asynchronous method dibawah ini.

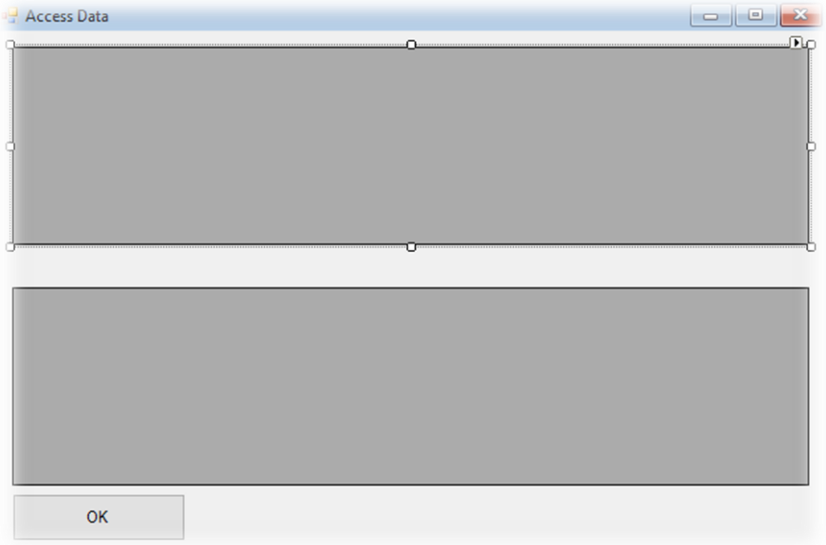
```
private async void AsynchronousProcess()
    {
        string value = await LongProcessing();
        lstAsynchronous.Items.Add(value);
    }
```

Dapat dilihat dimana nilai dari Task akan dimasukkan kedalam ListBox.

Jalankan program untuk melihat hasil yang telah kita buat.

Untuk latihan ketiga kita akan membuat simulasi dalam mengambil data dan memasukkannya kedalam DataGridView. Ikuti langkah-langkah dibawah ini.

Tambahkan sebuah form dan tambahkan beberapa control dan susun seperti gambar dibawah.



Pada jendela Code, tambahkan sebuah class dengan nama Siswa.

```
public class Siswa
    {
        public string Nim { get; set; }
        public string Nama { get; set; }
        public string Kota { get; set; }
    }
```

Buat sebuah method untuk mengisi data pada Generic List<Siswa>.

```
private List<Siswa> FillData()
    {
        return new List<Siswa>()
        {
            new Siswa(){Nim = "001",Nama = "Junindar",Kota = "Batam"},
            new Siswa(){Nim = "002",Nama = "Ahmad",Kota = "Tg Pinang"},
            new Siswa(){Nim = "003",Nama = "Andik",Kota = "Tg Balai"}
        };

    }
```

Selanjutnya tambahkan dua buah Task dimana Return Type nya adalah List<Siswa>

```
public Task<List<Siswa>> LoadData1()
        {
            return Task.Run(() =>
            {
                System.Threading.Thread.Sleep(5000);
                return FillData();
            });
        }

 public Task<List<Siswa>> LoadData2()
        {
            return Task.Run(() =>
            {
                System.Threading.Thread.Sleep(5000);
                return FillData();
            });
        }
```

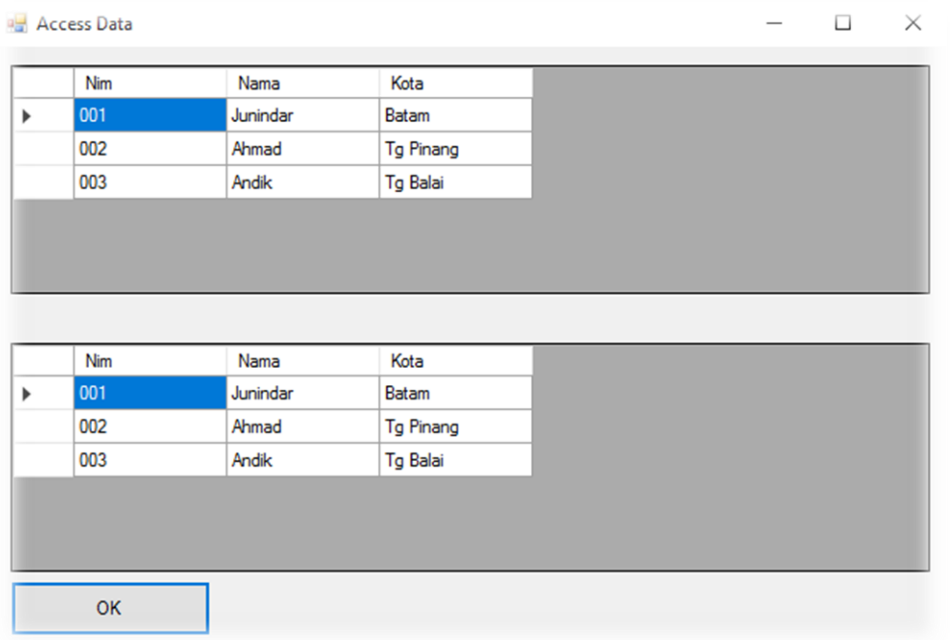
Lalu klik ganda button OK. Tambah async pada event handler.

```
private async void btnOK_Click(object sender, EventArgs e)
```

Dan panggil dua Task diatas seperti sintaks dibawah.

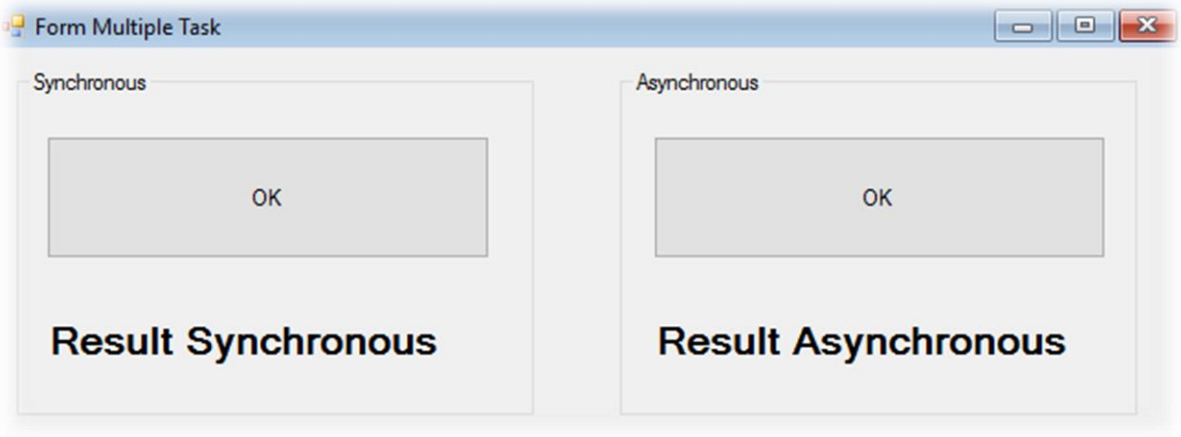
```
dataGridView1.DataSource = await LoadData1();
dataGridView2.DataSource = await LoadData2();
```

Selanjutnya jalankan program untuk melihat hasil nya. Pada saat proses masih berlangsung, user interface masih bisa berinteraksi, seperti resize form dan lain-lain.



Latihan ke empat adalah membuat Multi Task dan di eksekusi secara parallel (secara bersamaan). Pada latihan ini kita akan melakukan perbandingan antara cara konvensional dengan cara parallel.

Tambah sebuah form tambah control-control pada form lalu design seperti gambar dibawah.



Buka jendela code, selanjutnya buat 3 buah method seperti dibawah.

```
 private void Result1()
{
        int n = 500000000;
        int f = 1;
        for (int i = 1; i <= n; ++i)
            f *= i;
 }


 private void Result2()
 {
        int n = 500000000;
        int f = 1;
        for (int i = 1; i <= n; ++i)
            f += i;
}

 private void Result3()
{
        int n = 500000000;
        int f = 1;
        for (int i = 1; i <= n; ++i)
            f -= i;
}
```

Selanjutnya adalah dengan menambahkan 3 buah task seperti dibawah.

```
private static  Task LongProcessing1()
 {
        int n = 500000000;
        return Task.Run(() =>
        {
            int f = 1;
            for (int i = 1; i <= n; ++i)
                f *= i;

        });
 }
```

```
private static Task LongProcessing2()
{
        int n = 500000000;
        return Task.Run(() =>
        {
           int f = 1;
           for (int i = 1; i <= n; ++i)
              f += i;

        });
 }

private static Task LongProcessing3()
{
        int n = 500000000;
        return Task.Run(() =>
        {
           int f = 1;
           for (int i = 1; i <= n; ++i)
              f -= i;

        });
}
```

Klik ganda button Synchronous dan ketikkan sintaks dibawah.

```
        Stopwatch myStopwatch = new Stopwatch();
        TimeSpan myTimeSpan = new TimeSpan();
        myStopwatch.Reset();
        myStopwatch.Start();

        Result1();
        Result2();
        Result3();
        myStopwatch.Stop();
        myTimeSpan = myStopwatch.Elapsed;

        lblResultSync.Text = myTimeSpan.ToString();
```

Pada sintaks diatas, kita panggil 3 buah method diatas (Result1,Result2 dan Result3). Selanjutnya kita akan menghitung waktu yang di perlukan untuk menyelesaikan 3 buah proses tersebut. Dan waktu tersebut akan ditampilkan di label ResultSync.

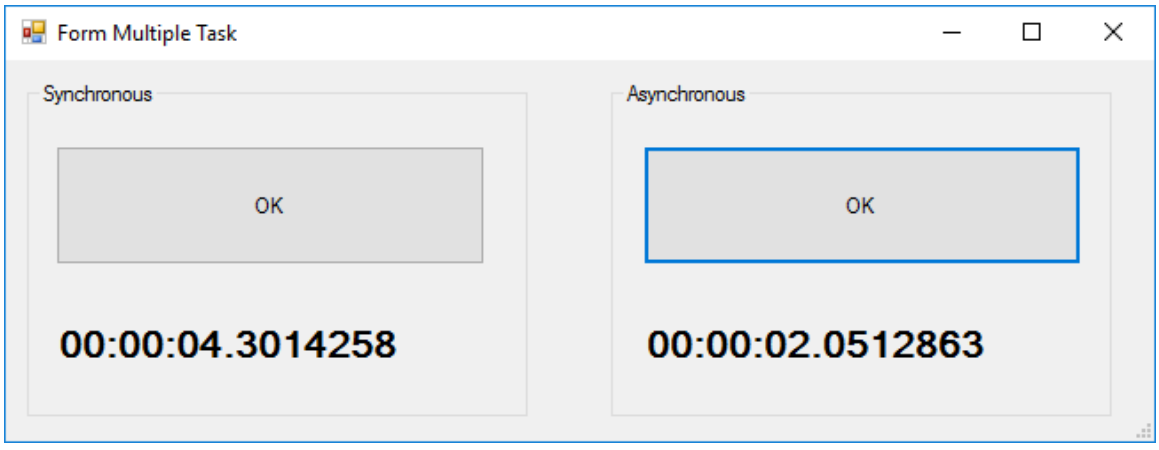
Untuk Asynchronous, klik ganda button Asynchronous dan ketikkan sintaks dibawah.

```
Stopwatch myStopwatch = new Stopwatch();
TimeSpan myTimeSpan = new TimeSpan();
myStopwatch.Reset();
myStopwatch.Start();

var allTask = Task.WhenAll(LongProcessing1(), LongProcessing2(), LongProcessing3());
await allTask;

myStopwatch.Stop();
myTimeSpan = myStopwatch.Elapsed;

lblResultAsync.Text = myTimeSpan.ToString();
```

Sama seperti button sebelumnya, tapi disini kita akan memanggil dan menjalankan 3 buah Task secara bersamaan. Dan menunggu hingga ketiga proses tersebut selesai.

```
var allTask = Task.WhenAll(LongProcessing1(), LongProcessing2(), LongProcessing3());
 await allTask;
```

Jalankan program dan coba klik kedua button pada form. Dan kita lihat hasil yang didapat dari dua buah button tersebut.



Dapat kita lihat, penggunaan parallel jauh lebih cepat dibanding dengan cara konvensional

Penutup

Pada artikel ini telah dijelaskan bagaimana melakukan pemograman asynchronous dengan menggunakan async – await.

Sedangkan untuk memudahkan dalam memahami isi artikel, maka penulis juga menyertakan dengan full source code project latihan ini, dan dapat di download disini

http://junindar.blogspot.co.id/2016/12/asynchronous-programming-async-dan.html

Referensi

1. www.msdn.microsoft.com
2. www.planetsourcecode.com
3. www.codeproject.com
4. www.aspnet.com

Masih banyak lagi referensi yang ada di Intenet. Anda tinggal cari di www.Google.com. Dengan kata kunci "**tutorial VB.Net"**

Biografi Penulis.

Junindar Lahir di Tanjung Pinang, 21 Juni 1982. Menyelesaikan Program S1 pada jurusan Teknik Informatika di Sekolah Tinggi Sains dan Teknologi Indonesia (ST-INTEN-Bandung). Junindar mendapatkan Award Microsoft MVP VB pertanggal 1 oktober 2009 hingga saat ini. Senang mengutak-atik computer yang berkaitan dengan bahasa pemrograman. Keahlian, sedikit mengerti beberapa bahasa pemrograman seperti : VB.Net, C#, SharePoint, ASP.NET, VBA. Reporting: Crystal Report dan Report Builder. Database: MS Access, MY SQL dan SQL Server. Simulation / Modeling Packages: Visio Enterprise, Rational Rose dan Power Designer. Dan senang bermain gitar, karena untuk bisa menjadi pemain gitar dan seorang programmer sama-sama membutuhkan seni. Pada saat ini bekerja di salah satu Perusahaan Consulting dan Project Management di Malaysia sebagai Senior Consultant. Memiliki beberapa sertifikasi dari Microsoft yaitu Microsoft Certified Professional Developer (MCPD – SharePoint 2010), MOS (Microsoft Office Specialist) dan MCT (Microsoft Certified Trainer) Mempunyai moto hidup: "**Jauh lebih baik menjadi Orang Bodoh yang giat belajar, dari pada orang Pintar yang tidak pernah mengimplementasikan ilmunya".**