

# pojokprogrammer

Majalah Elektronik

- Dasar Pemrograman OOP
- Dasar Pemrograman ASP.net
- Dasar Pemrograman WPF
- Alternatif Reporting di .NET
- Komponen Data Akses Windows

Menjadi  
**Programmer**  
Handal

- 4 Menjadi Programmer Handal
- 8 Dasar Pemrograman Berorientasi Obyek
- 14 Alternatif Reporting di Lingkungan Windows
- 20 Komponen Data Akses di Lingkungan Windows
- 28 Pengenalan Pemrograman ASP.net
- 32 Pengenalan Pemrograman WPF: Windows Presentation Foundation

**Editor:** Nur Hidayat

**Contributor:** Endar Zulfahrain, Hartoto Dinata, Gege Satya Putra, Fandi Susanto, Paulus Iman

**Designer:** Maydriade Bagoes Sasongko, Nur Hidayat  
**Information is correct at press time.**

Check [www.pojokprogrammer.net](http://www.pojokprogrammer.net) for updates.

#### LISENSI

COPYRIGHT 2003-2013 POJOKPROGRAMMER.NET

Seluruh dokumen di Majalah Elektronik ini dapat digunakan, dimodifikasi dan disebarluaskan secara bebas untuk tujuan bukan komersial (non-profit), dengan syarat tidak menghapus atau mengubah atribut penulis dan pernyataan copyright yang disertakan dalam dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari penulis dan PojokProgrammer.net.

#### DISCLAIMER

Penulis dan editor PojokProgrammer.net telah berusaha dengan sungguh-sungguh dan teliti dalam mempersiapkan seluruh materi yang ada di majalah elektronik ini. Walaupun segala upaya telah kami lakukan untuk keakuratan informasi, namun tiada gading yang tak retak dan ada kemungkinan masih tetap terjadi kesalahan penulisan artikel.

Pada hakekatnya setiap penulis memiliki hak penuh terhadap artikel yang telah diterbitkan di majalah elektronik ini, dan juga memiliki hak untuk memilih lisensi yang ingin digunakan selama masih bersifat terbuka.



Berawal dari keprihatinan akan keterbatasan sumber informasi pembelajaran pemrograman berbahasa Indonesia, maka kami mencoba mengumpulkan beberapa orang yang sering kali kerepotan karena muncul pertanyaan yang sama dan sifatnya trivial berulang-ulang ditanyakan oleh member di group Facebook [Programmer VB Indonesia](#) dan [Komunitas VB.net Indonesia](#), maka akhirnya diterbitkanlah majalah Elektronik ini.

Dengan Semangat Edisi Perdana, maka Edisi ini mengangkat tema utama "Menjadi Programmer Handal" karena kemampuan inilah pasti yang diinginkan oleh semua orang yang memulai belajar pemrograman. Artikel lainnya adalah pengenalan tentang sejumlah hal yang perlu dipahami oleh seorang **Programmer Handal**, meliputi konsep OOP, sejarah data akses, reporting, serta pengenalan ASP.net dan WPF.

Semoga Majalah Elektronik ini bisa terus terbit memajukan kualitas Programmer Indonesia.

**Nur Hidayat**

*Editor in Chief*

## SPECIAL THANKS TO:

Terima kasih sebesar-besarnya kepada **Endar Zulfahrain, Hartoto Dinata, Gege Satya Putra, Fandi Susanto, Paulus Iman, Maydriade Bagoes Sasongko** atas kontribusinya berupa artikel dan grafik yang dimuat dalam Edisi Perdana ini.

Edisi Perdana

# POJOK PROGRAMMER



MENJADI  
PROGRAMMER  
HANDAL



# Menjadi Programmer Handal

Pengalaman menunjukkan bahwa seorang programmer tidak perlu menghafal seluruh sintaks bahasa pemrograman



**S**alam untuk semuanya. Sebagai artikel pembuka dengan judul yang cukup *menyeramkan* seperti judul di atas tentunya lebih enak dibaca jika ditulis dengan sudut pandang yang berbeda. Jadi, saya tulis berdasarkan pengalaman saya pribadi.

## Programmer vs Developer

Menjadi seorang programmer yang handal tentu gak semudah membalik telapak tangan, tapi bukannya gak bisa. Faktanya, saya bukanlah seorang programmer yang handal. Pernah suatu ketika, ceritanya lagi tes program untuk seleksi penerimaan staf IT, waktu itu saya dapat soal bagaimana merubah bilangan angka menjadi bilangan huruf. Itu loh, yang sering ada di aplikasi kasir atau teller bank. Merubah 1000000000 menjadi "1 milyar", misalnya. Nyatanya, sekalipun saya sudah cukup berpengalaman membuat ragam macam aplikasi, jujur aja, saya gak bisa menjawab pertanyaan itu. Oops, rewind! Bukan gak bisa, tapi malas. Ada 2 soal, dikasi waktu 1 jam. Belum dipotong waktu untuk ngelamun. Yah, akibatnya, rasa malas itu

semakin membesar. Untungnya, karena saya orangnya termasuk oportunis, saya memanfaatkan saja fasilitas internet yang sudah terhubung ke laptop untuk tes soal tersebut. Mereka gak ada ngomong "jangan pakai internet ya". So? Buka cmd, ketik ping, lihat respon. Lantas buka Bing atau Google dan viola..... I'm done!!

## Yang menjadikan Anda seorang programmer yang hebat adalah berani menghadapi masalah

Stupid bin lamer, heh? Jujur aja, dulunya saya sering "ngatain" kepada mereka yang sukanya tukang contek kode dengan sebutan lamer (script kiddies – kalau dalam dunia hacking). Tapi tunggu dulu. Sebelum Anda mencap saya dengan sebutan yang sama, hehe, faktanya apa yang saya salin secara sempurna dari internet tentang soal tersebut ternyata tidak berjalan 100%. Ketika saya tes, ada bug nya. Saya tulis 1000000

malah hasilnya "1 juta nol" dan berbagai macam keunikan lainnya. Membuat saya benar-benar tersenyum jengkel di ruangan segiempat yang sempit kala itu. Oia, mejanya berbentuk lingkaran. Gak penting juga sih.

Anyway, a good developer should have the ability to heal the world (minjem liriknya Michael Jackson). Meskipun saya mencontek dari internet, tapi toh saya bisa memperbaiki kesalahan kodenya bukan? Jadi, untuk menjadi seorang programmer yang hebat, Anda gak perlu harus menguasai semuanya. Jujur aja, kalau saya disuruh bikin fungsi terbilang itu saat ini juga dari scratch, belum tentu saya bisa menyelesaikannya dalam waktu yang cepat.

Tapi yang menjadikan Anda seorang programmer yang hebat adalah berani menghadapi masalah. Begini, pernah suatu ketika saya berniat dalam hati harus menguasai secara mendalam semua fungsi dari bahasa programming. Pada saat menerima matkul C/C++, saya terkesima dengan fungsi-fungsi seperti cout, printf, scanf,

dan lain-lain. Dentuman hati saya mengatakan “saya harus mengerti semua fungsi yang lebih dari ini”. Kemudian, saya beli buku-buku yang tebalnya ngalahin kitab suci. Tebak apa? Jreng!! Dari sejuta halaman yang ditawarkan, untuk menjadikan saya seorang freelancer ternyata cuma diperlukan 100-200 halaman saja. Sisanya? Gak kepeke!! Ya masalah vektor lah, GUI lah, struktur data lah, socket lah, dll. Benar-benar gak kepeke! Rugi beli tuh buku.

Nah uniknya, dunia IT itu dinamis dan terus berkembang dengan cepat. Sempat tergoda untuk beli buku lagi, tapi ternyata buku-buku IT yang diterbitkan di toko buku itu selalu kalah cepat dengan current applied technology. Selalu ketinggalan. Soalnya, bagi seorang penulis buku, butuh 3-5 bulan untuk mendapat respon dari penerbit. Hasilnya begitu diterbitkan, ya udah basi. Kecuali kalau buku-buku IT nya membahas konsep khusus dan casenya unik.

Kembali ke topik. Lantaran menyadari gak semua topik programming bisa saya kuasai dan terpakai seluruhnya, maka saya ubah paradigma saya. Istilah kerennya, paradigma shifting. Bukan lagi harus mengetahui secara mendetail apa itu programming sebagai modal awal, melainkan harus berani menghadapi masalah. Nah, masalah demi masalah inilah yang akan membawa saya, termasuk Anda, untuk mengenal programming itu seperti apa. Bukan masalah sintaks, trust me,

Google dan Bing cukup powerful untuk masalah kamus sintaks. Tapi yang paling mahal adalah “Connecting the dots”, meminjam kata pengusaha ternama Richard Branson yang menaungi grup perusahaan Virgin Group. Ituloh, perusahaan yang membuat pesawat antariksa pertama untuk penerbangan sipil – Virgin Atlantic.

## **programmer itu sebenarnya hanya melihat masalah, menyusun opsi yang tersedia, dan menggabungkan solusi yang terbaik**

Dengan merangkai titik-titik permasalahan, saya jadi mengerti bagaimana menyambungkan programming database supaya jabat erat dengan programming UI (yang tentunya sangat berbeda jauh dengan apa yang diajarkan di buku). Jujur aja, saya gak tau konsep mendetail tentang UI. Tapi karena dihadapkan dengan masalah (requirement), mau gak mau ya harus pelajari juga toh meskipun gak 100% sampai detail banget. Intinya, cukuplah hadirkan sebuah solusi dari titik-titik permasalahan yang ada. Connecting the dots.

Jadi, tugas saya sebagai seorang programmer itu sebenarnya hanya melihat masalah, menyusun opsi yang tersedia, dan menggabungkan solusi yang terbaik. Nah, jika ini

dilakukan berulang-ulang, nanti skill nya akan terbentuk secara alami tanpa harus beli buku “Bagaimana Menjadi Programmer Hebat Sejagat”. Lebih jauh, kalau sense of problem solving nya itu udah terasah, maka Anda bisa maju ke hirarki berikutnya, yaitu developer. Developer adalah seorang programmer yang mengerti cara mengembangkan aplikasi yang dia atau orang lain buat. Sedangkan programmer hanyalah seorang problem solver yang terkadang cuma copas doank dari internet dengan sedikit tweak (seperti yang saya lakukan pada soal fungsi terbilang itu )

### **Acknowledge The Problem**

Kini Anda sudah tau kalau saya gak menguasai 100% tentang .NET. Damn, bahkan saya pun gak tau semua nama fungsi di dalam class SqlConnection. Tapi karena saya suka cari masalah, wkwkwk, perlahan-lahan saya kenal fungsi apa aja yang harus dipakai dari class SqlConnection. Paham ya cara paradigma shifting saya? Saya gak perlu harus tau fungsi SqlConnection. Function1() saat ini juga. Usia dan kesehatan saya jauh lebih berharga daripada melisting semua fungsi SqlConnection dan memahami tugas Function1() itu untuk apa. Alih-alih, saya cari masalah-masalah bisnis. Dan seperti tadi, Connecting the Dots, permasalahan-permasalahan tersebut akan menggiring saya secara otomatis ke Function1(). Istilahnya Up-Bottom, bukan Bottom Up.

Jadi, kenali dulu permasala-

hannya, baru bisa ketahuan library mana aja yang terkait (Connecting the dots). Jangan dibalik, saya sumpahin jadi programmer oon selamanya. Anda gak perlu tau semua fungsi overload dari MessageBox.Show yang ada 21 overloads!! Cukup kenali masalah yang ada, susun opsi yang tersedia, gabungkan solusi terbaik. Seumur hidup paling cuma pakai 2-3 overloads function dari MessageBox.Show. Gak perlu coba-coba fungsi yang tersisa. Sayang umur!! Masih ada gelar developer yang harus Anda raih!

Nah setelah paradigma shiftingnya terbentuk, tahap berikutnya adalah memahami permasalahan. Banyak programmer pemula yang kesulitan mendefinisikan mana yang jadi masalah. Mereka ibarat para elit politikus di Senayan sana yang tampak kesulitan mendefinisikan apa itu arti masalah dan solusi. Tenang, dulu saya begitu juga kok, hehe. Saya ambil contoh, kenapa terus menerus terjadi permasalahan ketidakserasian Daftar Pemilih Tetap (DPT) antara KPU dan Kemendagri? Dari dulu kasusnya itu mulu, bakalan bisa jadi masalah serius ketika pemilu. Lantas, ketika pemilu telah usai dan ditemukan ada kejanggalan, di manakah permasalahannya? Jika Anda ditugaskan untuk menghadirkan solusi IT bagi kasus tersebut, lantas, apa masalah utama yang harus Anda jawab? Apa memang karena ketimpangan data DPT antar dua lembaga? Apa karena mendadak ada orang mati sehingga DPT

nya jadi rancu? Apa karena ada yang berulang tahun yang ke 17 pada saat hari H-1 pemilu? Jika Anda diberi dana untuk membuat sebuah sistem IT KPU yang solid, titik permasalahan mana yang akan Anda masukkan ke dalam task-list? Ini bisa jadi kuis nih. Perlukah sistem IT KPU yang Anda develop harus mampu mengcover masalah-masalah orang mati? Apa benar orang mati bisa mengakibatkan data DPT bermasalah? Perlukah dilakukan validasi sampai ke tingkat RT? Jangan-jangan perlu validasi juga ke kuburan?? Hayooo, jawab! Hahahaha...

Apa pentingnya memahami mana masalah yang real (sebab) dan masalah yang fake (akibat)? Tentu saja untuk mendapatkan mana aja library .NET yang bisa dipakai. Ujung-ujungnya akan menghasilkan solusi yang tepat guna.

Eh penulis dodol! Katanya Up-Bottom. Kalau gak kenal dengan library .NET nya masing-masing,

gimana bisa tau yang cocok yang mana aja? Berarti harus tetap Bottom-Up donk!

Hehehe, kan ada internet! Internet udah menyediakan begitu banyak resource yang dibutuhkan tanpa harus memahami satu per satu semua fungsi dalam sebuah class. Ibaratnya begini, Up-Bottom itu menggunakan helikopter untuk menuju ke suatu tempat, sedangkan Bottom-Up itu menggunakan navigasi darat, termasuk kompas! Dengan menggunakan Up-Bottom, Anda bisa melihat lebih banyak jalan menuju Roma. Anda bisa melihat mana aja jalan yang buntu dari jauh. Berbanding terbalik bila Bottom-Up, harus menguasai semua event dan fungsi sebuah class, dan alasan-alasan teknis lainnya. Dan ada satu hal lagi kenapa harus Up-Bottom, yaitu untuk mempercepat akselerasi. Kalau berani menghadapi masalah, itu bisa membawa skill Anda berkenalan dengan ranah



programming yang berbeda lebih cepat daripada cara konvensional.

**BOTTOM-UP:** misalnya mahir di fungsi2 matematika dan hapal semua class-classnya. Tapi butuh waktu lebih lama bila diminta untuk membuat aplikasi arsitektur.

**UP-BOTTOM :** gak begitu menguasai semua class matematika tapi bisa membuat aplikasi arsitektur dengan menerapkan solusi matematis. Bahkan dia bisa membuat aplikasi non-matematik dengan pendekatan matematis. Misalnya memanfaatkan fungsi-fungsi linear dan eksponensial untuk sistem ketahanan pangan.

Tapi memang, tipe programmer itu berbeda-beda. Ada yang tugasnya bikin compiler, ada yang dekat dengan hardware, dll. Ada yang harus Bottom-Up, gak bisa diganggu gugat. Tapi yang kita bicarakan di sini adalah programmer yang akan melesat jauh dan itu adalah pilihan.

## Expand Your Read

Jangan malas baca dan bacanya juga jangan itu-itu aja. Jangan tentang hal-hal teknis melulu. Sekali-kali baca artikel tentang visi dan misi dari sebuah vendor, bukan produknya. Ini ibaratnya menaikkan helikopter yang Anda kendarai. Nanti dari visi dan misi tersebut, Anda bisa mengendus jenis permasalahan bisnis dan IT di masa mendatang dan Anda sudah lebih siap untuk hal itu. Saya pribadi, kalau sudah

dapat proyek database yang notabene hanya CRUD aja sebanyak 3-4 kali, biasanya saya udah muak dan menolak tipe proyek yang sama untuk saya kerjakan dengan tangan saya langsung. INSERT INTO, UPDATE SET, DELETE FROM, dari dulu dan sampai kiamat untuk RDBMS akan tetap seperti itu. Ngapain lama-lama nulis sintaks begituan? Selagi masih muda dan belum ada tunggangan hidup, jangan kelamaan nulis sintaks begituan. Diledekkan sama malaikat . Kecuali..... Projeknya unik dan ada nilai tambahnya.

## Conclusion

Sebagai kesimpulan, jangan malas untuk mencari solusi sebuah permasalahan. Cari aja masalah, apa kek, banyak! Iseng-iseng bikin aplikasi sendiri untuk KPU, Kemendagri, Dirlantas, KPK, BKKBN, dan lainnya. Kalau ada lomba INAICTA, ikut aja. Dulu saya ikut, bikin aplikasi Smart Hospital. Meskipun gak lolos lantaran saya udah gak kuliah lagi (sulit bagi waktu dengan yang masih kuliah waktu itu), tapi ada pengalamannya. Bahkan saya kaget dengan konsep yang mereka tawarkan. Yaitu si pasien rawat inap akan dipakaikan gelang khusus yang ada chipnya. Melalui gelang ini si dokter bisa melihat rekam medis si pasien secara digital. Dan berbagai macam kemudahan lainnya. Mau gak mau saya berkenalan dengan library-library yang sekiranya dibutuhkan – Connecting the dots.

Akhir kata, selamat belajar dan asah terus skill programming Anda. Programmer haruslah menjadi developer. Dan developer yang hebat harus bisa mengganti toolboxnya dari IDE Visual Studio menjadi Excel dan Power Point suatu saat nanti. Ya presentasi-presentasi ajah . Good luck!



**Endar Zulfahrain**

I'm a scientist who addicted with this universe that hides a love story behind it ...

"It's not about how many language of syntax you have mastered, but how many problems you have fixed" - Virgin\Code++

"it's so hard to understand that you can't make something different but you always demand something new" - zulfahrain



# Dasar Pemrograman Berorientasi Obyek

Sebuah paradigma yang sering dibicarakan namun kadang sulit dipahami oleh para programmer pemula



**S**ebelum lebih jauh membahas tentang OOP ada baiknya Anda mengerti dulu apa yang dimaksud dengan paradigma pemrograman. Paradigma pemrograman adalah **sudut pandang** dalam menyelesaikan suatu persoalan.

Dari cara pandang terhadap persoalan yang akan dipecahkan masalahnya dengan pemrograman ini dalam perkembangannya muncullah beberapa paradigma pemrograman, yaitu Procedural atau Imperative, Declarative, Functional, dan Object Oriented.

## Prosedural atau Imperatif

Paradigma ini didasari oleh konsep mesin Von Newman (stored program concept) sekelompok tempat penyimpanan (memori), yang dibedakan menjadi memori instruksi dan memori data, masing-masing memori tersebut dapat diberi nama dan nilai, selanjutnya instruksi akan dieksekusi satu persatu secara sekuensial oleh sebuah proses tunggal.

Program dalam paradigma ini berdasarkan pada struktur informasi di dalam memori dan manipulasi dari informasi yang disimpan tersebut. Kata kunci yang sering digunakan dalam paradigma ini adalah:

## Algoritma + Struktur Data = Program

Kelebihan dari paradigma ini adalah efisiensi eksekusi karena lebih dekat dengan konsep mesin, kekurangannya adalah batasan yang sangat mengikat sehingga terkadang menyulitkan programmer yang tidak terbiasa.

Contoh bahasa pemrograman yang menggunakan paradigma prosedural atau imperatif adalah: Algol, Pascal, Fortran, Basic, Cobol, C, dsb...

## Declarative, Prediktive atau Logic

Paradigma ini di dasari oleh pendefinisian relasi antar individu yang dinyatakan sebagai predikat.

Sebuah program logik adalah kumpulan aksioma (fakta dan aturan deduksi). Program dieksekusi, pemakai mengajukan pertanyaan (query), dan program akan menjawab apakah pernyataan itu dapat dideduksi dari aturan dan fakta yang ada. Program akan memakai aturan deduksi dan mencocokkan pertanyaan dengan fakta-fakta yang ada untuk menjawab pertanyaan.

## Functional

Paradigma ini di dasari oleh kon-

sep pemetaan dan fungsi pada matematika. Fungsi dapat berbentuk sebagai fungsi "primitif", atau komposisi dari fungsi-fungsi lain yang telah terdefinisi.

Pemrogram mengasumsikan bahwa ada fungsi-fungsi dasar yang dapat dilakukan. Penyelesaian masalah didasari atas aplikasi dari fungsi-fungsi tersebut.

Jadi dasar pemecahan persoalan adalah transformasional. Semua kelakuan program adalah suatu rantai transformasi dari sebuah keadaan awal menuju kesuatu rantai akhir, yang mungkin melalui keadaan antara melalui aplikasi fungsi.

Paradigma fungsional tidak lagi memperlakukan memori-sasi dan struktur data, tidak ada pemilihan antara data dan pemrograman, tidak ada lagi pengertian tentang "variable".

Pemrograman tidak perlu lagi mengetahui bagaimana mesin mengeksekusi atau bagaimana informasi disimpan dalam memori, setiap fungsi adalah kotak hitam, yang menjadi perhatiannya adalah awal dan akhir. Dengan merakit kotak hitam ini maka akan menghasilkan program yang besar.

## Object Oriented



Pemrograman berorientasi objek (object-oriented programming disingkat OOP) merupakan paradigm pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Bandingkan dengan logika pemrograman terstruktur.

Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya. Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam tehnik pengembangan perangkat lunak skala besar.

Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya. Beberapa keunggulan dari OOP ini antara lain:

## Maintainability

Maintainability maksudnya aplikasi yang dibuat dengan model OOP lebih mudah dipelihara dan dikelola.

Dengan semakin besarnya aplikasi yang Anda buat akan semakin sulit menangani error diakibatkan oleh ketergantungan antara kode yang satu dengan kode lainnya.

Untuk mengatasi hal ini Anda dapat mempersempit atau membatasi masalah dengan membuat modul-modul kecil yang dapat memecahkan masalah besar menjadi bagian yang kecil-kecil sehingga memudahkan dalam

pemecahan masalah.

## Extensibility

Extensibility maksudnya aplikasi yang dibuat dengan model OOP lebih mudah diperluas. Anda dapat membuat objek dari kelas yang dapat diperluas dengan cara menambahkan property, method.

Anda dapat membuat kelas turunan yang diperluas dari kelas induk sesuai dengan keinginan Anda. Ini akan menghemat waktu Anda karena anda tidak perlu membuat semua kode dari awal.

## Reusability

Reusability maksudnya Anda atau team programmer lain dapat menggunakan kembali

kode yang sudah pernah anda buat sebelumnya

Sebuah program bisa dikategorikan berorientasi obyek jika memenuhi empat unsur dasar berikut ini, yaitu Abstraksi (*abstraction*), Enkapsulasi (*encapsulation*), Inheritansi (*inheritance*), Polimorfisme (*polymorphism*)

## Abstraksi (Cara Pandang)

Abstraksi adalah cara pandang seseorang terhadap sesuatu. Cara Anda memandang sesuatu bisa berbeda dengan cara orang lain memandang benda yang sama.

Mungkin Anda memandang sebatang pohon adalah tempat orang bernaung. Namun bisa saja anak anda yang masih kecil

## CODE #1: Enkapsulasi

```
Public Class CMarketing

    Private Sub HubungiProspek()
    End Sub

    Private Sub KirimMateriPenjualan()
    End Sub

    Private Sub FollowUpPenawaran()
    End Sub

    Public Function getCustomer(ByVal custId As String) As Array
    End Function

End Class

Public Class CPengiriman

    Private Sub TerimaPesanan()
    End Sub

    Private Sub PackBarang()
    End Sub

    Private Sub KirimBarangKeCustomer(ByVal custId As String)
        Dim oMarketing As CMarketing
        oMarketing = New CMarketing
        Dim cust As Array
        cust = oMarketing.getCustomer(custId)
    End Sub

End Class
```

memandang pohon sebagai tempat untuk memanjat.

## Enkapsulasi (Pembungkusan)

Dalam berbagai jenis perusahaan bisanya ditemukan beberapa departemen-departemen. Katakanlah departemen penjualan dan departemen pengiriman barang.

Masing-masing departemen jelas bertanggungjawab terhadap prosedur pekerjaan internalnya masing-masing.

Katakanlah departemen Marketing mempunyai prosedur pekerjaan seperti menghubungi prospek, mengirim materi penjualan dan memfollow up penawaran. Dan bagian pengiriman barang misalnya mempunyai prosedur kerja menerima pesanan barang, mengepak barang dan mengirimkannya ke customer.

Jika kita modelkan dalam OOP akan seperti terlihat pada Source Code #1.

## Inheritansi (Penurunan Sifat)

Berbicara mengenai Inheritance berarti berbicara bagaimana satu kelas dapat diturunkan dari kelas lainnya.



Misalkan saja Karyawan pada perusahaan bisa banyak. Ada yang bekerja sebagai marketing namanya Wati dan ada pula yang bekerja sebagai programmer namanya Arman. Wati dan Arman yang sama-sama karyawan tentunya punya property dasar yang diwarisi dari property kelas Karyawan.

Properti-property standar karyawan bisa saja terdiri dari nama, alamat, dan gaji pokok. Properti standar karyawan ini bisa diturunkan langsung ke kelas marketing dan kelas programmer.

Kelas Marketing bisa mempunyai property tambahan misalnya kemampuan memasarkan. Sedangkan kelas programmer bisa mempunyai property khusus kemampuan buat kode. Seperti terlihat di diagram.

Anda lihat kelas Marketing dan Programmer diturunkan (Inherit) dari kelas Karyawan. Kedua kelas yang diturunkan ini kemudian disebut dengan sub kelas. Seperti terlihat pada Source Code #2

## Polymorphism (Berubah Bentuk)

Polimorfisme sesuai dengan namanya adalah dapat berubah bentuk dari bentuk dasarnya. Ada dua bentuk perubahan yang dapat Anda lakukan, yaitu **Overloading** atau **Overriding**.

### Overloading

Berikut aturan pendeklarasian method

overloading :

- Nama method harus sama
- Daftar parameter harus berbeda
- Return type boleh sama, juga boleh berbeda

### Overriding

Jika dalam suatu subclass kita mendefinisikan sebuah method yang sama dengan yang dimiliki superclass, maka method yang kita buat dalam subclass dikatakan meng-override superclass-nya.

Sehingga jika kita mencoba memanggil method tersebut dari instance subclass yang kita buat, maka method milik subclasslah yang akan dipanggil, bukan lagi method milik superclass.

Overriding mempunyai ciri-ciri sebagai berikut :

1. Nama method harus sama
2. Daftar parameter harus sama
3. Return type harus sama

Beberapa aturan yang perlu diperhatikan :

- Mode akses overriding method harus sama atau lebih luas daripada overridden method.
- Subclass hanya boleh meng-override method super class satu kali saja, tidak boleh ada lebih dari satu method pada kelas yang sama yang sama persis.
- Overriding method tidak boleh throw checked exceptions yang tidak dideklarasikan oleh overridden method.

Pada contoh kasus Anda dapat

mengimplementasikannya seperti ini. Marketing dan Programmer adalah sama-sama karyawan yang mungkin sama-sama mempunyai perhitungan gaji. Tetapi mungkin parameter ataupun factor-faktor yang mempengaruhi gaji mereka bisa berbeda.

Perubahan bentuk dengan perbedaan jumlah parameter pada kelas Anak/Subclass/derived class ini selanjutnya kita sebut sebagai Overloading.

### Abstract Class (Kelas Abstrak)

Yang di maksud dengan Abstract Class adalah Class Abstrak yang khusus dibuat untuk keperluan inheritance (penurunan sifat). Tujuan dari pembuatan abstract class ini adalah untuk membuat definisi umum bagi class-class yang akan menjadi Class turunan (subclass/derived) nya.

Abstract Class tidak bisa diinstantiasi. Abstract method tidak mempunyai implementasi. Abstract method ini bisa digunakan oleh Class turunannya dengan melakukan Override.

Abstract class hanya bisa digunakan sebagai super class/kelas induk (yang menurunkan sifat), tapi juga bisa diturunkan dari class abstract lainnya. Untuk mendeklarasikan sebuah abstract class.

Sebuah abstract class pada dasarnya tidak jauh beda dengan class lainnya, yakni juga berisi method yang menggambarkan karakteristik dari kelas abstract tersebut, bedanya yakni sebuah abstract class bisa berisi method tanpa diimplementasikan arti-

### CODE #2: Inheritance dan Polymorphism

```
Public Class CKaryawan
    Private _Nama As String
    Private _Alamat As String
    Private _GajiPokok As Double

    Public Property Nama
        Get
            Nama = _Nama
        End Get
        Set(ByVal value)
            _Nama = value
        End Set
    End Property

    Public Property Alamat
        Get
            Alamat = _alamat
        End Get
        Set(ByVal value)
            value = _alamat
        End Set
    End Property

    Public Property GajiPokok
        Get
            GajiPokok = _GajiPokok
        End Get
        Set(ByVal value)
            _GajiPokok = value
        End Set
    End Property

    Public Function getGaji(ByVal iAbsen As Integer, _
        ByVal dGajiPokok As Double) As Double
        Return iAbsen * dGajiPokok
    End Function
End Class

Public Class CProgrammer
    Inherits CKaryawan

    Private Overloads Function getGaji(ByVal iAbsen As Integer, _
        ByVal dGajiPokok As Double, _
        ByVal bonusSkill As Double) As Double
        Return iAbsen * dGajiPokok * bonusSkill
    End Function
End Class

Public Class CMarketing
    Inherits CKaryawan

    Private Overloads Function getGaji(ByVal iAbsen As Integer, _
        ByVal gajiPokok As Double, _
        ByVal bonusPemasaran As Double, _
        ByVal uangMinyak As Double) As Double
        Return (iAbsen * GajiPokok) + bonusPemasaran * uangMinyak
    End Function
End Class
```



nya sebuah method tanpa body, method seperti ini disebut method abstract.

### Kegunaan Class Abstract

Class Abstract berisi beberapa method dan beberapa method abstract. Class Abstract berisi sebagian implementasi, dan subclass yang melengkapi implementasinya.

Dengan kata lain Class Abstract memiliki beberapa kesamaan (Bagian yang diimplementasikan oleh subclass) dan memiliki perbedaan (method yang dimiliki sendiri oleh class abstract)

Deklarasikan method abstract, jika ada satu atau lebih subclass yang diharapkan mempunyai fungsionalitas yang sama tapi implementasi berbeda.

Gunakan class abstract untuk mendefinisikan behavior secara umum sebagai superclass, sedangkan subclass menyediakan implementasi detail.

Jika class abstract semua method merupakan method abstract, sebaiknya class abstract tersebut diganti menjadi Interface

Anda dapat membayangkan Kelas Karyawan sebagai kelas abstract. Lihat struktur diagram diatas. Karyawan bisa terdiri dari programmer dan marketing. Programmer dan Marketing ini selanjutnya kita sebut sebagai class turunan (sub class) sedangkan Karyawan kita sebut sebagai kelas yang menurunkan sifat (super class)

### Interface

Interface mirip dengan class memiliki property, method dan event. Bedanya dengan class biasa interface ini tidak memiliki implementasi.

Interface berfungsi sebagai antarmuka yang membentuk komunikasi dengan code lain. Misalnya membentuk hubungan antara sebuah object dengan object yang lain atau hubungan antara object sebagai penyedia dengan code pengguna.

Interface adalah jenis khusus dari blok yang hanya berisi method signature (atau constant). Interface mendefinisikan sebuah (signature) dari sebuah kumpulan method tanpa tubuh.

Interface mendefinisikan sebuah cara standar dan umum dalam menetapkan sifat-sifat dari class-class. Mereka menyediakan class-class, tanpa memperhatikan lokasinya dalam hirarki class, untuk mengimplementasikan sifat-sifat yang umum.

Dengan catatan bahwa interface-interface juga menunjukkan polimorfisme, dikarenakan program dapat memanggil method interface dan versi yang tepat dari method yang akan dieksekusi tergantung dari tipe object yang melewati pemanggil method interface

### Kegunaan Interface

Keperluan interface pada penerapannya biasanya adalah untuk "keperluan pengembangan" Class Anda di masa depan. Jika kode yang Anda buat semuanya berdasarkan class lalu terjadi perubahan pendefinisian Class induk seperti parameter atau lainnya maka sub class (kode yang menggunakan super class / class induk) akan bermasalah. Dengan menggunakan Interface Anda dapat merubah implementasi.

Untuk menggunakan Interface Anda dapat menggunakan keyword *Implements*, maka IDE Visual Basic.net otomatis menu-

#### CODE #3: Interface

```
Public Interface ITunjangan
    Function tJabatan() As Boolean
    Function tIstri() As Boolean
    Function tAnak() As Boolean
End Interface

Public Class CKaryawan2
    Implements ITunjangan

    Public Function tAnak() As Boolean Implements ITunjangan.tAnak
        Return True
    End Function

    Public Function tIstri() As Boolean Implements ITunjangan.tIstri
        Return True
    End Function

    Public Function tJabatan() As Boolean Implements ITunjangan.tJabatan
        Return True
    End Function
End Class
```

liskan function yang di-implement di kelas tempat Anda membuat tulisan implements.

Visual Basic 6.0 disebut semi OOP atau OOP banci karena OOP di vb6 hanya bisa melakukan implemets dan tidak bisa melakukan inherits.

So dah jelas ya bedanya kalau Inherits bisa diperoleh dari kelas abstract sebagian dari method atau parameternya. Sedangkan Implements mengambil semua interface dari kelas induk mentah-mentah.

## Hello World Rasa OOP

OK baiklah pada ahir tulisan ini saya berikan contoh sebuah aplikasi Hello World dan menerapkan semua konsep OOP yang sudah kita bahas dalam artikel ini. Silakan pelajari source code #4 di samping ini.

## Tentang Penulis



**Ir. Hartoto Dinata** adalah CEO & Founder at XBasicPro (<http://xbasicpro.com>), Internet Marketing Manager at PT. Ody Lestari Advertising Company and guru kursus online at Kiat Bisnis Online.

## CODE #4: Hello World Rasa OOP

```
''' <summary>
''' Interface Printer
''' </summary>
Public Interface Printer
    Sub PrintOut(ByVal message As String)
End Interface

''' <summary>
''' implementasi interface Printer
''' </summary>
Public Class SystemOutPrinter
    Implements Printer
    Public Sub PrintOut(message As String) Implements Printer.PrintOut
        Console.WriteLine(message)
    End Sub
End Class

''' <summary>
''' Class berisi pesan yang ingin ditampilkan
''' </summary>
Public Class Message
    Private message As String
    ' constructor
    Public Sub New(ByVal message As String)
        Me.message = message
    End Sub
    ' fungsi untuk menampilkan pesan
    Public Sub PrintOut(printer As Printer)
        printer.PrintOut(Me.message)
    End Sub
    ' fungsi untuk mengubah pesan menjadi string
    Public Overrides Function ToString() As String
        Return Me.message
    End Function
End Class

''' <summary>
''' contoh class abstract
''' harus dibuat object inheritance kalau mau pake
''' </summary>
Public MustInherit Class AbstractPrinterFactory
    Public Shared Function GetFactory() As AbstractPrinterFactory
        Return New SystemOutPrinterFactory()
    End Function
    Public MustOverride Function GetPrinter() As Printer
End Class

''' <summary>
''' hasil turunan abstract class
''' </summary>
Public Class SystemOutPrinterFactory
    Inherits AbstractPrinterFactory
    Public Overrides Function GetPrinter() As Printer
        Return New SystemOutPrinter()
    End Function
End Class

''' <summary>
''' module utama
''' </summary>
Module HelloWorld
    Public Sub Main()
        Dim message As New Message("Hello world!")
        Dim factory As AbstractPrinterFactory
        Dim printer As Printer
        factory = SystemOutPrinterFactory.GetFactory()
        printer = factory.GetPrinter()
        message.PrintOut(printer)
    End Sub
End Module
```

# Alternatif Reporting Di Lingkungan .Net Framework

Crystal Report rupanya hanya satu dari banyak alternatif pembuatan report di lingkungan .Net Framework

**D**alam pengolahan data dan informasi, setidaknya ada 3 tahapan yang penting yang dapat digambarkan dalam diagram berikut:

- Input biasanya berupa data yang diinput oleh user ke dalam sistem dan disimpan dalam media tertentu misalnya database.
- Process merupakan pengolahan data sedemikian rupa dengan proses bisnis tertentu sehingga menjadi informasi yang berguna.
- Output merupakan hasil pengolahan data, yaitu informasi yang berguna bagi user untuk menjalankan proses bisnis. Contoh output yang lazim dijumpai adalah report.

Dengan adanya report di dalam sebuah sistem, user mendapatkan informasi keadaan bisnisnya saat ini, dan dari informasi tersebut user dapat mengambil langkah yang tepat untuk perkembangan bisnisnya

di masa depan.

Software untuk membuat report disebut Reporting Tool. Software ini digunakan oleh developer untuk membuat laporan (report).

## Pilih Reporting Tool Favoritmu

Beberapa reporting tool yang pernah digunakan oleh penulis diantaranya adalah Crystal Report, SQL Server Reporting Services, Client Processing Report (RDL), dan Telerik Report.

## Crystal Report

Software Crystal Report tidak asing lagi di telinga developer, baik developer pemula ataupun yang sudah veteran, terutama developer yang dulu pernah menggunakan Visual Basic 6. Di platform .NET, Crystal Report dibundling bersama dengan Visual Studio sejak Visual Studio .NET 2002 sampai Visual Studio 2008 yang dinamai Crystal Report Basic. Namun, sejak Visual Studio 2010 sampai versi terkini, Visual Studio 2012, Crystal Report tidak lagi dibundling di dalam Visual Studio, melainkan harus diunduh

terpisah dari website SAP, dan namanya diganti menjadi Crystal Report Developer Version for Microsoft Visual Studio yang dapat digunakan secara gratis namun tentunya dengan beberapa keterbatasan. Untuk versi lengkapnya, kita harus membeli software SAP Crystal Report. Berikut beberapa pro-cons penggunaan Crystal Report menurut penulis:

Pro:

- Proses instalasi dan penggunaan designer relatif mudah dipelajari, developer mestinya melakukan instalasi dengan mudah.
- Report designer sangat fleksibel, developer dapat dengan mudah membuat page header, report header, group header, dan sebagainya.
- Report dapat di-embed di aplikasi dan akan dibuild bersama dengan aplikasi atau terpisah. Jika report terpisah dengan aplikasi,



report dapat dimodifikasi tanpa melakukan build ulang aplikasi.

- Dapat menggunakan Strongly Typed DataSet sebagai data source.

Cons:

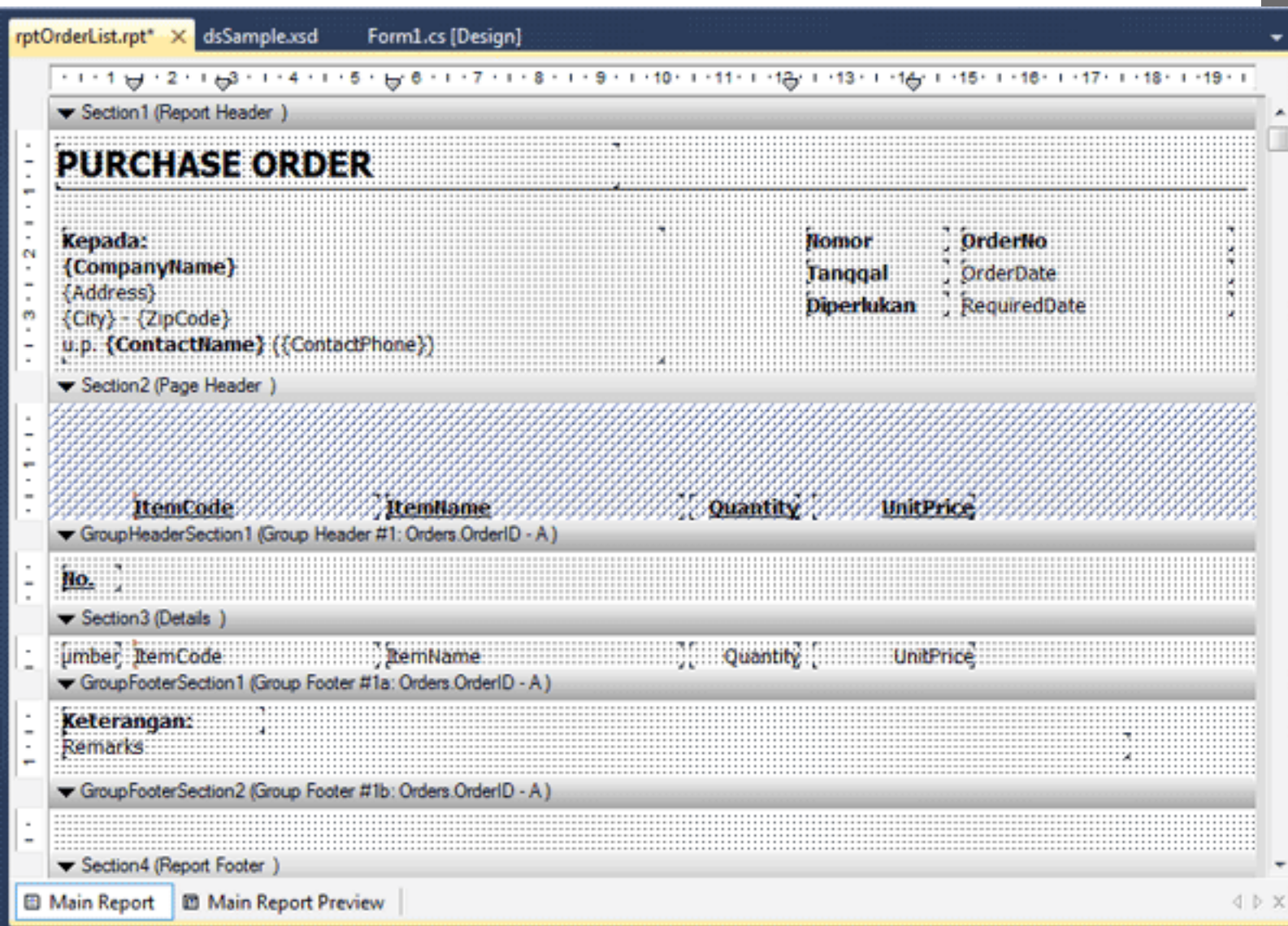
- Fitur yang terbatas untuk versi Crystal Report Developer Version. Penulis pernah mengalami kesulitan saat membuat report crosstab (seperti pivot table di Excel), yang di dalamnya

ada grouping dan summary field dengan formula yang cukup kompleks.

- Adakalanya Crystal Report tidak ada melakukan refresh nama field. Penulis pernah menggunakan stored procedure Oracle sebagai data source, dan saat stored procedure dimodifikasi dengan menambahkan field, kendalanya Crystal Report tidak bisa menambahkan field baru tersebut sehingga penulis harus menambahkan

ulang (dihapus lalu create lagi) stored procedure sebagai data source.

- Untuk Crystal Report Developer Version, mulai Visual Studio 2010, installernya harus didownload terpisah di website SAP. Sebelum Visual Studio 2010, installer Crystal Report dibundle bersama installer Visual Studio. Installer yang harus didownload ukurannya juga cukup besar.



## SQL Server Reporting Services (SSRS)

Sejak kemunculan pertamanya di SQL Server 2000, SQL Server Reporting Services atau yang lebih singkat disebut SSRS, semakin banyak digunakan dalam aplikasi yang dibangun dengan .NET. Installer SSRS hadir bersama dengan installer SQL Server, artinya ketika kita menginstall SQL Server, akan ada pilihan apakah juga akan menginstall Reporting Services. Untuk membuat report dengan SSRS, developer membutuhkan Business Intelligence Development Studio

(di SQL Server 2005 sampai SQL Server 2008 R2) atau SQL Server Data Tools (di SQL Server 2012), keduanya merupakan tools yang dibangun di atas Visual Studio Shell. Developer tentunya akan merasa sangat familiar karena tool yang digunakan untuk coding sama persis dengan tool yang digunakan untuk membuat report. Dengan SSRS, developer dapat membangun Report Server yang bertugas khusus untuk mengenerate report, sehingga beban Application Server dapat dimaksimalkan khusus hanya untuk aplikasi. SSRS juga menawarkan fitur

Subscribe Report, yaitu report dapat digenerate secara otomatis di waktu tertentu yang sudah diset oleh developer, dan report tersebut langsung dikirimkan ke user, misalnya melalui email atau sharing folder. SSRS juga dapat digunakan di versi gratis dari SQL Server, yaitu SQL Server Express with Advanced Services. Tentunya, ada keterbatasan fitur SSRS di SQL Server Express ini. Berikut beberapa pro-cons penggunaan Crystal Report menurut penulis:

Pro:

- Terintegrasi dengan SQL

The screenshot displays the SSRS design view for a report titled 'Sales Order Details.trdx [Design]'. The report layout includes a header section 'Order Details' and a table with five columns: Product No., Name, Quantity, Unit Price, and Line Total. The table rows are populated with data from the 'Fields' collection, such as '[=Fields.ProductNumber]' and '[=Fields.Name]'. The design view also shows a 'Group Explorer' pane at the bottom, which lists the report groups and their associated header, footer, and detail sections. The 'LabelsGroup' is expanded, showing the 'Detail' section with the 'reportHeaderSection1' header.

Product No.	Name	Quantity	Unit Price	Line Total
[=Fields.ProductNumber]	[=Fields.Name]	[=Fields.OrderQty]	[=Fields.UnitPrice]	[=Fields.LineTotal]

Group Explorer

Name	Header	Footer	Grouping	Filter	Sorting
LabelsGroup	LabelsGroupHeader	LabelsGroupFooter			
Detail	reportHeaderSection1			= Fields.Sal...	

Server dan jika diinstall di server yang sama dengan SQL Server, tidak perlu membeli lisensi yang terpisah dengan SQL Server.

- Karena arsitekturnya yang merupakan server-based reporting, kita dapat membangun server yang dedicated untuk reporting, server reporting ini yang akan bekerja khusus untuk mengenerate report dan terpisah dari aplikasi atau application server. Jika kita ingin membangun sebuah enterprise system yang membutuhkan infrastruktur server reporting yang khusus (dedicated), SSRS merupakan pilihan yang sangat tepat.
- Designer menggunakan Visual Studio, sehingga developer mestinya familiar dengan designer reportnya. Namun, jika versi Visual Studio yang digunakan untuk membuat aplikasi dengan versi SSRS berbeda, developer harus menjalankan dua versi Visual Studio bersamaan, misalnya SSRS 2008R2 menggunakan Visual Studio 2008, sedangkan untuk membuat aplikasi menggunakan Visual Studio 2012 karena developer akan menggunakan .NET Framework 4.0.
- Terintegrasi penuh dengan platform Sharepoint dan SQL Server Analysis Service (SSAS) sebagai platform Business Intelligence.

Cons:

- Proses instalasinya perlu dipelajari terlebih dahulu dan tidak semudah melakukan instalasi reporting tool lain, misalnya instalasi Crystal Report. Namun menurut penulis, proses instalasi dapat dipelajari dengan relatif mudah.
- Report perlu dideploy dulu ke report server sebelum dapat digunakan oleh aplikasi. Proses deployment sebenarnya sangat mudah, dapat dilakukan langsung dari Visual Studio.
- Tidak dapat menggunakan Strongly Typed DataSet sebagai data source. Data source hanya dapat menggunakan command SQL, table, view, atau stored procedure.

## Client Report Definition (RDLC)

Yaitu report yang dapat dibuat bersama dengan code aplikasi, dan mempunyai ekstensi .RDLC. Sebenarnya RDLC dengan SSRS ada kemiripan, report yang dibuat dengan SSRS mempunyai ekstensi .RDL, dan mempunyai XML schema yang sama dengan file .RDLC, sehingga file .RDLC sebenarnya dapat dikonversi menjadi report SSRS (RDL) dengan kondisi tertentu. Report RDLC ini akan dibuild bersama dengan code aplikasi atau dapat pula dibuild menjadi file DLL, sehingga apabila kita ingin melakukan modifikasi report, kita harus melakukan build/compile ulang aplikasi. Baik, RDLC

maupun RDL, membutuhkan komponen Report Viewer yang dapat dijumpai di toolbox Visual Studio untuk menampilkan report di dalam aplikasi .NET. Berikut beberapa pro-cons penggunaan Crystal Report menurut penulis:

Pro:

- Designer terintegrasi dengan Visual Studio, sehingga saat membuat report tidak perlu pindah aplikasi.
- RDLC secara default langsung terinstall bersama dengan Visual Studio.
- Kemudahan migrasi ke SQL Server Reporting Service (RDL).
- Dapat menggunakan Strongly Typed DataSet sebagai data source.

Cons:

- Report harus dibuild bersama dengan aplikasi atau assembly lain (DLL), sehingga ketika kita melakukan modifikasi report, aplikasi atau assembly tersebut harus dibuild ulang.

## Telerik Report

Untuk menggunakan reporting tool ini, kita harus merogoh kocek sebanyak \$599 per developer, namun sebenarnya kita dapat mendownload trial version untuk mencobanya terlebih dahulu. Menurut penulis, Telerik Report cukup mudah digunakan, Telerik Report Designer nya menyerupai



Reporting Service (SSRS) dan Microsoft Office, yaitu menu-menunya diletakkan di dalam ribbon. Integrasi Telerik Report dengan aplikasi berplatform .NET juga sangat mudah. Berikut beberapa pro-cons penggunaan Crystal Report menurut penulis:

Pro:

- Menurut penulis, report designer sangat intuitif, report designer dari Telerik Report menggabungkan kemudahan designer Crystal Report dan kelebihan designer SSRS.

- Terintegrasi dengan tool Telerik lain, yaitu OpenAccess.
- Seperti halnya Crystal Report, report dapat dibuild bersama dengan aplikasi atau terpisah.
- Designer ada dua, yaitu designer yang terintegrasi dengan Visual Studio (developer dapat membuat report langsung di Visual Studio) dan aplikasi report designer khusus Telerik Report.

Cons:

- Developer harus membeli lisensi Telerik Report


## Server Reporting vs Client Reporting

Yang dimaksud penulis dengan Server Reporting adalah, report dapat dieksekusi di server dedicated, terpisah dengan aplikasi. Dengan keterpisahan dari aplikasi ini, tentunya ada proses generate report tidak membebani aplikasi. Berikut ini contoh arsitektur sistem yang di dalamnya terdapat report server:

Dari reporting tool yang disebutkan di atas, hanya SSRS yang mempunyai arsitektur server reporting ini. Untuk Crystal Report hanya versi SAP

ClubReport.rdl [Design] ✕

Design Preview



**CLUB REPORT**

[@BranchID.Label]

«Expr»

«Expr»

«Expr»

	Daily	MTD	Target	Achievement %	Month To Go	Projection	Projection %	YTD
Fresh Member Unit	[FreshMemberUnit_	[FreshMemberUnit_	[FreshMemberUnit_	«Expr»	«Expr»	«Expr»	«Expr»	[FreshMemberL
Renewal Unit	[RenewalUnit_daily	[RenewalUnit_mtd]	[RenewalUnit_targ	«Expr»	«Expr»	«Expr»	«Expr»	[RenewalUnit_y
Upgrade Unit	[UpgradeUnit_daily]	[UpgradeUnit_mtd]	[UpgradeUnit_targ]	«Expr»	«Expr»	«Expr»	«Expr»	[UpgradeUnit_y
Total Unit Sold	«Expr»	«Expr»	«Expr»	«Expr»	«Expr»	«Expr»	«Expr»	«Expr»
Fresh Member Revenue	[FreshMemberReve	[FreshMemberRev]	[FreshMemberRev]	«Expr»	«Expr»	«Expr»	«Expr»	[FreshMemberR
Renewal Revenue	[RenewalRevenue_	[RenewalRevenue	[RenewalRevenue	«Expr»	«Expr»	«Expr»	«Expr»	[RenewalRever
Upgrade Revenue	[UpgradeRevenue_	[UpgradeRevenue_	[UpgradeRevenue_	«Expr»	«Expr»	«Expr»	«Expr»	[UpgradeReven
Tota Member Revenue	«Expr»	«Expr»	«Expr»	«Expr»	«Expr»	«Expr»	«Expr»	«Expr»

Row Groups

Column Groups

«(Details)»

Crystal Report (bukan Crystal Report Developer Version) yang mempunyai arsitektur server reporting.

Sedangkan Client Reporting adalah, report dieksekusi di dalam process yang sama dengan aplikasi. Di satu sisi hal ini bisa memberatkan aplikasi yang sedang berjalan, tetapi di sisi deployment aplikasi, tentunya ini lebih sederhana dibanding Server Reporting.

### **Bagaimana dengan Microsoft Excel?**

Mungkin ada pembaca yang bertanya, apa kita bisa membuat report dengan Microsoft Excel? Untuk pertanyaan ini, penulis harus memberi jawaban ganda, ya dan tidak. Jika report yang

akan dibuat sangat sederhana, hanya menampilkan hasil query dari database dan tidak ada pengolahan data, Excel bisa menjadi pilihan sebagai reporting tool, developer dapat dengan mudah membuat file Excel menggunakan Excel Interop atau dengan library lain misalnya NPOI, EPPlus, atau yang lainnya. Namun, apabila report yang akan dibuat cukup kompleks, apalagi ada perhitungan yang cukup banyak, atau mungkin sampai perlu membuat grafik, saran dari penulis, lebih baik menggunakan reporting tool misalnya salah satu dari software reporting tool yang telah disebutkan di atas. Sebenarnya semua reporting tool tersebut dapat melakukan konversi output ke Excel, Word, dan PDF.

---

### **Tentang Penulis:**



#### **Paulus Iman**

Penulis saat ini bekerja sebagai Senior .NET Developer di salah satu perusahaan

konsultan IT di Jakarta Barat, mempunyai sertifikat MCTS dan MCSD, selalu mengikuti perkembangan .NET sejak versi 1.0.

---



# Komponen Akses Data Di Lingkungan Windows

Memahami model-model akses data yang tersedia di lingkungan Windows sangat penting dalam membuat sebuah aplikasi bermutu.

Pelajaran sejarah adalah salah pelajaran favoritku di sekolah, kecuali PSPB :-). Dalam artikel ini saya akan mencoba membahas sejarah perkembangan model komponen data akses (Data Access Component) di lingkungan Windows dan bagaimana evolusinya sampai sekarang... eh... evolusi kan pelajaran biologi ya? bukan sejarah... halah... Artikel ini akan membahas konsep Data Access Universal dan model akses data secara umum di Lingkungan Windows, membahas Data Access Providers dan Consumers, serta membahas model data akses dalam .NET framework. Selamat membaca...

## Sejarah Singkat Data Access Universal

Pada awalnya tidak ada keseragaman metode dalam mengakses data. Setiap sistem database memiliki API (Application Programming Interface) dan metode aksesnya sendiri-sendiri. Sebagai contoh Clipper dan Foxpro pada masa DOS, masing masing memiliki bahasa pemrogramannya sendiri, walaupun ada kesamaan tidaklah terlalu banyak. Kelihatannya tidak terlalu sulit, setiap programmer

hanya perlu memahami semua fungsi dalam API database yang dipakainya. Kesulitannya adalah ketika perlu berpindah ke sistem database yang lain. Pengetahuan tentang sistem database lama seakan tidak diperlukan lagi karena harus mempelajari sistem database yang baru dari awal lagi. Lama kelamaan hal seperti ini tidak lagi bisa dilakukan karena semakin banyaknya vendor yang menawarkan sistem database mereka sendiri, dengan segala kelebihan dan kekurangannya tentunya. Sulit bagi programmer untuk bisa memahami seluruhnya kecuali ada semacam standarisasi dalam metode akses database yang berbeda-beda tersebut. Berikut ini adalah standarisasi yang dilakukan dalam platform Windows.

## ODBC

Open Database Connectivity (ODBC) membantu programmer memanfaatkan berbagai sistem database tanpa perlu mengetahui detail API yang dari sistem database bersangkutan. Untuk dapat mencapai ini ODBC menyediakan model akses databasenya dan semua vendor sistem database cukup

mengimplementasikan model akses tersebut sehingga ODBC dapat mengakses sistem database bersangkutan. Implementasi model akses database ini biasa kita sebut sebagai driver. Dengan cara ini programmer hanya perlu mengetahui API dari ODBC saja untuk dapat mengakses bermacam-macam sistem database.

Dengan cara seperti ini, para programmer dapat dengan mudah berpindah dari satu sistem database ke sistem database lainnya dan memanfaatkan ketrampilan yang sudah mereka dapatkan sebelumnya. Lebih jauh lagi, programmer bisa membuat aplikasi yang tidak terikat pada satu sistem database tertentu. Pendekatan ini baik untuk aplikasi yang dibuat untuk konsumsi umum karena memberikan kebebasan bagi customer bersangkutan dalam memilih sistem database yang akan mereka gunakan, bisa jadi customer bersangkutan sudah menginvestasikan dana cukup besar untuk sistem





database tertentu dan tidak ingin mengeluarkan dana lagi untuk membeli sistem database lainnya.

ODBC adalah sebuah lompatan besar dalam mempermudah pengembangan aplikasi database. Namun bukan berarti tanpa kelemahan. Pertama, ODBC hanya mendukung akses untuk data yang terstruktur seperti sistem database relational. Untuk data yang tidak terstruktur, atau data dengan sistem hirarki seperti LDAP, kita tidak bisa menggunakan ODBC. Kedua, ODBC hanya dapat menangani perintah SQL, sebagai standar, dan hasilnya harus dapat direpresentasikan dalam bentuk baris dan kolom. Tapi secara umum ODBC bisa dikatakan sukses besar jika dibandingkan dengan kondisi sebelum ODBC diperkenalkan...

## OLE DB

Object Linking and Embedding Database (OLE-DB) adalah langkah berikutnya dalam standarisasi metode akses database, dan masih banyak

sekali digunakan saat ini. OLE-DB merupakan aplikasi dari pengetahuan yang didapatkan saat mengembangkan ODBC untuk membuat model akses data yang lebih baik. OLE-DB juga menandai perubahan strategi Microsoft untuk menggunakan API berbasis COM sehingga memudahkan OLE-DB digunakan dalam berbagai macam bahasa pemrograman, sekaligus perpindahan ke sistem operasi 32-bit dengan diluncurkannya Windows 95.

Seperti umumnya sebuah program, ODBC menjadi semakin besar dan gemuk karena berbagai macam perbaikannya. Sedangkan API yang disediakan OLE-DB jauh lebih ringkas serta memberikan performa yang lebih baik dibandingkan ODBC. Uniknya, saat pertama kali diluncurkan, OLE-DB hanya memberikan provider untuk ODBC. Provider OLE-DB untuk ODBC ini hanya sebuah wrapper terhadap API ODBC dan tidak memberikan perbaikan kinerja. Hal ini rupanya ditujukan agar para programmer menjadi

terbiasa dengan API yang baru, sambil menunggu tersedianya provider yang lebih efisien dibuat untuk mengakses sistem database langsung tanpa melalui ODBC.

## OLE-DB Providers

OLE-DB tidak terlalu bergantung pada struktur fisik sistem database yang diaksesnya, sehingga OLE-DB dapat mendukung sumber database relasional maupun hirarki. OLE-DB juga tidak mengharuskan query dalam bentuk SQL. OLE-DB terbagi dalam 2 bagian, yaitu OLE-DB Provider dan OLE-DB Consumer. ODBC mengharuskan para vendor database membuat provider khusus untuk sistem database mereka, dan ini bukan pekerjaan mudah. Namun dalam OLE-DB, jauh lebih mudah untuk membuat provider kita sendiri dengan format sumber data yang kita tentukan sendiri. Sebuah provider OLE-DB hanya perlu melakukan 4 langkah berikut ini:

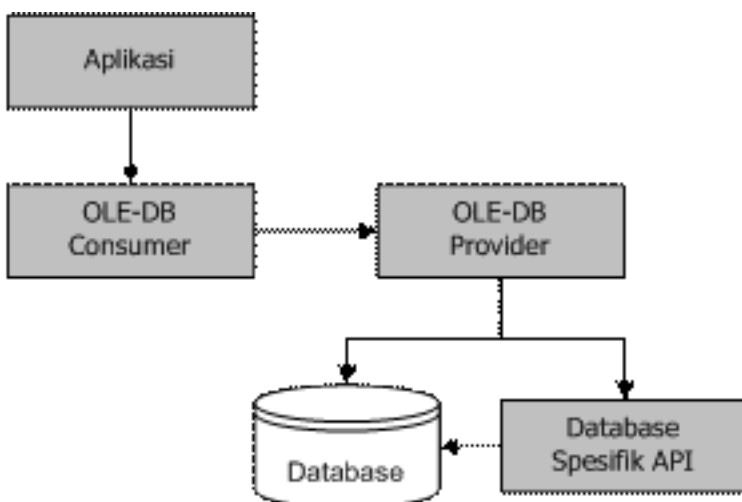
1. Buka sesi.
2. Proses perintah yang diberikan.
3. Akses data yang diminta.
4. Siapkan hasilnya.

## OLE-DB Consumers

Bagian kedua dari OLE-DB framework adalah OLE-DB consumer. Bagian ini merupakan lapisan yang berkomunikasi langsung dengan provider. Lapisan ini malukan langkah-langkah berikut:

1. Tentukan sumber datanya.
2. Buka sesi.
3. Berikan perintah.

**Gambar #1: OLEDB**



4. Kembalikan hasil yang didapat.

Gambar #1 menunjukkan bagaimana cara OLE-DB bekerja.

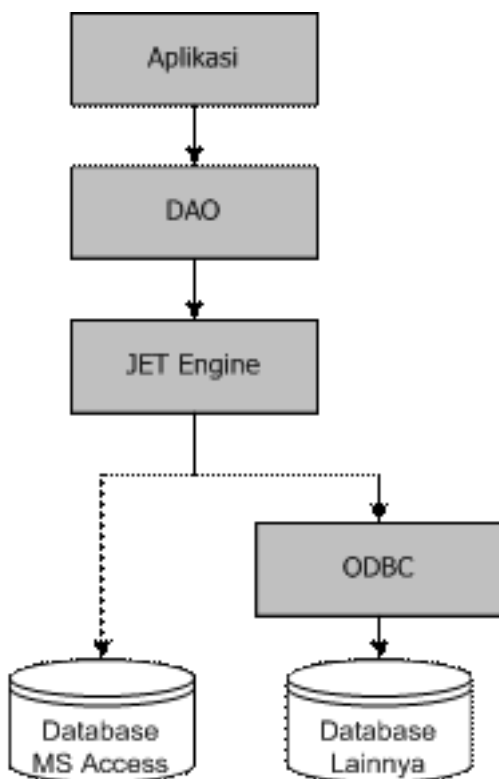
## Data Access Consumers

Para programmer yang menggunakan bahasa yang mendukung pointer – seperti C, C++ dan sebagainya – dapat berkomunikasi dengan database langsung menggunakan API yang disediakan ODBC atau OLE-DB. Namun untuk programmer yang menggunakan bahasa pemrograman yang tidak mendukung pointer – seperti Visual Basic -memerlukan lapisan lain. Disinilah DAO, RDO, ADO, and ADO.NET dapat berperan.

## DAO

Bersamaan dengan

**Gambar #2: DAO**



diluncurkannya Visual Basic 2.0, para programmer diperkenalkan dengan sebuah metode akses data baru yang dikenal sebagai Data Access Objects (DAO). Ini adalah usaha pertama Microsoft membuat Data Access Consumer. Walaupun versi awal kemampuannya sangat terbatas, Namun ini adalah awal sejumlah library yang memudahkan programmer dalam mengakses data karena sangat membantu programmer dengan bahasa tingkat tinggi seperti Visual Basic untuk memanfaatkan keunggulan ODBC. Sebelumnya ODBC API hanya bisa diakses melalui bahasa dengan level lebih rendah seperti C dan C++.

DAO dikembangkan menggunakan library JET. JET sendiri adalah library/engine yang digunakan oleh Microsoft untuk membangun aplikasi database Microsoft Access. DAO bersungsi sebagai lapisan yang menghubungkan aplikasi dengan database itu sendiri. Tujuannya tentu untuk memudahkan kerja programmer.

Versi awal dari DAO ini sudah dimasukkan dalam paket pemrograman Visual Basic 2.0, namun secara resmi baru diberikan nama DAO 1.0 ketika Microsoft Access 1.0 diluncurkan. Versi ini baru mendukung koneksi langsung ke database Access dan ODBC. Versi 2.0 kemudian diluncurkan untuk mendukung OLE-DB dan versi 2.5 dan 3.0 juga akhirnya diluncurkan untuk

mendukung ODBC 2.0 dan sistem operasi 32-bit yang baru diluncurkan, Windows 95.

Namun ada satu kelemahan mendasar yang dimiliki oleh DAO, yaitu DAO hanya dapat berkomunikasi langsung dengan JET engine. Untuk mengakses database selain Microsoft Access menggunakan ODBC maka harus berkomunikasi dahulu dengan JET engine. Hal ini membuat kinerja DAO dianggap kurang memuaskan.

## RDO

Remote Data Objects (RDO) adalah upaya Microsoft mengatasi lambatnya kinerja DAO. Saat perlu mengakses database selain MS Access maka RDO langsung berkomunikasi dengan ODBC tanpa menggunakan JET engine. Dengan cara ini kinerja RDO jauh meningkat dibandingkan dengan DAO. JET hanya digunakan ketika akan mengakses database MS Access.

Hal lain yang menjadi keunggulan RDO adalah fitur client-side cursor saat melakukan navigasi data. Berlawanan dengan DAO yang hanya mendukung server-side cursor. Fitur ini mampu mempercepat kinerja aplikasi dan mengurangi beban koneksi aplikasi pada server database.

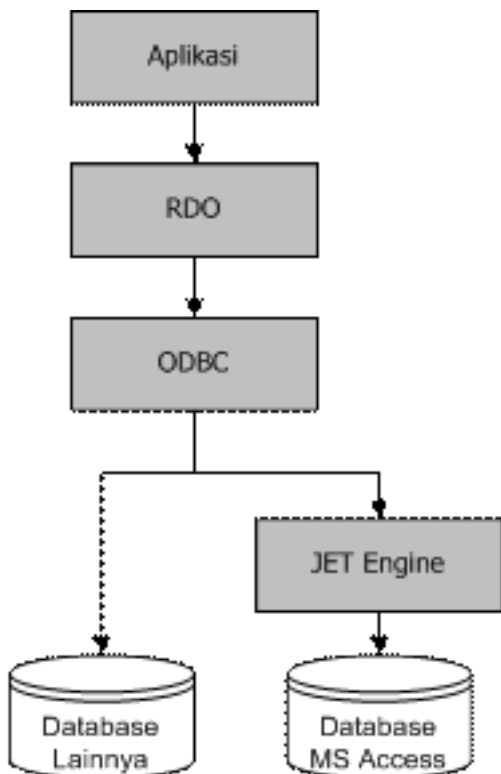
Namun secara umum RDO tidak mendapatkan tanggapan yang serius dari para pengembang perangkat lunak. RDO utamanya ditujukan untuk bagi customer komersil untuk membangun

aplikasi berskala enterprise, serta bagi para pengembang yang tidak mau menggunakan DAO karena kinerjanya yang buruk. Dengan alasan ini RDO hanya tersedia pada Visual Basic Enterprise Edition. Akibatnya tidak semua programmer memiliki akses untuk memanfaatkannya. Programmer yang membangun aplikasi kecil yang tidak terlalu bermasalah dengan kinerja juga enggan berpindah ke RDO. Ditambah lagi dengan diluncurkannya ODBCDirect – sebuah add-on untuk DAO yang mampu meningkatkan kinerja DAO – membuat popularitas RDO semakin berkurang di kalangan pengembang perangkat lunak.

## ADO

Microsoft memperkenalkan

**Gambar #3: RDO**



ActiveX Data Objects (ADO) untuk menyediakan API dengan level yang lebih tinggi saat memanfaatkan OLE-DB untuk berkomunikasi dengan database. Microsoft juga belajar banyak dari pengalaman sewaktu mengembangkan DAO dan RDO untuk membuat API yang lebih ringan, dan lebih efisien. ADO ditujukan untuk menggantikan DAO dan RDO. Pada saat awal diluncurkan ADO (dan OLE-DB) diyakini sebagai solusi untuk mengakses berbagai macam database sampai email, file teks bahkan spreadsheet.

ADO juga memperkenalkan metode baru dalam mengakses database. Pada masa DAO dan RDO, para programmer harus mengerjakan beberapa langkah yang cukup rumit sebelum bisa menjalankan sebuah perintah sederhana. Sebagai contoh, untuk menjalankan sebuah perintah SQL dalam RDO, programmer tidak bisa serta merta membuat obyek `rdoQuery`, tapi sebelumnya harus membuat obyek `rdoEngine`, kemudian `rdoEnvironment` sebagai child-nya, kemudian `rdoConnection` baru kemudian obyek `rdoQuery` bisa dibuat. Dalam DAO situasinya tidak jauh berbeda. Dalam ADO, semua langkah-langkah ini dipermudah. Programmer dapat langsung membuat obyek `command` dengan memberikan parameter `connection` dan langsung menjalankannya. Obyek `command` bahkan dapat berdiri sendiri walaupun obyek

connection belum dibuatkan.

Salah satu kelebihan lain adalah ADO menyediakan provider model. Provider model membuat ADO mampu berkomunikasi dengan sumber data selain OLE-DB. Model ini juga memudahkan pengembang perangkat lunak independen membuat provider mereka sendiri, provider ini nantinya akan digunakan ADO untuk berkomunikasi dengan sumber data bersangkutan. Sebagai contoh adalah provider ODBC. Ketika programmer ingin mengakses sumber data ODBC, maka ADO akan berkomunikasi langsung dengan ODBC, tidak melalui OLE-DB. Metode akses langsung ini semakin meningkatkan kinerja yang diberikan ADO dibandingkan dengan Data Consumer pendahulunya.

ADO juga memberikan sejumlah kemudahan bagi para programmer yang merasa kesulitan dengan model akses data yang diberikan DAO dan RDO. Kemudahan itu antara lain adalah:

- Kemampuan melakukan Batch Update — Untuk pertama kalinya perubahan data pada beberapa record sekaligus bisa dilakukan di memori terlebih dahulu, perubahan ini kemudian bisa dilakukan secara permanen ke database dengan perintah `UpdateBatch`.
- Dukungan Disconnected Data Access — dengan melakukan perubahan data tanpa terhubung dengan

server (disconnected state) mampu mengurangi beban pada server database, hasil perubahan kemudian dapat dilakukan secara permanen menggunakan Batch Update.

- Dukungan Multiple Recordsets — ADO juga mampu menjalankan query yang menghasilkan beberapa kumpulan data (multiple recordset) sekaligus. Pada ADO.NET fitur ini dikenal dengan nama Multiple Active Result Sets (MARS).

Walaupun begitu banyak perbaikan yang ditawarkan ADO, bukan berarti tidak memiliki kelemahan. Sebagai contoh adalah dukungan Disconnected Data Access yang tidak terlalu populer kalangan programmer, bahkan mungkin ada yang tidak mengetahuinya. Para

programmer lebih memilih untuk membiarkan koneksi database tetap terbuka sambil melakukan perubahan terhadap data.

Para programmer yang mencoba langsung menutup koneksi database setelah mengambil data mendapati bahwa mereka harus membuka kembali koneksi database kembali untuk meneruskan perubahan data di memori secara permanen ke database. Proses buka tutup koneksi database ini rupanya merupakan proses yang cukup memakan waktu sehingga justru menurunkan kinerja aplikasi secara keseluruhan. Sebagai akibatnya, para programmer lebih memilih membiarkan satu koneksi ke database tetap terbuka dan memanfaatkan koneksi ini untuk semua proses akses database pada aplikasi.

## Akses Data di .NET

Bersamaan dengan diluncurkannya .NET

Framework, Microsoft memperkenalkan model akses data baru yang diberi nama ADO.NET. Singkatan ActiveX Data Object sebenarnya tidak lagi relevan karena ADO.NET bukanlah komponen ActiveX, namun merupakan model yang benar-benar baru dibangun menggunakan .NET framework. Microsoft sepertinya mempertahankan nama ADO karena kesuksesannya yang

luar biasa.

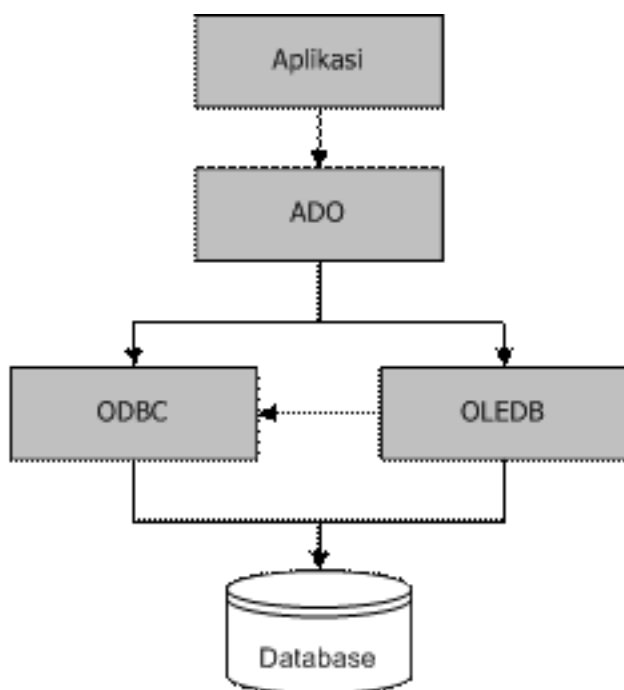
ADO.NET tetap memberikan dukungan komunikasi data menggunakan ODBC dan OLE-DB, namun menawarkan pilihan lain yaitu dengan menggunakan Data Provider yang khusus dibangun untuk database tertentu. Data provider ini mampu memberikan kinerja yang lebih baik karena dapat memanfaatkan optimisasi yang ditawarkan oleh vendor database bersangkutan. Dengan menggunakan custom-code yang spesifik untuk database tertentu juga menghilangkan sejumlah overhead yang terjadi ketika menggunakan generic-code seperti ODBC dan OLE-DB. Rilis awal yaitu ADO.NET 1.0, baru memberikan dukungan untuk Database SQL Server dan OLE-DB. Rilis berikutnya – ADO.NET 1.1 – Microsoft menambahkan dukungan untuk database Oracle dan ODBC. Pada saat yang hampir bersamaan Oracle juga merilis data provider milik mereka sendiri, yaitu ODP.NET 9.2.

Gambar berikut ini menggambarkan cara ADO.NET bekerja.

Berbeda dengan ADO yang menawarkan konsep recordset dan cursor, ADO.NET memperkenalkan model yang benar-benar baru meliputi lima obyek dasar sebagai berikut:

- Connection — berfungsi untuk membuat dan memelihara koneksi ke

**Gambar #4: ADO**





database. Parameter yang digunakan untuk membuat koneksi bisa jadi ada sedikit perbedaan antara satu database dengan database lainnya.

- Command — berfungsi untuk menyimpan query (perintah SQL) yang nantinya akan dikirimkan ke database termasuk dengan semua parameter yang diperlukan.
- DataReader — digunakan untuk membaca hasil query yang dikembalikan oleh database. DataReader hanya memberikan akses maju-saja (forward-only) namun sangat cepat untuk membaca seluruh data/record hasil query..
- DataSet — obyek inilah yang membuat ADO.NET sangat berbeda dengan metode data akses yang ada sebelumnya.

Obyek ini yang berada di memori dan bertindak sebagai tempat penyimpanan data/record yang didapat dari server database.

DataSet sendiri tidak bisa berkomunikasi langsung dengan server database dan tidak mengetahui dari mana data yang disimpannya berasal.

- DataAdapter — obyek inilah yang bertugas menjembatani DataSet dengan database sebenarnya. DataAdapter bertugas untuk menarik data/record dari database dan menyimpan kembali penambahan, perubahan atau penghapusan data/record pada DataSet kembali ke database.

ADO.NET membuat lompatan yang sangat besar. Salah satunya adalah adanya dukungan penuh untuk model akses data terputus (disconnected).

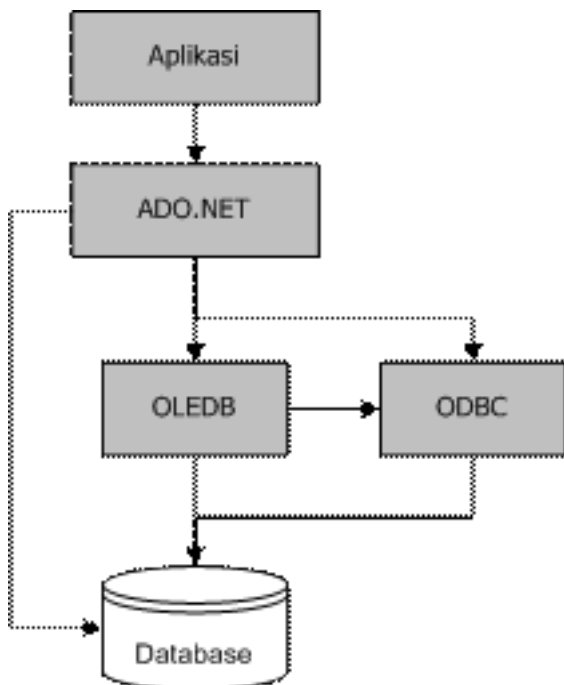
Mempertahankan sebuah koneksi yang tidak terputus ke database server adalah proses yang cukup mahal harganya. Terutama dari sisi database server yang mengharuskan pengalokasian sumber daya – terutama CPU dan memory – untuk koneksi tersebut. Ketika hanya ada satu koneksi saja yang perlu dipertahankan, mungkin tidak terlalu menjadi masalah, lain halnya jika koneksi yang perlu dipertahankan tidak sedikit, misalkan pada

aplikasi web, tentunya jumlah memory dan CPU yang terpakai akan jauh lebih besar. Namun jika kita menggunakan metode akses data yang terputus, sumber daya yang semula terpakai dapat langsung dibebaskan dan dipakai oleh proses lain.

Fitur lain yang meningkatkan performa akses data adalah diperkenalkannya connection pooling. Mempertahankan sebuah koneksi ke database adalah proses yang mahal, namun membuka dan menutup koneksi database adalah proses yang lebih mahal lagi. Connection pooling mampu mengatasi masalah ini. Ketika koneksi yang kita buat dalam program ditutup, ADO.NET tidak langsung menutup koneksi tersebut, namun menyimpannya dalam sebuah pool sampai jangka waktu tertentu. Pada saat itu ketika ada proses yang membutuhkan koneksi, maka ADO.NET tidak perlu lagi membuat koneksi baru, namun cukup menggunakan koneksi yang sudah tersedia di pool.

Keunggulan lain dari ADO.NET adalah dukungan terhadap XML. Secara internal obyek DataSet menyimpan data di memory dalam bentuk XML. Dukungan XML ini memudahkan ADO.NET dalam melakukan proses filtering dan sorting data yang tersimpan di memory. Dukungan XML juga memudahkan proses pengambilan data, penulisan data kembali ke database dan mengubah ke dalam format lainnya.

**Gambar #5: ADO.NET**



## ADO.NET 2.0

Perkembangan teknologi data akses sudah begitu canggih dengan adanya ADO.NET, namun rupanya Microsoft masih menemukan cara untuk mengembangkannya, yaitu dengan diperkenalkannya ADO.NET 2.0. Perubahan yang ada pada ADO.NET 2.0 tidak seradikal perubahan dari ADO ke ADO.NET, namun lebih berupa penambahan fitur-fitur baru, yang memudahkan para programmer dalam proses pengembangan aplikasi. Desain asal ADO.NET 2.0 masih sama dengan ADO.NET 1.x sehingga semua aplikasi yang sebelumnya dibuat menggunakan ADO.NET 1.x akan 100% kompatibel dengan ADO.NET 2.0.

Salah satu tujuan dari ADO.NET 2.0 adalah peningkatan performa. Bukan berarti performa ADO.NET 1.x buruk, namun peningkatan dapat dilakukan misalnya dengan

tambahan dukungan penulisan data beberapa row sekaligus (bulk insert) yang sebelumnya harus dilakukan satu persatu.

Fitur yang tambahan lainnya adalah notifikasi ketika ada perubahan data di database. Fitur lainnya adalah dukungan Multiple Active Result Sets (MARS), dengan cara ini, dengan satu query dapat memberikan beberapa hasil sekaligus, sehingga mengurangi proses komunikasi antar aplikasi dengan database.

ADO.NET 2.0 juga memudahkan programmer untuk membuat aplikasi yang mendukung beberapa produk database dengan menggunakan satu coding saja, artinya coding yang sama mampu mendukung beberapa sistem database sekaligus. Hal ini sangat berguna sekali ketika kita ingin aplikasi ini tidak hanya mendukung satu sistem database saja. Namun perlu diperhatikan, bahwa

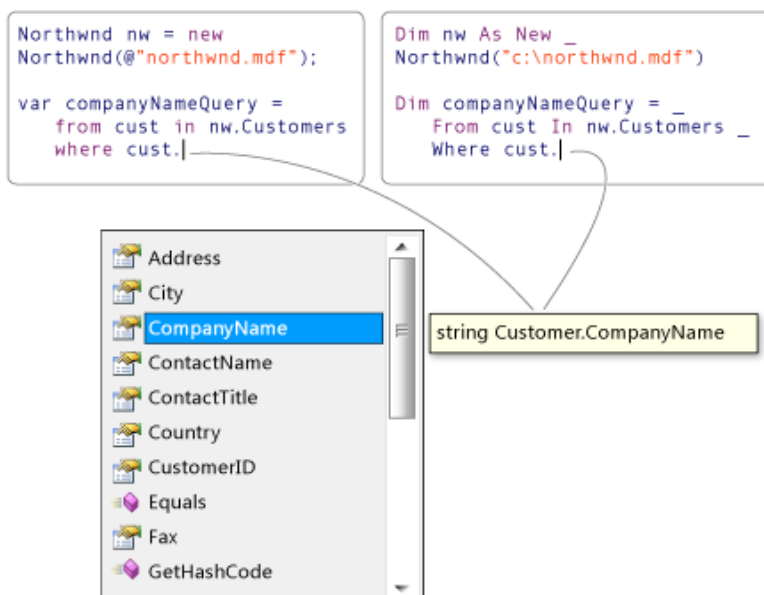
3.0 dirilis bersamaan dengan dirilisnya Visual Studio 2008 sebagai bagian dari .NET framework 3.5. Ada loncatan pada versi .NET framework dikarenakan .NET framework 3.0 sebelumnya sudah dirilis sebagai bagian dari Windows Vista, namun versi ADO.NET pada framework 3.0 masih versi 2.0. Salah satu fitur utama dari ADO.NET 3.0 adalah Object Relational Mapping (ORM) dan Language Integration Query (LINQ). Dengan menggunakan OR/M dan LINQ memungkinkan kita mengakses data secara lebih mudah dan transparan. Sebagai contoh, dalam .NET 2.0 kita masih sibuk dengan proses buka tutup koneksi ke database, maka hal ini tidak akan terlihat lagi dalam ADO.NET 3.0 berkat fitur ORM dan LINQ ini.

## LINQ

Language-Integrated Query (LINQ) adalah teknologi akses data yang diperkenalkan bersamaan dengan Visual Studio 2008 dan .NET Framework versi 3.5. LINQ ditujukan untuk mengisi celah antara ranah database dan ranah pemrograman berorientasi obyek.

Sebelumnya, query ke database hanyalah berupa string biasa berisikan SQL statement yang akan kita eksekusi, tidak ada pengecekan tipe data dan dukungan IntelliSense. Dengan adanya LINQ, query database menjadi bahasa yang native di C# dan Visual Basic.

**Gambar #6: Dukungan IntelliSense di LINQ**



sintaks SQL tetap spesifik untuk sistem database tertentu, misalkan sintaks Transact-SQL di SQL Server tidak bisa dijalankan di Oracle.

## ADO.NET 3.0

ADO.NET

## Entity Framework

Entity Framework adalah solusi untuk Object Relational Mapping sehingga memungkinkan para pengembang software melihat data sebagai kumpulan object dan properties, misalkan customers dan customer orders tanpa perlu memikirkan bagaimana bentuk table dan column di database.

Versi awal Entity Framework (EFv1) ada di .NET Framework 3.5 Service Pack 1 and Visual Studio 2008 Service Pack 1, yang dirilis pada tahun 2008. Versi ini banyak mendapatkan kritik karena kurang fleksibilitasnya, misalkan mengharuskan developer untuk membuat ulang Data Model.

Versi berikutnya langsung lompat menjadi Entity Framework

4.0 (EFv4), dirilis tahun 2010 bersamaan dengan Visual Studio 2010 dan .NET Framework 4.0 mencoba untuk mengatasi kekurangan yang ada di versi sebelumnya.

Versi ketiga adalah Entity Framework 4.1 memperkenalkan alternatif pendekatan penggunaan EF kepada para developer, yaitu Database First, Model First, atau Code First.

Entity Framework 5.0 diperkenalkan bersamaan dengan rilis Visual Studio 2012 dan ..NET framework 4.5.

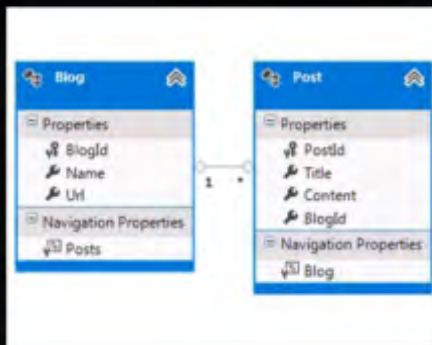
Dan paling mutakhir adalah Entity Framework 6.0 yang diperkenalkan bersamaan dengan rilis Visual Studio 2013. Dan sekarang Entity

Framework sudah dirilis sebagai project Open Source dengan Apache License v2. Source code dapat dilihat di CodePlex menggunakan Git. Peningkatan utama pada versi ini ada di fitur Code First.

## Tentang Penulis:

### Nur Hidayat

Penulis sudah mengikuti perkembangan .NET framework sejak versi 1.0 bersamaan dengan peluncuran Visual Studio 2002. Dan juga mengikuti perkembangan database SQL Server dan Oracle. Saat ini penulis bekerja sebagai Staff Operasional PLIK/MPLIK di PT Multi Data Rancana Prima.



```
public class Blog
{
    public int BlogId { get; set; }
    public string Name { get; set; }

    public List<Post> Posts { get; set; }
}

public class Post
{
    public int PostId { get; set; }
    public string Title { get; set; }
    public string Content { get; set; }

    public int BlogId { get; set; }
    public Blog Blog { get; set; }
}
```



### Model First

- Create model in designer
- Database created from model
- Classes auto-generated from model

### Code First (New Database)

- Define classes & mapping in code
- Database created from model
- Use Migrations to evolve database



### Database First

- Reverse engineer model in designer
- Classes auto-generated from model

### Code First (Existing Database)

- Define classes & mapping in code
- Reverse engineer tools available

# Mengenai ASP.net

ASP.Net adalah teknologi yang dikhususkan untuk mengembangkan aplikasi berbasis web dinamis dengan menggunakan platform .NET

**A**SP.Net adalah teknologi yang dikhususkan untuk mengembangkan aplikasi berbasis web dinamis dengan menggunakan platform .NET. ASP.Net berbeda dengan ASP Klasik. ASP.Net sudah ter-compile, sehingga lebih cepat dan lebih aman dibanding ASP.

ASP.Net mempunyai extension \*.aspx dan biasanya ditulis dengan menggunakan bahasa Visual Basic atau C#. ASP.Net mendukung control HTML, dan mempunyai control-control bawaan. ASP.Net mempunyai 3 jenis teknologi server, yaitu Web pages, Web Forms, dan MVC

## ASP.net Web Pages

Web pages adalah framework yang sederhana untuk membangun sebuah web dinamis. Web pages sudah mendukung razor syntax. Dengan web pages kita juga bisa membuat fungsi-fungsi helper. Beberapa keuntungan dari web pages ini adalah:

1. Sangat mudah untuk dipelajari dan dipahami
2. Mirip dengan ASP klasik dan PHP
3. Bisa server scripting dengan Visual Basic atau C#
4. Full HTML, CSS dan Javascript kontrol.

Jika ingin membangun single web yang sederhana, maka disarankan menggunakan web pages.

## ASP.net Web Forms

Salah satu mode pengembangan yang di support oleh ASP.Net yang menawarkan konsep self Postback dan ViewState. Postback maksudnya adalah melakukan post data dari sebuah form di halaman/page yang sama. Sedangkan ViewState dimaksudkan untuk memaintain nilai dari sebuah control sewaktu terjadinya Postback. Dengan Web Forms, Microsoft membawa model Visual Basic ke dunia web.

Kelebihan :

1. Support banyak server control (server control disini adalah: asp label, asp textbox, asp dropdown, asp gridview, asp listview dan lain-lain)
2. Support ViewState
3. Event driven programming yang didukung penuh oleh:
  - Code behind
  - Mekanisme self postback
  - ViewState.
4. Rapid application development

Kekurangan:

1. Project Architecture, bentuk



arsitektur dari project tidak fix. Suka-suka si developer

2. Tidak bisa dilakukan unit testing.
3. Performance.

Viewstate menjadi dua sisi yang berbeda. Dari segi performa akan mengurangi, dikarenakan viewstate ini disimpan di page itu sendiri dan mengakibatkan membengkaknya ukuran page.

## ASP.net MVC

MVC adalah sebuah pattern yang memperkenalkan konsep Model-View-Controller. Apa itu model? View? Controller? Nanti aja dibahas di sesi khusus ASP.Net MVC. Sekarang mari kita bahas dulu mengenai kelebihan dan kekurangan.

Kelebihan ASP.Net MVC :

1. Project Architecture yang lebih bagus, karena menggunakan konsep MVC.
2. Mendukung Test Driven Development dan tingkat Reusability yang lebih baik dibanding Web Forms. Karena controller tidak di bound ke view secara spesifik.
3. Performance yang yahud,



karena tidak support view state.

4. Karena tidak mendukung server control, maka pengembang mempunyai kekuasaan penuh terhadap HTML control. Ini sangat penting jika kita bekerja dengan framework javascript seperti JQuery dll.
5. Support Paralel Development.

### Membuat Project Baru

Bukalah visual studio anda (saya menggunakan visual studio 2010). Untuk memulai project baru, ikutilah langkah-langkah berikut:

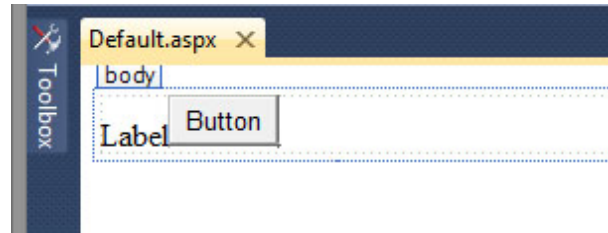
1. Klik menu File -> New -> Project
2. Setelah muncul dialog box, pada sidebar sebelah kiri, pilih Web (Visual Basic)
3. Kemudian pilih ASP.Net Web Application
4. Pada bagian Name, tuliskan nama project Anda.

Jika sudah selesai, tekanlah tombol OK, kemudian kita akan berada dalam window development. Disinilah kita mengembangkan web yang akan dibangun. Pada sidebar sebelah kiri ada toolbox, di sebelah kanan ada Properties dan Solution Explorer.

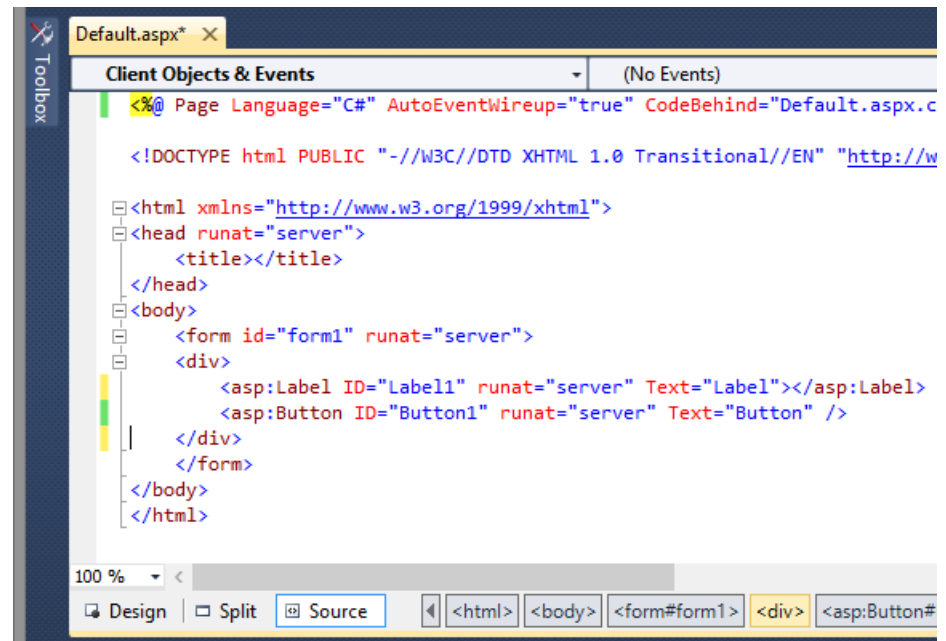
Biasanya, jika kita membuat sebuah project baru, maka di bagian solution explorer akan terdapat file Default.aspx. Untuk awal latihan, biarkan saja file ini bernama Default.aspx. Kalau di PHP, setara namanya dengan file index.php.

### First Code

Tambahkan dua buah control asp ke dalam content. Drag saja control Label dan control Button ke dalam site.



Form anda akan tampak seperti di atas jika ditampilkan dalam mode Design. Seperti pada GUI Dreamweaver, maka pada Visual Studio juga bisa ditampilkan dalam mode design dan mode source. Untuk mengubah tampilan dari mode design ke mode source, biasanya pada bagian kiri bawah terdapat tombol untuk mengganti view dari mode design ke mode source seperti di bawah ini.



Nah, sekarang kita akan melakukan scripting di codebehind. Pada solution explorer, double click pada Default.aspx.vb. Atau pada

source mode/design mode click kanan pada sebarang bagian, kemudian pilihlah View Code.

Jika sukses, maka kita akan masuk ke halaman code behind,

dimana kita bisa menuliskan code vb disini. Misalkan saya akan mengubah text pada label yang akan dilakukan pada saat form load. Gampang

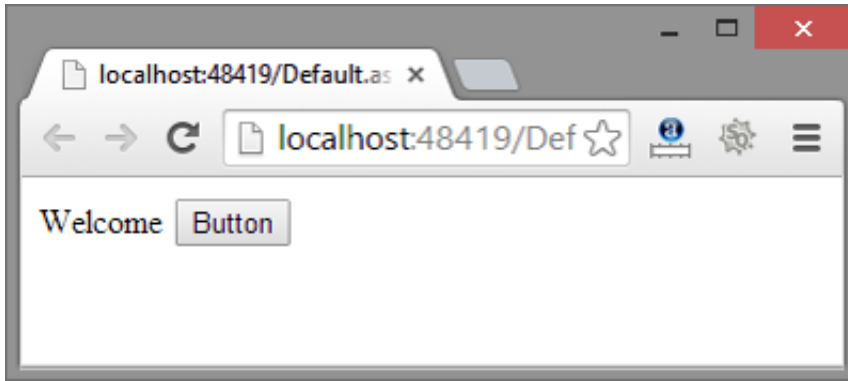
saja, kita akses event page load, kemudian kita set property text.

Jika sudah selesai, maka silahkan

```
Public Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
        Label1.Text = "Welcome"
    End Sub
End Class
```

tekan tombol "F5", atau Ctrl+F5 jika tidak ingin melakukan debug. Maka pada browser Anda akan muncul seperti ini...



So, sekarang kita ingin melakukan *sesuatu* pada saat kita menekan Button1. Silahkan akses event click pada Button1. Caranya bisa dengan melakukan **double click** pada Button1 di mode design. Kemudian kita tambahkan code berikut:

```
Protected Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    Label1.Text = "Button 1 telah di klik."
End Sub
```

Sudah dapat ditebak, maka pada saat Button1 di klik, maka text pada Label1 akan berubah menjadi seperti apa yang di assign pada code di atas.

Jika tidak ingin melibatkan control, dan ingin melakukan display variabel ke dalam element HTML, bisa juga kita lakukan seperti ini:

```
<div>
    <h2><% Response.Write("Ini adalah teks tanpa control") %></h2>
    <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
    <asp:Button ID="Button1" runat="server" Text="Button" />
</div>
```

Jika tanpa menggunakan Response.Write, akan seperti ini:

```
<div>
    <h2><%= "Ini adalah teks tanpa control" %></h2>
    <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
    <asp:Button ID="Button1" runat="server" Text="Button" />
</div>
```

## Mengenal ViewState

Kita telah sama-sama mengetahui kalau pada www selalu menggunakan protokol standar yang bersifat stateless. Jika setiap kali kita merequest halaman web, maka control tidak akan berisi apapun (stateless). Dengan View State, maka jika anda memiliki sebuah form, dan mengisi value pada textbox, kemudian anda pergi ke halaman lain, dan kembali lagi ke halaman tadi, maka value textbox tidak akan hilang. Kecuali View State pada control text tersebut di set False, maka text yang diisi sebelumnya akan hilang.

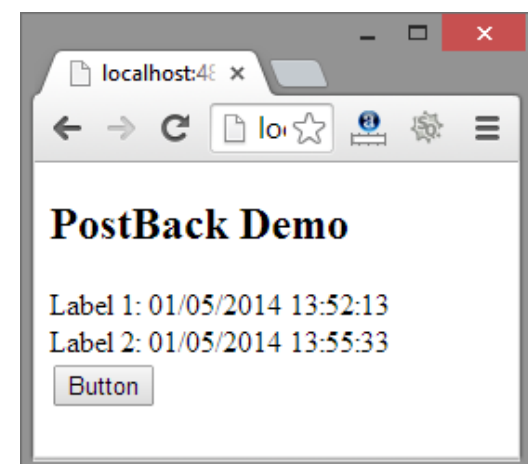
## Memahami Page PostBack

PostBack adalah situasi dimana sebuah halaman di load kembali oleh server namun **bukan** untuk pertama kali. Biasanya

ketika melakukan klik terhadap Button atau memilih item dalam DropDownList. Untuk lebih jelasnya lihat code berikut:

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    If Not IsPostBack Then
        Label1.Text = "Label 1: " & Now
    End If
    Label2.Text = "Label 2: " & Now
End Sub
```

Ketika kita menekan F5 pada Visual Studio, dan menampilkan hasil di browser, apakah Label1 dan Label2 memiliki nilai yang sama? Jawabnya ya, Label1 dan Label2 menghasilkan nilai yang sama. Nah, sekarang tekan tombol pada page, maka waktu yang ditampilkan pada Label1 dan Label2 akan berbeda karena page sudah di load untuk kedua kali, artinya sudah di-PostBack. Nilai pada Label1 sama dengan nilai Label1 pada saat di load pertama kali. Sedangkan nilai pada Label2 akan mengikuti waktu terkini pada saat halaman di refresh.



## Latihan 1 Membuat Calculator

Kita akan membuat sebuah calculator sederhana dengan menggunakan event click button.

Disini kita akan menggunakan Select-Case untuk menentukan operasi matematika yang dipilih user.

Buatlah 2 buah Textbox, 1 buah Dropdown, 1 Button dan 1 Label ke halaman aspx, sehingga kita punya sebuah file source Calculator.aspx seperti ini,

Nah, jika kita menjalankan pada browser, fungsi berjalan dengan baik. Akan tetapi ada kesalahan seperti terlihat pada gambar di samping ini

See, pada saat kita melakukan submit form, maka item dalam DropDownList akan terus bertambah. Untuk itu kita

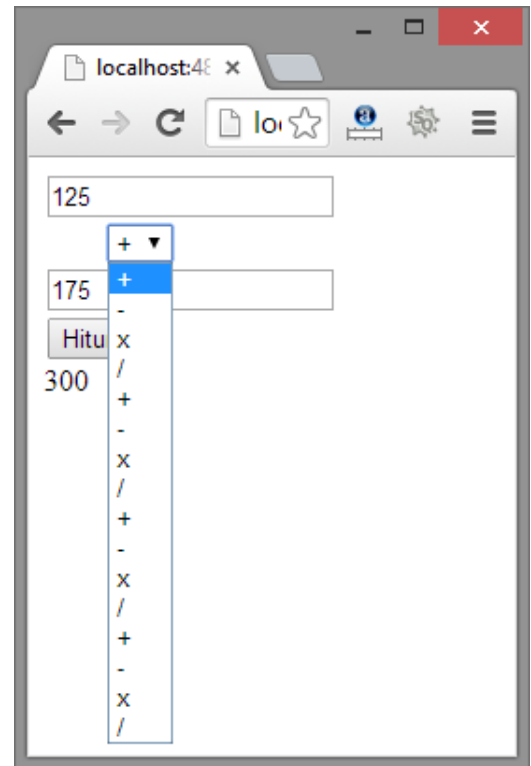
```
<div>
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox><br>
<asp:DropDownList ID="DropDownList1" runat="server"></asp:DropDownList><br>
<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox><br>
<asp:Button ID="Button1" runat="server" Text="Hitung" /><br>
<asp:Label ID="Label1" runat="server" Text="Hasil"></asp:Label>
</div>
```

Nah, sekarang saya ingin menambahkan Item ke DropDownList, ada banyak cara, semisal dari GUI kita bisa menambahkan Item. Tapi saya ingin menambahkan item pada event Page\_Load. Lengkapnya code nya adalah begini:

memerlukan sesuatu cara agar pengisian item ke DropDownList hanya dilakukan sekali saja di awal. Disinilah gunanya kita melakukan pengecekan, apakah halaman ini adalahPostBack atau tidak. So, pada event Page\_Load, akan saya modifikasi untuk

```
' Page Load
' inialisasi DropDownList
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    DropDownList1.Items.Add("+")
    DropDownList1.Items.Add("-")
    DropDownList1.Items.Add("x")
    DropDownList1.Items.Add("/")
End Sub

' Page Load
' Perhitungan aritmatika
Protected Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    ' Melakukan pengecekan terhadap jenis operasi aritmatika
    ' di DropDownList sesuai pilihan user
    Select Case DropDownList1.Text
        Case "+"
            Label1.Text = CInt(TextBox1.Text) + CInt(TextBox2.Text)
        Case "-"
            Label1.Text = CInt(TextBox1.Text) - CInt(TextBox2.Text)
        Case "x"
            Label1.Text = CInt(TextBox1.Text) * CInt(TextBox2.Text)
        Case "/"
            Label1.Text = CInt(TextBox1.Text) / CInt(TextBox2.Text)
    End Select
End Sub
```



melakukan pengecekanPostBack atau tidaknya. Sedikit modifikasi akan menjadi solusi bagi anda:

```
If Not IsPostBack Then
    DropDownList1.Items.Add("+")
    DropDownList1.Items.Add("-")
    DropDownList1.Items.Add("x")
    DropDownList1.Items.Add("/")
End If
```

### Profil Penulis:



**Gege Satya Putra, ST**

Lulusan Teknik Sipil UGM yang akhirnya nyasar ke dunia programming karena hobby.

Sempat bekerja sebagai software engineer di beberapa perusahaan IT di Medan dan menjadi pengajar di LP3I. Saat ini penulis aktif sebagai Staff MIS dan dosen di Wilmar Business Institute

**CODE #5: Aplikasi Kalulator Sederhana - Calculator.aspx**

```

<%@ Page Language="vb" AutoEventWireup="false" CodeBehind="Calculator.aspx.vb"
Inherits="WebApplication3.Calculator" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox><br />
            <asp:DropDownList ID="DropDownList1" runat="server"></asp:DropDownList><br />
            <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox><br />
            <asp:Button ID="Button1" runat="server" Text="Hitung" /><br />
            <asp:Label ID="Label1" runat="server" Text="Hasil"></asp:Label>
        </div>
    </form>
</body>
</html>

```

**CODE #6: Aplikasi Kalulator Sederhana - Calculator.aspx.vb**

```

Public Class Calculator
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
        If Not IsPostBack Then
            DropDownList1.Items.Add("+")
            DropDownList1.Items.Add("-")
            DropDownList1.Items.Add("x")
            DropDownList1.Items.Add("/")
        End If
    End Sub

    Protected Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        'Melakukan pengecekan terhadap posisi operator aritmatika di DrowpDownList
        Select Case DropDownList1.Text
            Case "+"
                Label1.Text = CInt(TextBox1.Text) + CInt(TextBox2.Text)
            Case "-"
                Label1.Text = CInt(TextBox1.Text) - CInt(TextBox2.Text)
            Case "x"
                Label1.Text = CInt(TextBox1.Text) * CInt(TextBox2.Text)
            Case "/"
                Label1.Text = CInt(TextBox1.Text) / CInt(TextBox2.Text)
        End Select
    End Sub
End Class

```



# One ASP.NET

## ASP.NET Web Stack

### Sites

MVC

Web  
Pages

Web  
Forms

Single  
Pages

Web  
API

SignalR

### Services



ASP.NET



Windows Azure



Windows Server 2012



Visual Studio

*Developers Summit 2013 Action!*

# POJOK

Edisi Perdana

# PROGRAMER

Majalah Elektronik

# Mengenai WPF: Windows Presentation Foundation

WPF merupakan teknologi penerus untuk pengembangan aplikasi berbasis windows.



Sebelum mulai membaca, perlu diketahui bahwa penulis bukanlah ahli, bahkan belum bisa dibilang mahir dalam menggunakan WPF. Akan tetapi di sini Penulis akan mencoba berbagi pengetahuan dan keterampilan mengenai WPF.

WPF merupakan kepanjangan dari Windows Presentation Foundation. WPF merupakan teknologi penerus untuk pengembangan aplikasi berbasis windows. Meskipun merupakan teknologi penerus, WPF hampir sama sekali tidak ada kaitan dengan teknologi sebelumnya yaitu WinForms. WPF dikembangkan terutama untuk memisahkan logika pemrograman (*business logic*) dengan desain tampilan (*User Interface*). Pada pemrograman WPF, tampilan didesain dengan bahasa XAML, sedangkan logika program yang sering disebut dengan "code behind" bisa menggunakan C# atau Visual Basic.

Berikut beberapa kelebihan WPF dibandingkan WinForms:

1. Semua tampilan pada

aplikasi WPF dirender dengan Direct3D, sehingga proses tampilan akan dikelola oleh GPU dan tidak memberatkan CPU.

2. Ukuran kontrol pada WPF menggunakan ukuran yang tidak bergantung dengan perangkat (*device-independent unit*) yang sebanding dengan 1/96 inci. Sehingga tampilan akan tampak sama meski pada resolusi monitor yang berbeda.
3. Pemisahan antara logika program dan desain tampilan.
4. Tampilan yang fleksibel dan mendukung templating.

Namun kelemahannya adalah, ketika aplikasi WPF dijalankan pertama kali setelah reboot, waktu loading awalnya lebih lama dibanding dengan aplikasi winforms. Hal ini diistilahkan sebagai "*cold start*".

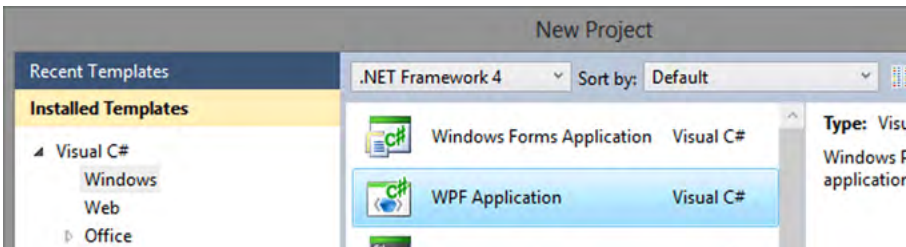
Kebanyakan orang yang telah membuat aplikasi dengan WPF, tidak ingin kembali lagi ke winforms. Menurut penulis pribadi, hal ini disebabkan karena

fleksibilitas dan kemampuan WPF yang menyebabkan hal-hal yang sulit dilakukan di winforms menjadi mudah dilakukan. Akan tetapi menguasai pemrograman WPF bukanlah hal yang mudah dilakukan, perlu banyak penyesuaian dan pembelajaran bahkan bagi programmer yang sudah berpengalaman sekalipun. Hal-hal yang wajib dikuasai sebelum memulai WPF antara lain:

- Menguasai konsep OOP (*Field, Properties, Method, Constructors, Inheritance, Interface*), serta konsep tentang *Event* dan *Delegates*
- Memahami XAML untuk desain antarmuka, meliputi Layout, Kontrol-kontrol pada WPF, Data Binding, dan Data Templating
- Memahami design pattern MVVM, seperti konsep *INotifyPropertyChanged*, dan *ObservableCollection*

Selain hal-hal di atas, terdapat juga topik-topik lain seperti *ValueConverter*, namespace pada XAML, *Dependency Properties* dan *Attached Properties*, animasi





dan sebagainya. Akan tetapi hal-hal tersebut dapat ditelusuri lebih lanjut setelah cukup mahir dasar WPF.

Untuk membuat aplikasi WPF, penulis sarankan untuk menggunakan minimal Visual Studio 2010 karena beberapa control penting seperti DataGrid dan DatePicker sudah tersedia. Untuk membuat aplikasi WPF dari Visual Studio 2010, pilih menu "new project", maka akan tampil window seperti gambar di bawah. Pastikan yang terpilih adalah "WPF Application", beri

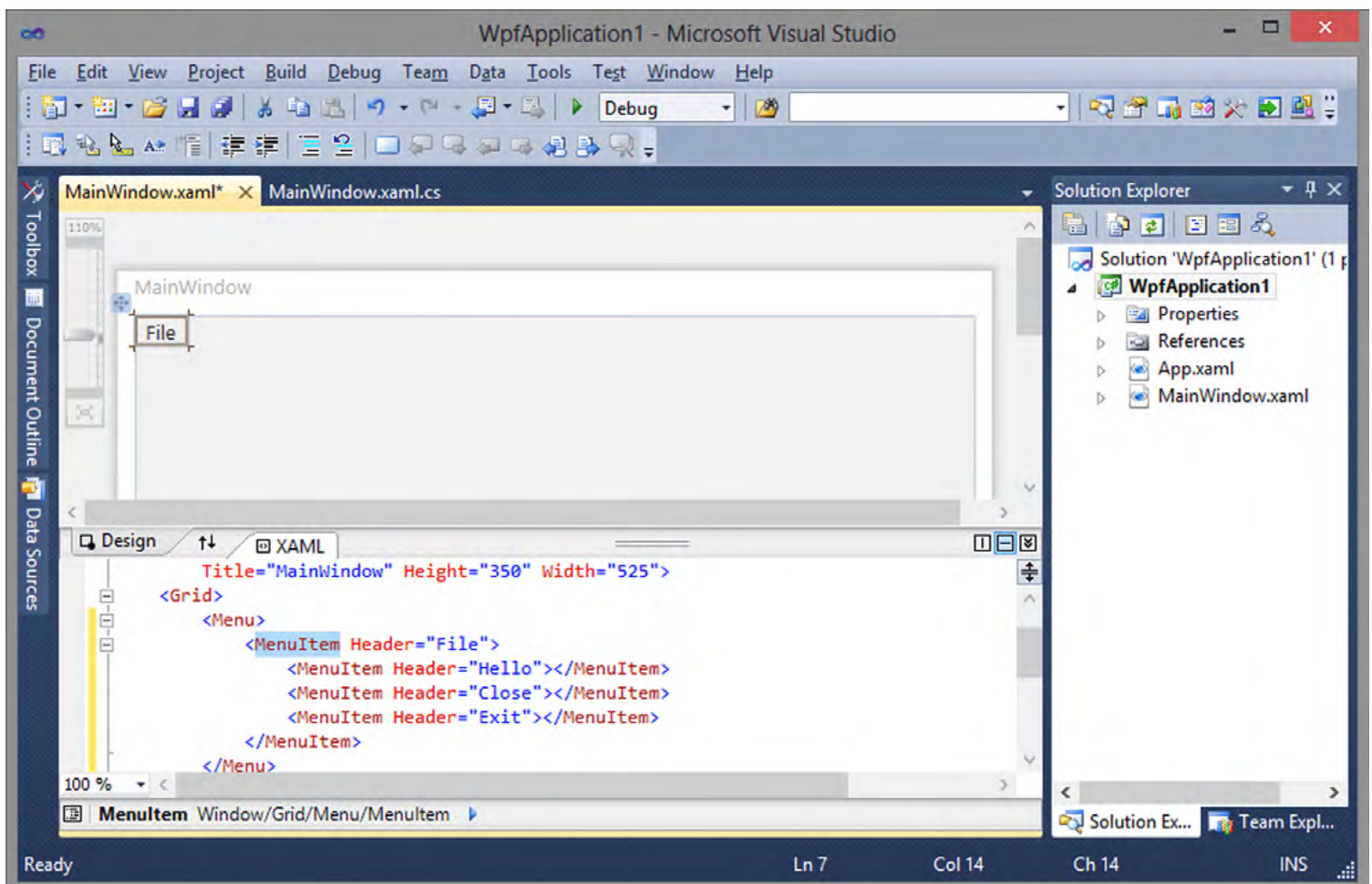
nama project dan klik OK, maka akan tampak tampilan window kosong dan kode XAML di bagian bawahnya.

Seperti yang telah diterangkan di atas, aplikasi WPF didesain menggunakan XAML. Berbeda dengan winforms yang harus mendrag kontrol dari toolbox, pada WPF, selain mendrag dari toolbox, kita juga dapat mengetikkan langsung kode XAML pada tempat yang sudah tersedia.

Apabila kita melihat kotak

"Solution Explorer", terdapat dua buah file, yaitu App.xaml dan MainWindow.xaml. Di sebelah kiri masing-masing file terdapat panah kecil yang jika diklik akan tampil App.xaml.cs dan MainWindow.xaml.cs untuk masing-masing file. File xaml adalah file berisi kode xaml yaitu desain UI aplikasi, sedangkan xaml.cs adalah file berisi code behind.

App.xaml adalah xaml yang berlaku untuk seluruh aplikasi. Selain untuk mengedit property "StartupUri" yang menyatakan window yang pertama kali tampil saat program dijalankan, App.xaml ini juga digunakan untuk menyimpan Style, Template atau resource yang berlaku untuk seluruh aplikasi.



Sekarang kita akan mencoba bermain-main sedikit dengan MainWindow.xaml. Untuk mendesain tampilan pada WPF, kita bisa menggunakan panel Toolbox untuk mendrag kontrol ke dalam window dan juga menggunakan panel Properties untuk mengedit properties kontrol. Akan tetapi biasanya lebih sering diperlukan untuk mengetik langsung kode XAML.

Untuk mendesain tampilan, penulis biasanya mengatur tampilan seperti tampak pada gambar di bawah, agar lebih mudah memantau kode XAML yang sedang dibuat. Sebelum mengakhiri tutorial kali ini, ada baiknya kita tambahkan sedikit interaksi pada program yang kita buat ini. Tambahkan kode program berikut di dalam grid (di antara <Grid> dan </Grid>):

```
<Menu>
  <MenuItem Header="File">
    <MenuItem Header="Hello"></MenuItem>
    <MenuItem Header="Close"></MenuItem>
    <MenuItem Header="Exit"></MenuItem>
  </MenuItem>
</Menu>
```

Kemudian pada tag pembuka MenuItem dengan `Header="Hello"`, ketikkan `Name="mnuHello"` dan kemudian `Click=""`. Pada intellisense yang muncul pada saat mengetik `Click`, pilih "<New Event Handler>" dan tekan TAB, sehingga baris MenuItem itu akan menjadi:

```
<MenuItem Header="Hello" Name="mnuHello" Click="mnuHello_Click"></MenuItem>
```

Pada saat mengetik TAB pada pilihan "<New Event Handler>", Visual Studio otomatis menambahkan event bernama

`mnuHello_Click` pada code behind. Untuk melihatnya, dapat menekan tombol F7 atau buka file MainWindow.xaml.cs dari Solution Explorer. Apabila langkah di atas telah diikuti dengan benar, maka di dalam file tersebut sudah terdapat event `mnuHello_Click`. Tambahkan `MessageBox` halo dunia pada event tersebut sehingga kode program event tersebut menjadi:

```
private void mnuHello_Click(object sender, RoutedEventArgs e) {
    MessageBox.Show("Halo Dunia");
}
```

Silahkan coba jalankan program yang telah dibuat (tekan F5), akan tampil menu File yang jika diklik akan tampil pilihan Hello, Close dan Exit. Coba klik tombol Hello, maka akan tampil pesan "Halo Dunia".

Jika dilihat dengan seksama, maka dapat dilihat bahwa menubar yang kita buat sebetulnya mengisi seluruh ruang dalam windows. Ini disebabkan

karena nilai default dari `HorizontalAlignment` dan `VerticalAlignment` apabila tidak disebutkan adalah "Stretch" yang mengakibatkan menubar mengisi seluruh ruang kosong secara horizontal maupun vertikal yang tersedia pada "Container" menubar tersebut.

Container yang saya maksud adalah elemen yang menampung menubar tersebut, dalam hal

ini adalah Grid. Agar menubar kita tampak seperti menubar pada umumnya, tambahkan `VerticalAlignment="Top"` pada tag pembuka Menu.

Selain kode program menu Hello, ada baiknya kita tambahkan juga kode program Exit yang juga cukup sederhana. Perlu DITEKANKAN juga di sini bahwa program yang kita buat ini sedikit melenceng dari teknik MVVM

yang dianjurkan bersama dengan WPF. Teknik ini lebih mirip teknik event pada WinForms. Akan tetapi penjelasan ini cukup untuk memberikan sedikit gambaran pemrograman WPF.

Kayaknya segini dulu perkenalan WPFnya. Untuk berikutnya, kita akan berkenalan dengan layout pada WPF, antara lain Grid, StackPanel, DockPanel, WrapPanel dan Canvas. Berikut seluruh kode program aplikasi yang dibuat pada tutorial ini:

## Profil Penulis:

### Fandi susanto

Penulis telah aktif di bidang pemrograman sejak tahun 2007. Sempat mengajar STMIK-MDP di Palembang tahun 2009-2012, saat ini mengkhususkan diri sebagai programmer freelance dan blogger pada situs [icodeformoney.com](http://icodeformoney.com)

Penulis dapat dihubungi melalui email [fandi.social@gmail.com](mailto:fandi.social@gmail.com)



**CODE #11: Hello World Rasa WPF - XAML Code**

```
<Window x:Class="latihanWPF.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525" WindowState="Maximized">
    <Grid>
        <Menu VerticalAlignment="Top" HorizontalAlignment="Stretch">
            <MenuItem Header="File">
                <MenuItem Header="Hello" Name="mnuHello" Click="mnuHello_Click"></MenuItem>
                <MenuItem Header="Close"></MenuItem>
                <MenuItem Header="Exit" Name="mnuExit" Click="mnuExit_Click"></MenuItem>
            </MenuItem>
        </Menu>
    </Grid>
</Window>
```

**CODE #12: Hello World Rasa WPF - Code Behind**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace latihanWPF {
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window {
        public MainWindow() {
            InitializeComponent();
        }

        private void mnuHello_Click(object sender, RoutedEventArgs e) {
            MessageBox.Show("Halo Dunia");
        }

        private void mnuExit_Click(object sender, RoutedEventArgs e) {
            Application.Current.Shutdown();
        }
    }
}
```

POJOKPROGRAMMER EMAGZ

# POJOK PROGRAMMER

Majalah Elektronik



## Nantikan Highlight di Edisi Berikutnya!

- Haruskan Move On dari VB6
- Membuat Laporan Tentang Crystal Report
- Anatomi SQL Statements
- Dasar-Dasar Pemrograman Java
- Membuat Sendiri Aplikasi SMS
- Aplikasi ASP.net MVC Sederhana

Seluruh dokumen di Pojok Programmer E-Magz dapat digunakan, dan disebarakan secara bebas untuk tujuan bukan komersial (non-profit), dengan syarat tidak menghapus atau mengubah atribut penulis dan pernyataan hak cipta yang disertakan dalam dokumen. Tidak diperbolehkan melakukan penulisan ulang atau modifikasi tulisan, kecuali mendapatkan ijin terlebih dahulu dari Penulis.