Berkenalan dengan LINQ pada VB 2008

Jerry Peter
Jerry.peter@gmail.com
http://www.ruangkecil.or.id

LINQ singkatan dari *Language Integrated Query*, LINQ sendiri merupakan fasiltias baru yang disertakan pada .NET Framework 3.5 dan telah terintegrasi jika kita menginstall Visual Studio 2008. Lalu apa bedanya sih dengan Query yang selama ini digunakan dalam database? Salah satu perbedaannya LINQ selain bisa melakukan query pada database juga dapat melakukan query terhadap data dalam format XML, Entities, Object dan sebagainya.

Secara singkat konsepnya sama dengan ODBC yang menjadi jembatan dalam mengakses database dalam berbagi format. LINQ disini juga menjadi jembatan perantara dalam mengakses berbagai format struktur data. Selama format data mendukung *IEnumerable Type* maka LINQ dapat melakukan query kedalamnya.

Kalau boleh dianalogikan mungkin akan lebih menyenangkan jika kita memiliki 1 buah kunci untuk masuk kedalam semua ruangan yang ada, jadi kita tidak perlu repot membawa/mempunyai banyak kunci untuk masuk kedalam ruangan yang berbeda. Dengan LINQ kita mempunyai sebuah kunci untuk dapat masuk dan melakukan Query kedalam berbagai format data yang berbeda.

Artikel ini akan membahas pengunaan LINQ untuk melakukan query kedalam berbagai format data seperti LINQ to XML, LINQ to Object, LINQ to SQL

Sebagai contoh, seperti biasa kita mulai ritual awal dengan coba membuat script “Hello World” untuk melihat proses LINQ.

```
Dim aKata() As String={"Hello","World","Visual","Studio","LINQ"}
Dim LINQ = From item In aKata _
           Select item
```

```
For Each k In LINQ
    Console.WriteLine(k)
Next
```

Baris 1 : proses deklarasi variable ARRAY String
Baris 2,3 : Sintaks pengunaan LINQ, untuk mencoba melakukan query kedalam format data ARRAY string yang telah di deklarasikan sebelumnya. Hasil proses query akan disimpan kedalam variable *LINQ*.
Baris 4,5,6 : Proses membaca data hasil query dengan LINQ dan menampilkannya pada console.

Dari contoh diatas kita lihat LINQ melakukan Query data kedalam sebuah ARRAY collection, secara sintaks juga mungkin tidak jauh berbeda dengan pengunaan QUERY pada SQL, perbedaanya keyword FORM digunakan diawal dan keyword SELECT digunakan pada bagian akhir. Selain Form dan Select kita juga dapat mengunakan WHERE dan ORDER seperti pada SQL Query untuk melakukan filter data dan pengurutan data.

WHERE
Berikut sample pengunaan WHERE dalam LINQ untuk melakukan filter data.

```
Dim aNumbers() As Integer = {0, 1, 2, 3, 4, 5, 6}
Dim LINQ = From num In aNumbers _
                Where num Mod 2 = 0 _
                Select num
For Each number In LINQ
    Console.Write(number & " ")
Next
```

Baris 1 : Pembuatan variable array integer
Baris 2,3,4 : Sintaks pengunaan WHERE pada LINQ untuk melakukan filter data, disini dilakukan filter terhadap nilai2 integer yg habis dibagi 2 dengan (MOD), dari proses ini hasil yg diperoleh hanyalah data2 bilangan genap.
Baris 5,6,7 : Proses loop pembacaan data dan menampilkan data pada console.

ORDER
Selanjutnya contoh berikut pengunaan ORDER dalam melakukan pengurutan data pada LINQ.

```
Dim aData() As Char = {"B", "D", "C", "A", "F", "E", "G"}
Dim LINQ = From item In aData _
            Order By item _
            Select item

For Each item In LINQ
    Console.Write(item & " ")
```

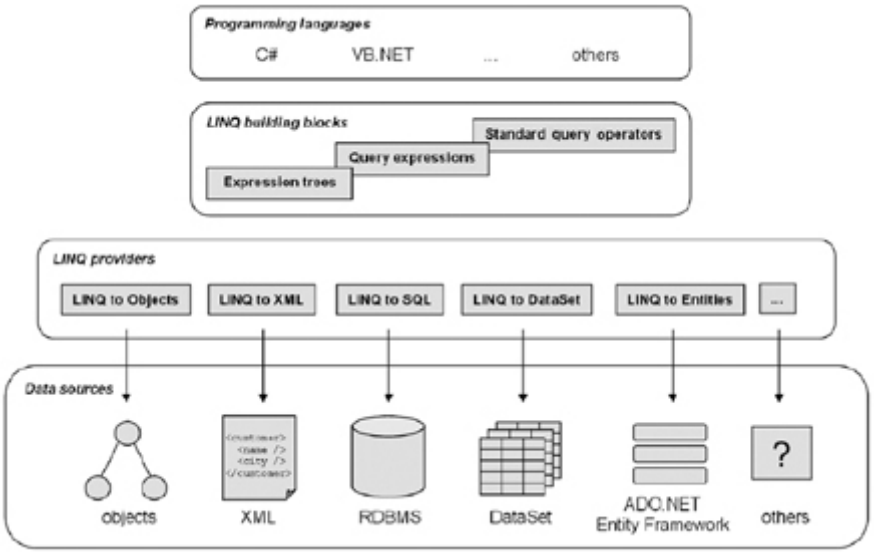
```
Next
```

Baris 1 : Pembuatan data Array character yang urutannya masih berantakan
Baris 2,3,4 : Melakukan proses query Array data dan melakukan Index pada hasil yang dapatkan.
Baris 5,6,7 : Proses loop pembacaan data dan memampilkan data pada console

Contoh2 diatas hanya sebagian proses pengunaan LINQ, hanya sebagai perkenalan awal bagaimana LINQ dapat melakukan Query kedalam format data selain Query yang biasa dilakukan dalam struktur database. Keyword-keyword yg digunakan pada LINQ ini juga mungkin hamper sama dengan penguaan Keyword pada Query dalam SQL, kalau dilihat perbedaanya hanya pada letak penulisannya saja.

Teknologi LINQ ini sendiri diperkenalkan pertama kali oleh Anders Hejlsberg dalam Microsoft Profesional Developers Conference (PDC) tahun 2005, tujuannya adalah membuat standarisasi dan memudahkan pattern proses pengaksesan data. Dengan adanya standart pengaksesan data maka para developer/programmer dapat melakukan cara yg sama dalam memproses beberapa format data (Database, XML ataupun Collection data object)

Berikut diagram arsitektur LINQ secara global dalam proses pengaksesan data.



Gambar 1
(LINQ Arsitektur)

Secara global terlihat LINQ dapat digunakan untuk melakukan query kedalam

beberapa format data berikut :

- LINQ to Object
- LINQ to XML
- LINQ to Database (SQL dan Dataset)
- LINQ to Entity

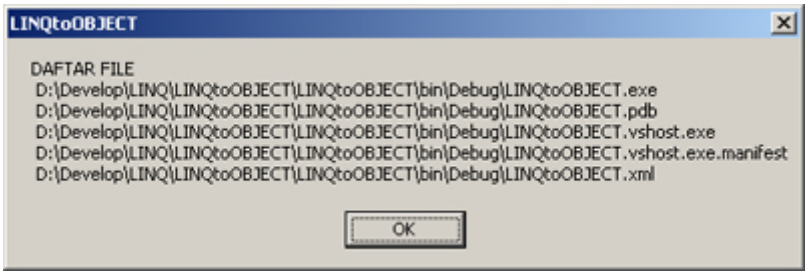
LINQ TO OBJECT

Sekarang kita coba kemapuan LINQ dalam melakukan proses query kedalam OBJECT collection (Array, ArrayList, HashTable, Directory, List dan juga user define collection).

Berikut sample pengunaan LINQ dalam mengakses system file pada computer kita.

```
Dim FileList = From file In My.Computer.FileSystem.GetFiles(CurDir())
_
               Select file

Dim hasil = "DAFTAR FILE" + vbCrLf
For Each file In FileList
     hasil = hasil + " " + file.ToString + vbCrLf
Next

MsgBox(hasil)
```

Proses diatas melakukan Query terhadap system File yang ada computer kita, dan kemudian menampilkan daftar hasil query yg didapatkan. Proses diatas akan menampilkan data seperti gambar berikut



Gambar 2
(Hasil proses LINQ to Object)

Local Type Inference

Sebelumnya lebih jauh membahas LINQ, kita berkenalan sebentar dengan istilah *Local Type Inference* dalam visual basic karena tehnik ini akan digunakan pada proses LINQ, istilah LOCAL TYPE INFENRENCE ini adalah kemampuan

menyimpulkan type data yang kita berikan kedalam sebuah variable berdasarkan nilai yang dimasukan.

Contoh:

```
Dim cData = "Jakarta – Indonesia"
Dim nNilai = 666
```

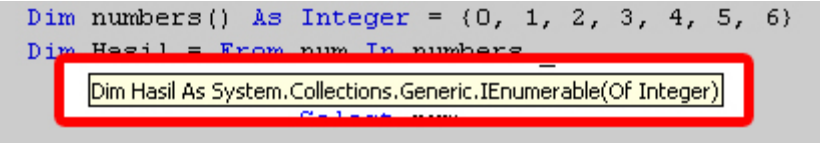
Dengan local type inference, maka secara automatis akan segera menyimpulkan bahwa variable *cData* memiliki type String dan *nNilai* memiliki type data Integer.

Pada sintaks LINQ kita akan sering menuliskan proses seperti berikut :

```
Dim Hasil = From num In numbers _
            Where num Mod 2 = 0 _
            Select num
```

Dari sintaks LINQ diatas kita mengunakan kemampuan Local type inference pada VB, dari proses diatas automatis variable akan berisi type data IEnumerable(T) collection walaupun kita tidak mendeklarasikan sebelumnya.

Catatan:
*IEnumerable(T), Maksud tanda **(T)** disini adalah type data sesuai hasil proses, bisa berupa IEnumerable (of String/char/integer/dsb)*
Jika kita mengarahkan cursor mouse keatas nama variable HASIL maka kita akan segera diinformasikan type data yang diberikan untuk variable tsb. Hal ini juga berlaku saat kita melakukan proses LINQ pada XML atau database.



Gambar 3
(Local type inference)

Sekarang kita kembali lagi kedalam pembahasan tentang LINQ to OBJECT, object-object apa saja sih sebenarnya yang dapat diproses mengunakan LINQ. Menurut MSDN semua object collection dapat kita query mengunakan LINQ, object2 ini antara lain Array, ArrayList, HashTable, Dirctory ataupun type collection object yg kita buat sendiri.

Sekarang kita coba saja satu persatu pengunaan LINQ pada object2 tersebut.

ARRAY COLLECTION
Berikut contoh pengunaan LINQ pada ARRAY Collection, disini kita akan membuat

sebuah variable array berisi nama-nama hari, kemudian kita melakukan query untuk data array yang memiliki huruf depan “S” saja.

```
      Dim aHARI() As String = _
      {"senin",   "selasa",   "rabu",   "kamis",   "jum'at",   "sabtu",
"minggu"}

      Dim LINQ = From item In aHARI _
                 Where Microsoft.VisualBasic.Left(item, 1) = "s" _
                 Select item

      Dim hasil = "HASIL QUERY" & vbCrLf
      For Each item In LINQ
          hasil = hasil & item & vbCrLf
      Next

      MsgBox(hasil)
```

Dari proses diatas akan dihasilkan nilai “Senin, Selasa, Sabtu” pada messagebox.

Gambar 4
(LINQ to ARRAY)

ARRAYLIST COLLECTION

Sekarang kita coba mengunakan LINQ pada ARRAYLIST Collection, contoh berikut kita akan membuat array list berisi nama-nama kota, kemudian kita melakukan query untuk kota2 yang berhuruf depan “J”.

```
        Dim aKOTA As New ArrayList
        aKOTA.Add("Jakarta")
        aKOTA.Add("Bandung")
        aKOTA.Add("Malang")
        aKOTA.Add("Jambi")
        aKOTA.Add("Surabaya")
        aKOTA.Add("Medan")

        Dim LINQ = From item In aKOTA _
                   Where Microsoft.VisualBasic.Left(item, 1) = "J" _
```

```
                Select item

        Dim hasil = "HASIL QUERY KOTA" & vbCrLf
        For Each item In LINQ
            hasil = hasil & item & vbCrLf
        Next
        MsgBox(hasil)
```

Dari hasil diatas akan terdapat data “Jakarta dan Jambi” pada messagebox.

Gambar 5
(LINQ to ARRAY LIST)

HASHTABLE COLLECTION

Berikut contoh query data LINQ pada HASHTABLE Collection, disini kita akan membuat variable HashTable yg berisi data2 String berikut keyed untuk masing2 data tsb, kemudian kita akan mengquery data untuk keyed yg lebih dari 3 saja.

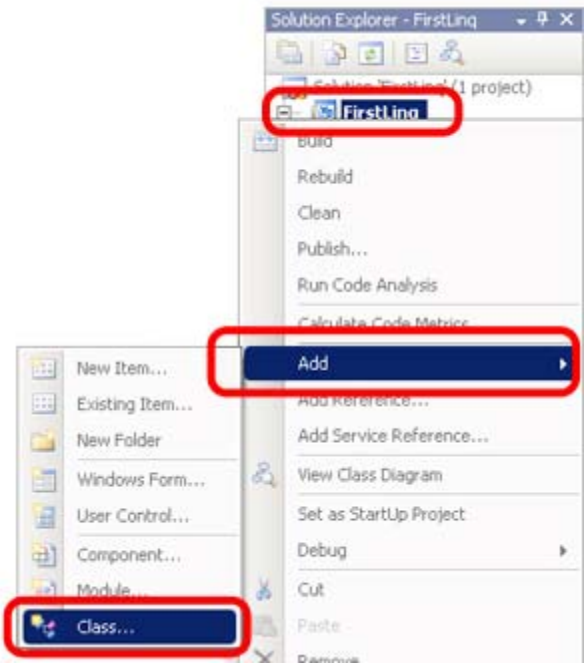
```
    Dim oHashTable As New Hashtable
    oHashTable.Add(1, "Jerry")
    oHashTable.Add(2, "Peter")
    oHashTable.Add(3, "Visual")
    oHashTable.Add(4, "Studio")
    oHashTable.Add(5, "Hash")
    oHashTable.Add(6, "Table")
    oHashTable.Add(7, "LINQ")

    Dim LINQ = From item In oHashTable _
               Where item.key > 3 _
                 Order By item.key _
               Select item

    Dim HASIL = "HASIL QUERY HASHTABLE" & vbCrLf
    For Each item As DictionaryEntry In LINQ
      HASIL=HASIL & "KEY:" & item.Key & " VALUE:" & item.Value &
vbCrLf
    Next

    MsgBox(HASIL)
```

Dari proses tersebut akan dihasilkan data “Studio, Hash, Table, LINQ” pada messagebox yang muncul.

Gambar 6
(LINQ to HashTable)

USER DEFINE ARRAYLIST COLLECTION

Proses terakhir kita coba mengunakan LINQ untuk mengakses sebuah Type data yang collection yang kita buat sendiri. Pada contoh berikut kita coba membuat sebuah class object baru PRODUCT, untuk membuat class baru ini click kanan pada nama project yang ada pada Solution explorer → add → class.



Gambar 7
(Menambah class kedalam project)

Berikan nama class tersebut PRODUCT.VB, Kemudian buat struktur class dengan 4 buah property dengan sintaks berikut

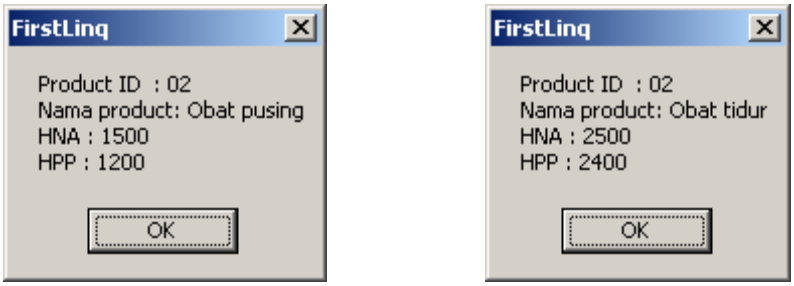
```
Public Class product
    Public ProductID As String
    Public Nama As String
    Public Supplier As String
    Public HNA As Integer
    Public HPP As Integer
End Class
```

Sekarang kita kembali pada form project kita, tambahkan sebuah tombol kedalam form dan isi script proses berikut pada event Click tombol yang baru dibuat.

```
        Dim product1 As New product With {.ProductID = "01", _
                                          .Nama = "Obat batuk", _
                                          .Supplier = "SUPPLIER_A", _
                                          .HNA = 1000, _
                                          .HPP = 900}
        Dim product2 As New product With {.ProductID = "02", _
                                          .Nama = "Obat pusing", _
                                          .Supplier = "SUPPLIER_A", _
                                          .HNA = 1500, _
                                          .HPP = 1200}
        Dim product3 As New product With {.ProductID = "02", _
                                          .Nama = "Obat tidur", _
                                          .Supplier = "SUPPLIER_B", _
                                          .HNA = 2500, _
                                          .HPP = 2400}

        Dim aProduct As New ArrayList()
        aProduct.Add(product1)
        aProduct.Add(product2)
        aProduct.Add(product3)

        Dim LINQ = From item As product In aProduct _
                   Where item.HPP > 1000 _
                   Select item

        For Each item In LINQ
            MsgBox("Product ID  : " & item.ProductID & vbCrLf & _
                   "Nama product: " & item.Nama & vbCrLf & _
                   "HNA : " & item.HNA & vbCrLf & _
```

```
                "HPP : " & item.HPP)
        Next
```

Baris 1 – 15 : Membuat 3 buah variable baru berikut nilainya dari class product.vb
Baris 16 – 19 : Membuat sebuah variable ArrayList dan menambahkan 3 buah variable product yg telah dibuat sebelumnya.
Baris 21 – 23 : Melakukan proses LINQ untuk mengambil data product yg memiliki HPP lebih besar dari 1000.
Baris 24 – 29 : Melakukan proses loop membaca data hasil proses LINQ sebelumnya, dari proses ini seharusnya muncul 2 kali messagebox yang menampilkan data product 001 dan 002.



Gambar 8
(LINQ to Custom user define class)

Contoh diatas hanya gambaran mengunakan User define collection, dan mengakses -nya mengunakan LINQ, pada real applikasi seharusnya lebih effektif mengunakan database untuk kasus pembuatan daftar barang ☺

Masih ada beberapa fasilitas lain yang dapat kita gunakan dalam LINQ to Object seperti JOIN, AGGREGATE, DISTINCT, GROUP dsb.

LINQ to XML

Kemampuan LINQ berikutnya adalah melakukan Query kedalam format struktur data XML, sebagai contoh sederhana kita mempunyai XML yang berisi contact seperti berikut:

```
<?xml version="1.0" encoding="utf-8" ?>
<Contact>
  <ContactPerson id="01">
    <nama>Jerry Peter</nama>
    <email>jerry.peter@gmail.com</email>
    <blog>www.ruangkecil.or.id</blog>
  </person>
  <ContactPerson id="02">
    <nama>Jerry</nama>
    <email>jerry@gmail.com</email>
    <blog>www.jerry.com</blog>
  </ContactPerson >
  <ContactPerson id="03">
    <nama>peter</nama>
    <email>peter@yahoo.com</email>
    <blog>www.peter.com</blog>
  </ContactPerson >
</Contact>
```

Dari format diatas terlihat masing-masing *contactPerson* memiliki (Nama, email dan blog), selanjutnya misalkan kita ingin mengambil data NAMA saja dari struktur XML contact diatas. Yang perlu kita lakukan adalah melakukan load data XML tersebut kedalam sebuah variable kemudian dengan bantuan LINQ kita tuliskan sintaks sederhana berikut:

```
Dim LINQ = From nama In ContactXML...<nama> _
           Select nama.value
```

Sederhana sekali bukan ☺

Mungkin terlihat keyword yang baru dari sintaks diatas yaitu tanda … (titik titik titik)
Keyword ini terdapat pada VS 2008 dan dapat kita gunakan untuk mengambil data TAG ELEMENT XML yang langsung kita perlukan, dari contoh diatas kita mengambil semua element dengan TAG <NAMA> dari struktur data XML yang ada.

Jika pernah mengunakan pembacaan struktur DOM (Document Object Model) mungkin keyword … ini sama dengan pengunaan pembacaan dengan method *getElementByTagName*.

Pada VS 2008 kita juga dapat menuliskan langsung struktur format XML kedalam

sebuah variable dengan cara berikut:

```
Dim ContactXML = <?xml version="1.0" encoding="utf-8"?>
                 <Contact>
                     <ContactPerson id="01">
                         <nama>Jerry Peter</nama>
                         <email>jerry.peter@gmail.com</email>
                         <blog>www.ruangkecil.or.id</blog>
                     </ContactPerson>
                     <ContactPerson id="02">
                         <nama>Jerry</nama>
                         <email>jerry@gmail.com</email>
                         <blog>www.jerry.com</blog>
                     </ContactPerson>
                     <ContactPerson id="03">
                         <nama>peter</nama>
                         <email>peter@yahoo.com</email>
                         <blog>www.peter.com</blog>
                     </ContactPerson>
                 </Contact>
```

Dengan penulisan diatas maka secara automatis variable ContactXML akan memiliki format type data XDocument, dan juga langsung memapung data format XML yang kita tuliskan.

Jika diperhatikan lagi penulisan nilai data diatas kita tidak perlu memisahkan masing-masing baris dengan mengunakan separator " _ " dan VS2008 tetap mengetahui maksud data selanjutnya masih tetap bagian dari format XML yang sedang kita tuliskan. Dan jika kita mencoba menuliskan sebuah TAG baru kedalam format XML yg sedang dituliskan VS2008 juga secara automatis membuatkan TAG penutup untuk kita.

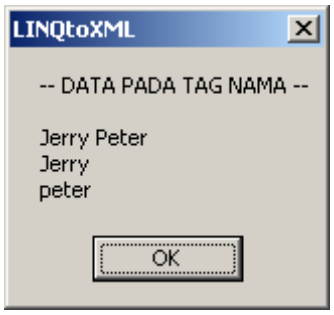
Sekarang coba kita gabungkan script diatas kedalam sebuah contoh project sederhana pembacaan XML dengan LINQ, pertama buatlah sebuah project Visual Basic pada Visual Studio kemdian tambahkan sebuah tombol kedalam form yang ada, dan tuliskan script berikut kedalam event click tombol tersebut.

```
Dim ContactXML = <?xml version="1.0" encoding="utf-8"?>
                 <Contact>
                     <ContactPerson id="01">
                         <nama>Jerry Peter</nama>
                         <email>jerry.peter@gmail.com</email>
                         <blog>www.ruangkecil.or.id</blog>
                     </ContactPerson>
                     <ContactPerson id="02">
                         <nama>Jerry</nama>
                         <email>jerry@gmail.com</email>
                         <blog>www.jerry.com</blog>
```

```
        </ContactPerson>
        <ContactPerson id="03">
            <nama>peter</nama>
            <email>peter@yahoo.com</email>
            <blog>www.peter.com</blog>
        </ContactPerson>
    </Contact>
```

```
Dim LINQ = From nama In ContactXML...<nama> _
           Select nama.Value

Dim hasil = "-- DATA PADA TAG NAMA --" & vbCrLf & vbCrLf
For Each nama In LINQ
    hasil = hasil & nama.toString & vbCrLf
Next

MsgBox(hasil)
```

Setelah selesai coba RUN project tersebut, dan click tombol yang ada. Jika lancar seharusnya akan muncul MessageBox dengan data seperti gambar berikut:



Gambar 9
(LINQ to XML)

Perlu diperhatikan sedikit penulisan SELECT pada LINQ diatas kita menuliskan sintaks berikut, dimana kita langsung mengambil VALUE dari tag <nama>

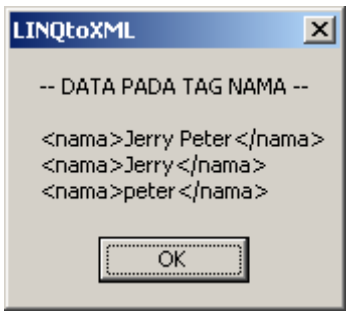
```
Dim LINQ = From nama In ContactXML...<nama> _
           Select nama.Value
```

Jika kita ganti script diatas dengan script berikut

```
Dim LINQ = From nama In ContactXML...<nama> _
```

```
Select nama
```

Dan coba jalankan kembali program tersebut maka akan ditampilkan hasil seperti gambar berikut:



Gambar 10
(LINQ to XML)

Data yang dihasilkan akan berisi semua TAG NAMA secara lengkap berikut dengan data TAG yang XML-nya. Pengunaan VALUE sebelumnya mungkin sama dengan penulisan innerHtml jika kita pernah mengunakan pembacaan dengan DOM.

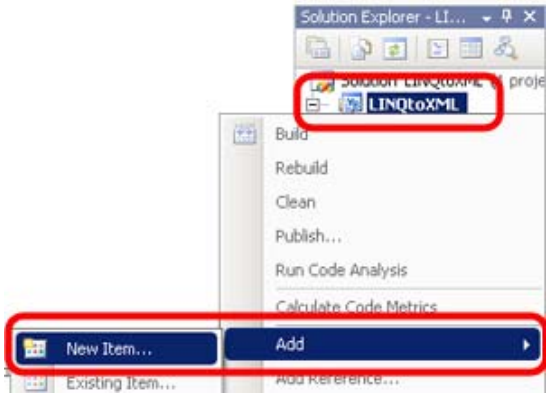
Load XML data

Selanjutnya kita mencoba melakukan pembacaan data XML pada sebuah file terpisah bukan dengan penulisan langsung seperti pada contoh sebelumnya.

Untuk melakukan LOAD data XML kedalam sebuah variable kita akan mengunakan fasilitas *load* pada object *XDocument*, penulisan sintaksnya seperti berikut:

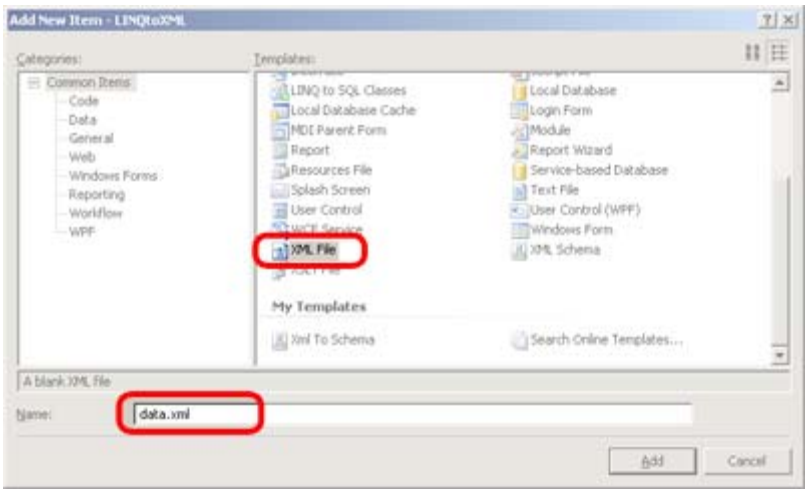
```
Dim <nama variable> = XDocument.Load(<URL data xml>)
```

Untuk mencoba kita gunakan contoh sebelumnya, namun sekarang kita coba pisahkan data XML kedalam file terpisah. Untuk membuat XML data pada file terpisah kita perlu menambahkan sebuah file XML dari menu Solution Explorer Click kanan pilih ADD → new item.



Gambar 11
(Menambah item baru kedalam project)

Kemudian pilih XML File untuk tipe document yg akan ditambahkan, dan beri nama DATA.XML untuk contoh ini. Setelah selesai tuliskan ulang struktur XML sebelumnya ada cukup di copy dan paste ulang data sebelumnya.



Gambar 12
(menambah XML File)

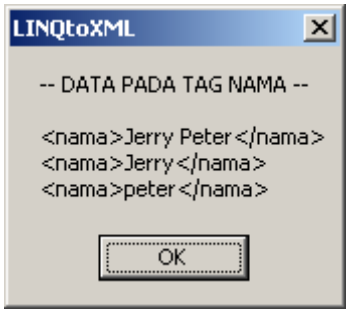
```
<?xml version="1.0" encoding="utf-8" ?>
<Contact>
  <ContactPerson id="01">
    <nama>Jerry Peter</nama>
    <email>jerry.peter@gmail.com</email>
    <blog>www.ruangkecil.or.id</blog>
    </ContactPerson>
```

```
    <ContactPerson id="02">
      <nama>Jerry</nama>
      <email>jerry@gmail.com</email>
      <blog>www.jerry.com</blog>
    </ContactPerson>
    <ContactPerson id="03">
      <nama>peter</nama>
      <email>peter@yahoo.com</email>
      <blog>www.peter.com</blog>
    </ContactPerson>
  </Contact>
```

Dari form yang ada sebelumnya tambahkan sebuah tombol baru kedalamnya dan tuliskan script berikut kedalam event click tombol yg baru ditambahkan untuk melakukan LOAD data XML dari file terpisah dan melakukan pembacaan data didalamnya.

```
        Dim contactXML = XDocument.Load(CurDir() & "\data.xml")

        Dim LINQ = From nama In contactXML...<nama> _
           Select nama

        Dim hasil = "-- DATA PADA TAG NAMA --" & vbCrLf & vbCrLf
        For Each nama In LINQ
            hasil = hasil & nama.ToString & vbCrLf
        Next

        MsgBox(hasil)
```

Selanjutnya coba jalankan dan kembali project tersebut dan tekan tombol yang baru ditambahkan, jika tidak ada masalah seharusnya hasil masih sama dengan contoh sebelumnya dengan menampilkan MSGBOX baru berisi data TAG NAMA.



Gambar 13
(Hasil proses LINQ to XML)

Catatan:
Saat menekan tombol baru yang dibuat sebelumnya, mungkin muncul pesan error seperti berikut :

Gambar 14
(Error message)

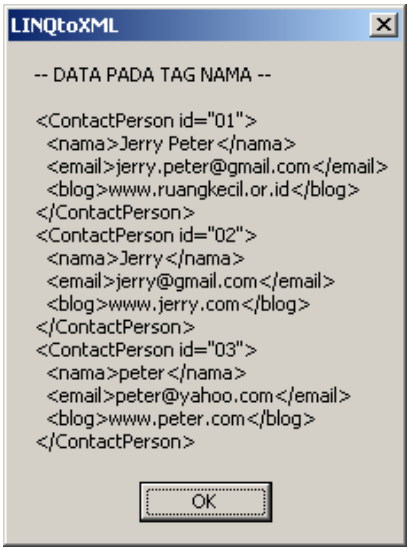
Tenang kesalahan bukan pada script kita, jika diperhatikan proses program curdir() pencarian data XML mengacu pada folder debug bukan pada root program project kita, hal ini karena saat melakukan debugging VS menjalankan program temporary yang dibuat pada folder [NamaProject] → BIN → [Debug], bukan pada folder root project.

Untuk mengatasi error tsb kita cukup copy saja DATA.XML kita kedalam folder tersebut untuk keperluan selama proses debugging.

Dari contoh sebelumnya kita coba gantikan proses LINQ yg ada dgn script berikut, dan coba jalankan kembali dan perhatikan hasil yang ditampilkan:

```
Dim LINQ = From nama In contactXML...<ContactPerson> _
   Select nama
```

Hasil proses dengan script pembacaan tag element <ContactPerson>



Gambar 15
(LINQ to XML)

Jika kita mencoba mencari sebuah tag Element <ContactPerson>, maka semua data ChildElement yang ada dari TAG yang kita minta akan ikut terambil. Dari contoh ini <ContactPerson> memiliki 3 buah childElement (<nama>,<email>,<url>).

Catatan:
Perlu diperhatikan juga pengunaan pencarian TAG element ini bersifat Case Sensitive atau penulisan nama TAG harus benar-benar sama , pengunaan Huruf besar dan kecil akan bepengaruh pada proses pencarian.

MEMBUAT XML DOCUMENT DENGAN LINQ

Penjelasan sebelumnya kita membahas pembacaan sebuah document XML, sekarang kita coba membuat sebuah XML dan mengkombinasikannya dengan pengunaan LINQ untuk proses pengambilan data.

Seperti dijelaskan sebelumnya kita dapat menuliskan langsung struktur XML document kedalam sebuah variable, selain XML statis kita juga dapat membuat dynamic XML untuk nilai data yg ingin kita tampilkan.

Sebagai contoh kita gunakan script berikut untuk membuat XML document yang mengambil data JAM dari computer kita.

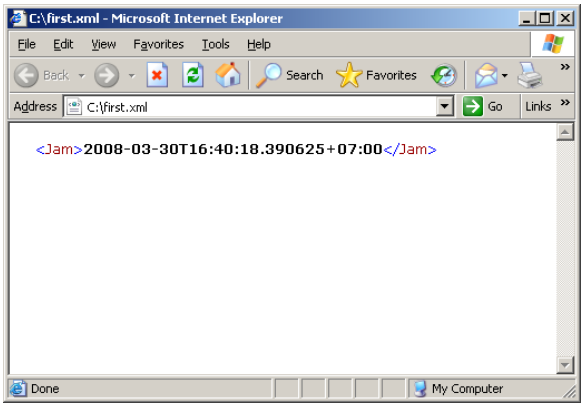
```
Dim oXML = <?xml version="1.0" encoding="utf-8"?>
           <Jam>
               <%= Now() %>
           </Jam>

My.Computer.FileSystem.WriteAllText _
          ("c:\first.xml", oXML.ToString, False)

Process.Start("c:\first.xml")
```

Baris 1,2,3,4 : Proses pembuatan XML data dengan 1 buah tag yang kita beri nama <jam>, isi value dari tag ini akan mengambil dynamic data dari waktu yang ada pada komputer kita dengan sintaks <%= Now() %>

Baris 6,7 : Memanfaatkan My keyword untuk membuat file XML baru C:\First.xml yang datanya diambil dari variable oXML yang telah dibuat sebelumnya.

Baris 9 : Membuka file c:\first.xml yang baru selesai dibuat.

Setelah selesai coba jalankan proses diatas, jika tidak ada masalah seharusnya akan ditampilkan data first.xml yang baru kita buat pada sebuah browser seperti gambar berikut:



Gambar 16
(Hasil create XML file)

Contoh diatas belum mengunakan LINQ dalam proses pembuatan data, kita baru mencoba proses sederhana membuat sebuah document XML yang mengambil data secara dynamic.

Sekarang kita coba mengunakan LINQ to OBJECT untuk membuat sebuah data XML, disini kita akan membaca daftar program yang berajalan pada computer kita.

Untuk mengambil daftar proses berjalan pada computer, kita mengunakan sintaks berikut :

```
System.Diagnostics.Process.GetProcesses
```

Dari sintaks diatas kita akan mendapatkan daftar proses yang sedang berjalan pada computer kita seperti yang terdapat pada Task manager, dari daftar tersebut kita akan mencoba membuat XML file yang berisi daftar proses berjalan pada computer kita, untuk mencoba proses ini tuliskan script berikut apda sebuah event tombol click yang ada pada form project kita.

Tambahkans sebuah tombol kedalam form yang ada pada project kita, kemudian tuliskan script berikut pada sebuah event click tombol tersebut:

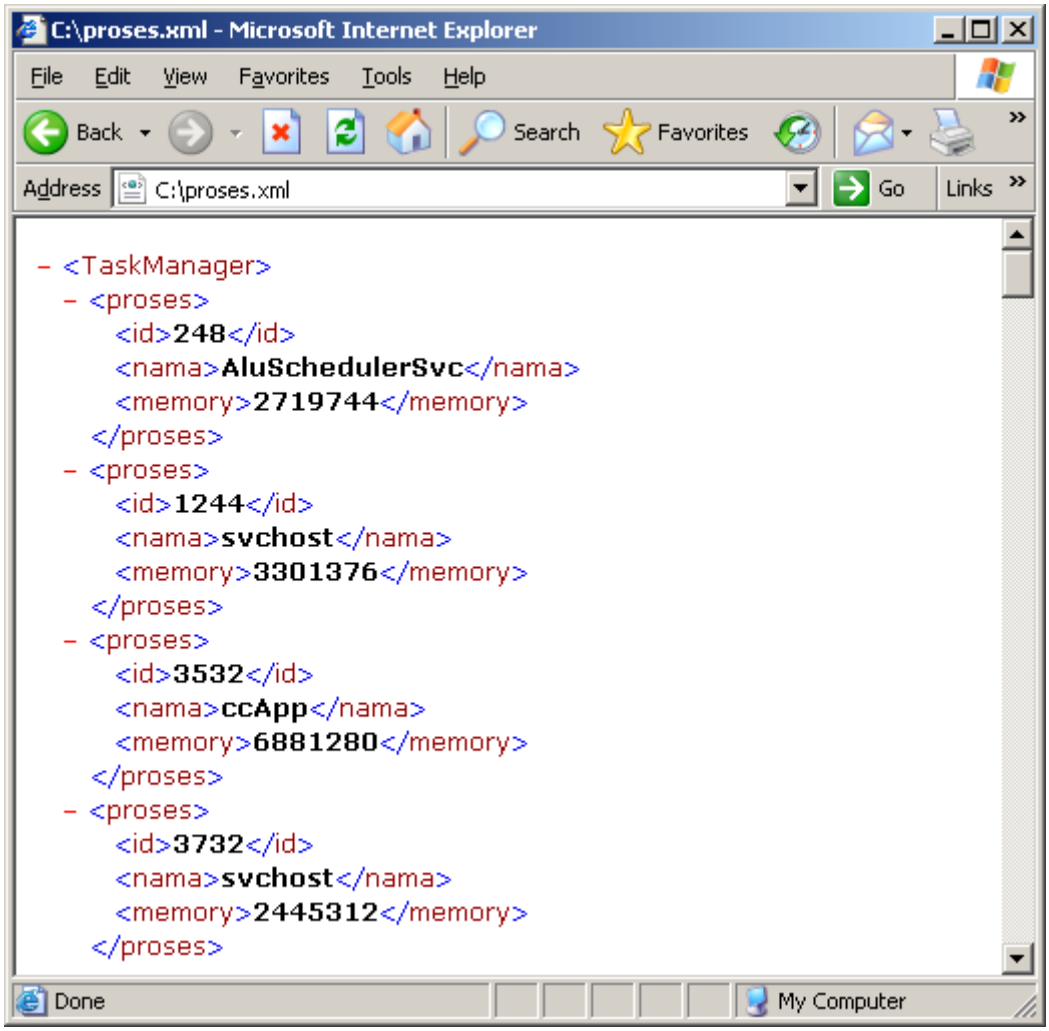
```
Dim oXML = <?xml version="1.0" encoding="utf-8"?>
           <TaskManager>
           <%= From p In System.Diagnostics.Process.GetProcesses _
               Select <proses>
                      <id><%= p.Id %></id>
                      <nama><%= p.ProcessName %></nama>
                      <memory><%= p.PagedMemorySize %></memory>
                      </proses> %>
            </TaskManager>
```

```
My.Computer.FileSystem.WriteAllText _
      ("c:\proses.xml", oXML.ToString, False)

Process.Start("c:\proses.xml")
```

Proses pembuatan dynamic data XML dengan LINQ terletak pada bagian script berikut:

```
<%= From p In System.Diagnostics.Process.GetProcesses _
    Select <proses>
             <id><%= p.Id %></id>
             <nama><%= p.ProcessName %></nama>
             <memory><%= p.PagedMemorySize %></memory>
           </proses> %>
```

Disini LINQ akan mengambil object proses berjalan pada system komputer kita, kemudian hasil query yang didapat akan dibuat sebagai struktur data ELEMENT XML <proses> yang didalamnya terdapat 3 buah childElement (id, nama dan memory). Jika kita jalankan proses tersebut nantinya akan terbentuk file c:\proses.xml yang berisi data proses yang sedang berjalan pada computer kita.
Bagian terakhir dari script diatas `Process.Start("c:\proses.xml")`adalah untuk menampilkan data hasil proses, secara default proses akan membuka browser untuk memampilkan data XML yang telah terbentuk seperti gambar berikut:

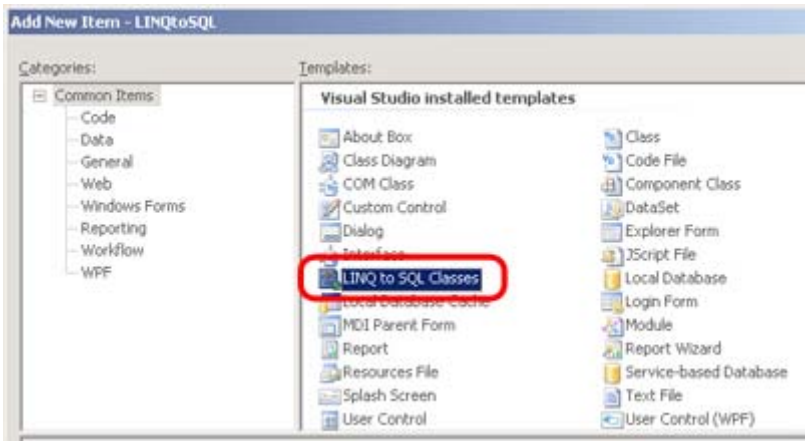


Gambar 17
(LINQ to XML)

LINQ TO SQL

Berikutnya kita akan membahas tentang pengunaan LINQ dalam melakukan query terhadap sebuah database, untuk memudahkan pengunaan LINQ to SQL pada visual Studio 2008 diperkenalkan sebuah template LINQ to SQL Classes. Dengan template ini kita akan lebih mudah membuat object model dari sebuah table database yang akan digunakan, saat membuat template ini akan dihadirkan sebuah Layar design yang diberinama Object Relational (O/R) Deginer, pada layar ini kita dapat melakukan Drag & Drop table yang ada dari server explorer dan selanjutnya Visual Studio akan membuatkan sebuah class object model untuk kita berikut visualisasi dari class tersebut. Biar tidak terlalu banyak teori berbentuk tulisan kita coba saja langsung pengunaan template ini.

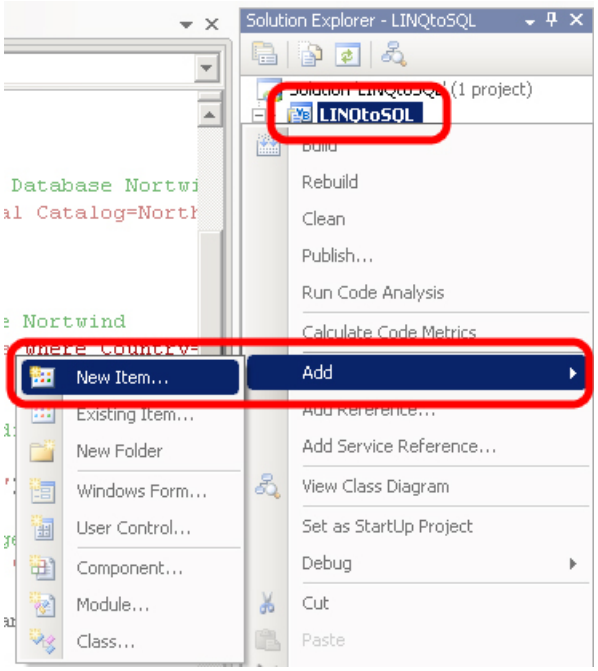
Untuk pengunaan LINQ to SQL pada visual studio 2008 telah ditambahkan sebuah template item yang bernama *LINQ to SQL Classes*.



Gambar 18
(LINQ to SQL Classes)

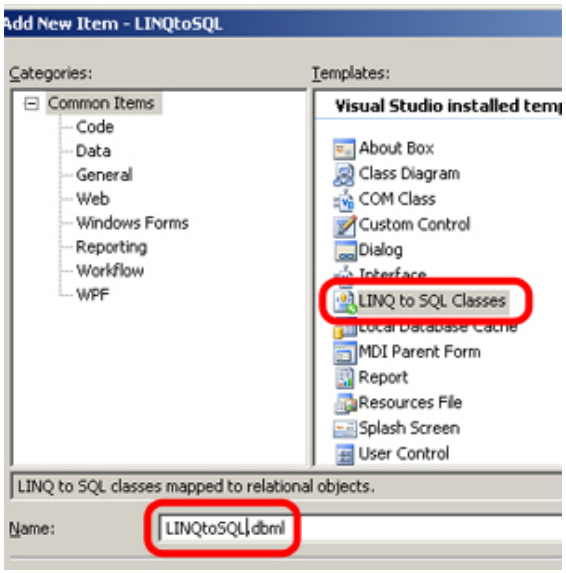
Template item ini akan membantu kita membuat sebuah class object dari struktur database yang ada, nantinya akan kita sebut sebagai *DataContext*, kemudian dari DataContext ini dapat kita gunakan untuk pembacaan dengan LINQ.

Pertama buatlah sebuah project visual basic baru, kemudian tambahkan item baru dengan click kanan pada project dalam Solution explorer dan pilih ADD → New item.



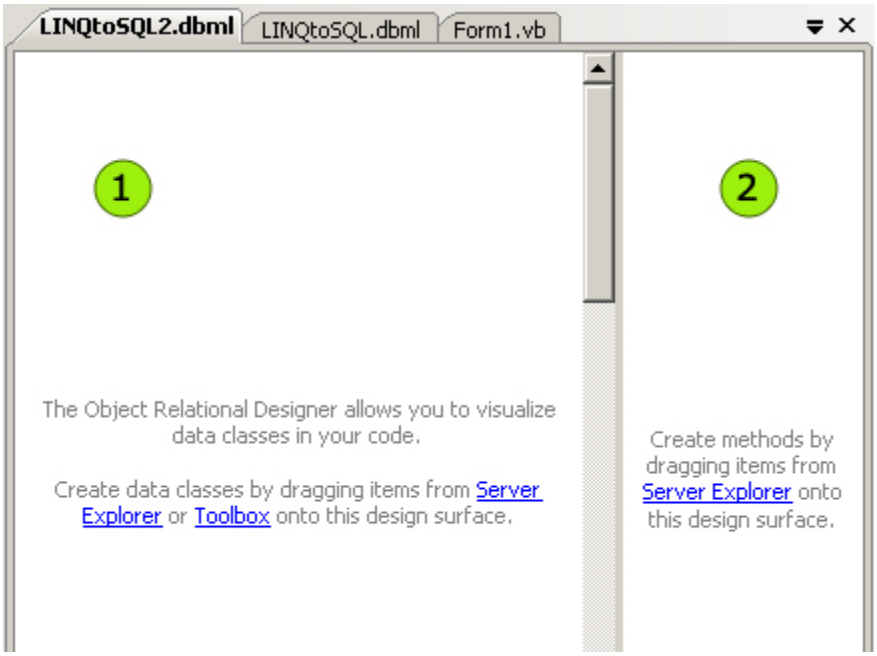
Gambar 19
(Menambah item baru kedalam project)

Dari template item yang tersedia pilih *LINQ to SQL Classes*, dan beri nama LINQtoSQL.dbml. kemudian click tombol ADD



Gambar 20
(LINQ to SQL Classes)

Setelaj proses ini kita akan mendapatkan sebuah file baru berextention .dbml pada project kita, dan juga akan muncul sebuah layar kerja *Object Relational* baru seperti gambar berikut:



Gambar 21
(Layar kerja O/R Designer)

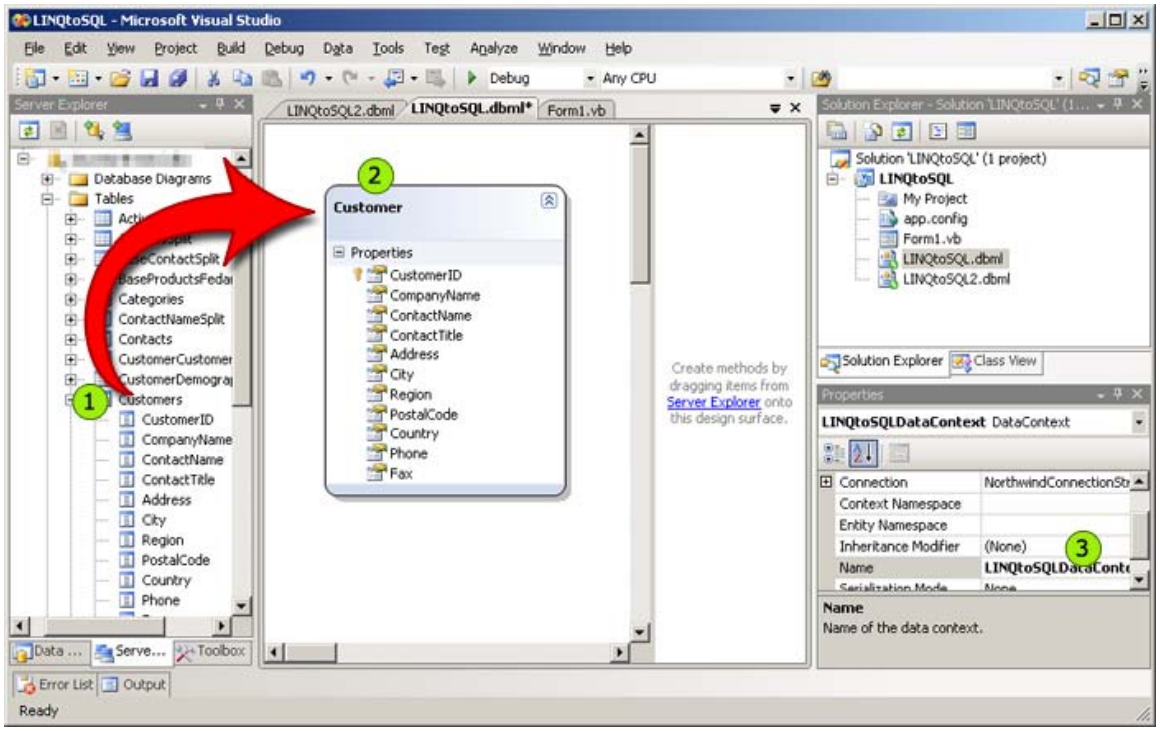
Pada bagian (1) kita dapat letakan table database kedalamnya, dan secara automatis VS2008 akan membuatkan sebuah class object baru dari table yang kita letakan didalamnya berikut dengan gambaran visualisasinya.

Pada bagian (2) dapat kita letakan method atau Store Procedure yang ada dari dalam database kita, nantinya kita dapat memanggil Store procedure tersebut sebagai sebuah method dalam proses pengunaannya.

Pada contoh ini saya akan mengunakan table *Customers* dari database *Northwind* dari SQL Server Express edititon yang telah terinstall di computer saya.
jika belum memiliki bisa download dari link berikut: www.microsoft.com/express

Untuk membuat class object baru, kita hanya perlu membuka panel Server Explorer kita yang telah terkonesi dengan database dan kemudian melakukan Drag and Drop data table yang akan dibuat Class objectnya dari Server Explorer (1) kedalam Area

Object Relational (2). Hasil proses drag and drop akan terbentuk class object baru yang bernama DataContext (3).



Gambar 22
(Membuat class object dengan O/R Designer)

Struktur class object *DataContext* yang terbentuk pada Area Object Relational (2), akan benar-benar sama dengan struktur database yang kita drag dari Server Explorer (1).

Setelah DataContext terbentuk, maka kita dapat mengunakan class tersebut pada form kerja kita. Untuk itu sekarang cobalah buka Code editor untuk Form1 yang akan kita gunakan, kemudian tambahkan deklarasikan variable baru yang mengambil struktur dari DataContext pada bagian atas script setelah baris *Public class Form1*.

```
Private dc As New LINQtoSQLDataContext
```

Catatan:
Nama dataContext dapat dilihat pada properties panel, seperti pada nomor (3) gambar diatas.

Kemudian tambahkan sebuah tombol baru kedalam form yang digunakan dan

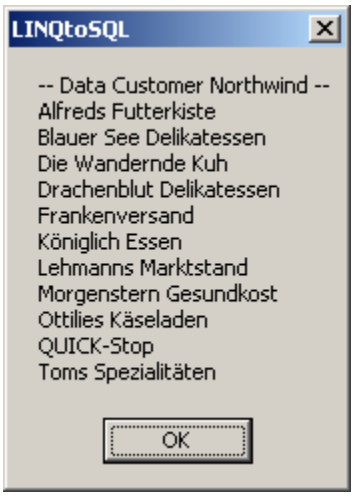
tambahkan script proses berikut kedalam event click.

```
Dim LINQ = From cust In dc.Customers _
           Where cust.Country.ToLower = "germany" _
           Order By cust.CompanyName _
           Select cust

Dim hasil = "-- Data Customer Northwind --" & vbCrLf
For Each cust In LINQ
    hasil = hasil & cust.CompanyName & vbCrLf
Next

MsgBox(hasil)
```

Baris 1,2,3,4 : Merupakan proses LINQ yang mengambil data cust dari dalam DataContext customer yang telah kita bentuk sebelumnya pada file .dbml, proses juga melakukan filter hanya pada data2 customer yang berasal dari *Germany*, dan kemudian melakukan pengaturan urutan berdasarkan *CompanyName*.

Baris (selanjutnya) : Setelah selesai melakukan proses query, selanjutnya proses akan menampilkan data2 yang didapat kedalam sebuah msgbox visual basic.

Jika tidak ada masalah maka akan dihasilkan msgbox seperti gambar berikut:



Gambar 23
(LINQ to SQL)

Dari gambaran pengunaan diatas mudah2an proses pengunaan Class Object LINQ to SQL dengan memanfaatkan template class pada file .dbml menjadi lebih jelas.

LINQ SQL dan dataReader

Sekarang mari kita bandingkan sedikit proses yang sama dengan mengunakan Native connection mengunakan connectionString dan DataReader.

Tambahkan sebuah tombol baru kedalam form yang ada, dan pada event click tombol tersebut tuliskan script proses berikut:

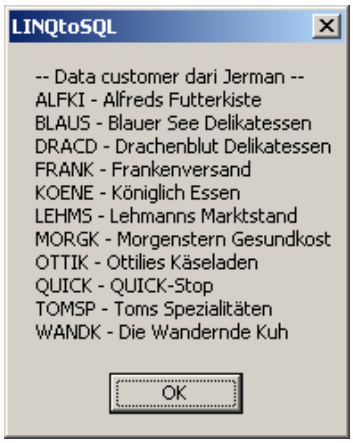
```
Dim connectionStr As String
Dim cn As SqlConnection
Dim cmd As SqlCommand
Dim dr As SqlDataReader

' Connection String untuk connection kedalam Database Nortwind
connectionStr = "Data Source=localhost;" _
                "Initial Catalog=Northwind;Integrated Security=True"
cn = New SqlConnection(connectionStr)
cn.Open()

' Query data Customer yang ada dalam database Nortwind
cmd=New SqlCommand("Select * from Customers where Country='Germany'",
cn)
dr = cmd.ExecuteReader

' Proses LINQ membaca DataReader yang telah dibuat sebelumnya
Dim LINQ = From cust In dr _
           Select ID = cust.item("CustomerID"), _
                  Name = cust.item("CompanyName")

' Menampilkan data hasil query kedalam messagebox
Dim hasil = "-- Data customer dari Jerman -- " & vbCrLf
For Each cust In LINQ
    hasil = hasil & cust.ID & " - " & cust.Name & vbCrLf
Next

MsgBox(hasil)
```

Catatan:

Ganti terlebih dahulu ConnectionStr yang digunakan diatas dengan Connection String dimana program akan digunakan

Setelah selesai cobalah jalankan program dan click tombol yang ada didalamnya, jika tidak terdapat masalah maka akan ditampilkan semua data Customers yang

berasal dari Negara Jerman seperti query yang kita lakukan.



Gambar 24
(DataReader)

Contoh diatas hanya sebagai pengenalan LINQ dalam mengakses dataReader yang telah ada, namun proses diatas terlihat tidak efisien karena kita melakukan 2 kali proses query, pertama Query untuk dataReader dan kedua query pada proses LINQ.

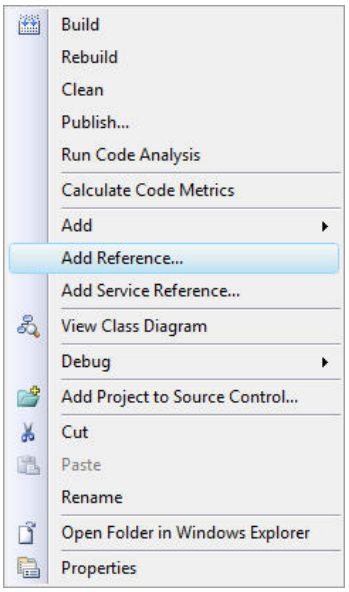
LINQ SQL dan DataContext

Dari 2 contoh sebelumnya kita telah mencoba membuat koneksi kedalam database memanfaatkan Template class LINQ to SQL yang telah disediakan VS2008 untuk membentuk DATACONTEXT, dan contoh kedua mengunakan koneksi yang biasa dilakukan mengunakan ConnectionString dan juga DataReader.

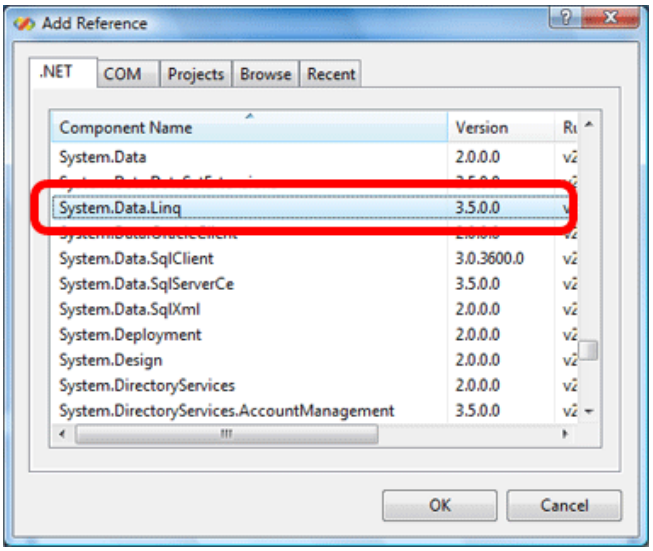
Contoh berikutnya kita akan mencoba mengkombinasikan kedua proses sebelumnya, kita akan membuat koneksi kedalam database mengunakan connectionString dan juga membuat sebuah object DataContext dengan menuliskan script untuk membaca isi table Customers seperti contoh2 sebelumnya

Agar dapat mengunakan dan membuat object DataContext dalam project, menambahkan Reference pengunaan *System.data.linq* dan juga meng-import *System.Data.Linq* namespace tersebut kedalam project kita.

Untuk menambahkan Referene project click kanan pada nama project dibagian Solution explorer → add reference, kemudian cari reference untuk System.data.linq yang ingin ditambahkan.



Gambar 25
(Add Preference)



Gambar 26
(Add Preference)

Kemudian untuk mengimport namespace tersebut cukup letakan script berikut dibagian awal project kita.

```
Imports System.Data.Linq
```

Setelah mengimport Namespace tersebut maka kita dapat membuat object DataContext dengan mengunakan script pada project kita, berikut sintaks

pembuatannya mengunakan sebuah connectionString.

```
Dim connectionStr = "Data Source=localhost;" & _
                    "Initial Catalog=Northwind;" & _
                    "Integrated Security=True"
Dim Db = New DataContext(connectionStr)
```

Setelah proses diatas kita telah memiliki sebuat object DataContext sama seperti kita membuatnya dengan mengunakan Template class SQL to LINQ pada contoh pertama, namun kali ini object yang kita buat bersifat onthefly (Virtual dan tersimpan didalam memory), kita tidak memiliki visualisasi gambar seperti contoh pertama.

Berikutnya untuk mengunakan salah satu table yang ada didalamnya kita akan memanfaatkan *getTable* method yang ada didalam object DataContext.
Untuk mengunakan getTable kita memerlukan struktur class dari table yang akan kita gunakan, disini kita akan memanfaatkan struktur class yang telah dibentuk dalam proses Template classes LINQ to SQL, sintaks lengkap pengunaan getTable adalah sebagai berikut:

```
Public Function GetTable(Of TEntity As Class)() As Table(Of TEntity)
```

Dari sintaks diatas terlihat bawa function/method getTable memerlukan 1 parameter ber-type Class yang harus disertakan dalam pengunaannya, kemudian proses getTable akan mengembalikan hasil berupa data berType Table yang akan ditampung kedalam parameter class yang kita berikan.

Dari contoh kita sebelumnya kita perlu menuliskan script berikut untuk mengambil data table Customer dalam database Northwind, kemudian memampung data table kedalam format class Customer yang telah dibentuk pada file .dbml.

```
Dim Customers As Table(Of Customer) = Db.GetTable(Of Customer)()
```

Secara lengkap penulisan script pengunaan DataContext dan juga class Customer yang telah terbentuk adalah seperti dibawah.

```
'---- Membuat object DataContext ------
Dim connectionStr = "Data Source=localhost;" & _
                    "Initial Catalog=Northwind;" & _
                    "Integrated Security=True"
Dim Db = New DataContext(connectionStr)

'---- Membaca table CUSTOMERS kedalam class Customer  ------
Dim Customers As Table(Of Customer) = Db.GetTable(Of Customer)()

'---- Melakukan query kedalam class customers table ------
Dim Query = From cust In Customers _
            Where cust.Country = "germany" _
```

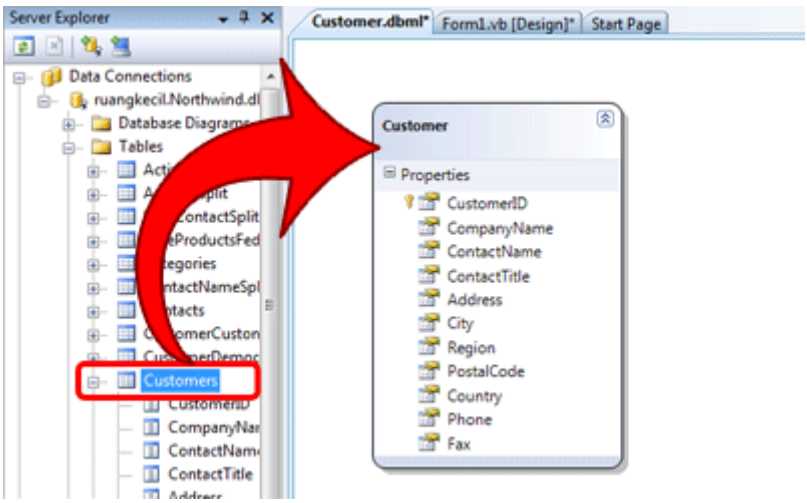
```
            Select cust

'---- Membaca data hasil query LINQ  ------
Dim hasil = "-- Data customer dari Jerman -- " & vbCrLf
For Each cust In Query
   hasil = hasil & cust.CustomerID & "-" & cust.Country & vbCrLf
Next
MsgBox(hasil)
```

BINDING data LINQ kedalam Grid

Sekarang kita akan mencoba melakukan Binding data hasil proses query LINQ kedalam sebuah object Grid. Untuk mencoba proses ini buatlah sebuah project Visual Basic baru dan beri nama BINDING.

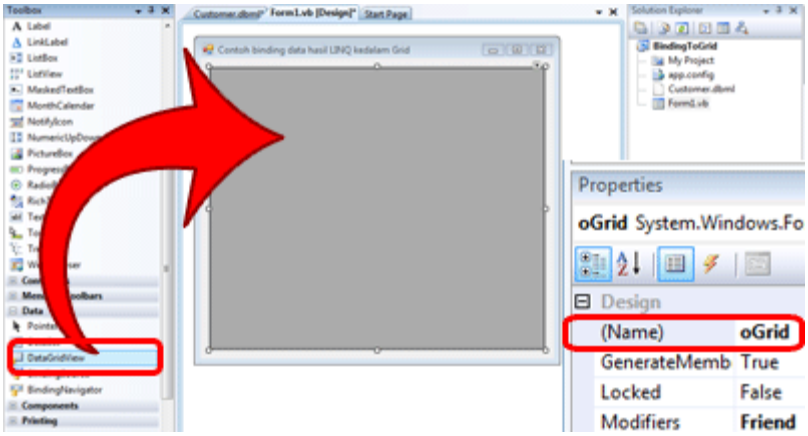
Setelah project terbentuk tambahkan sebuah template clasess LINQ to SQL kedalam project dan beri nama Customer.dbml (lihat contoh proses menambahkan class ini pada penjelasan sebelumnya).

Setelah class terbentuk dan Layar O/R kosong ditampilkan, lakukan Drag & Drop table customers dari table Northwinds kedalam layar O/R yang ada.



Gambar 27
(Import Data Struktur kedalam O/R Designer)

Setelah class customer terbentuk pada layar O/R, sekarang kita kembali pada form VB yang ada dan tambahkan sebuah object DataGridView kedalamnya dan beri nama oGrid.



Gambar 28
(Add Grid object kedalam form)

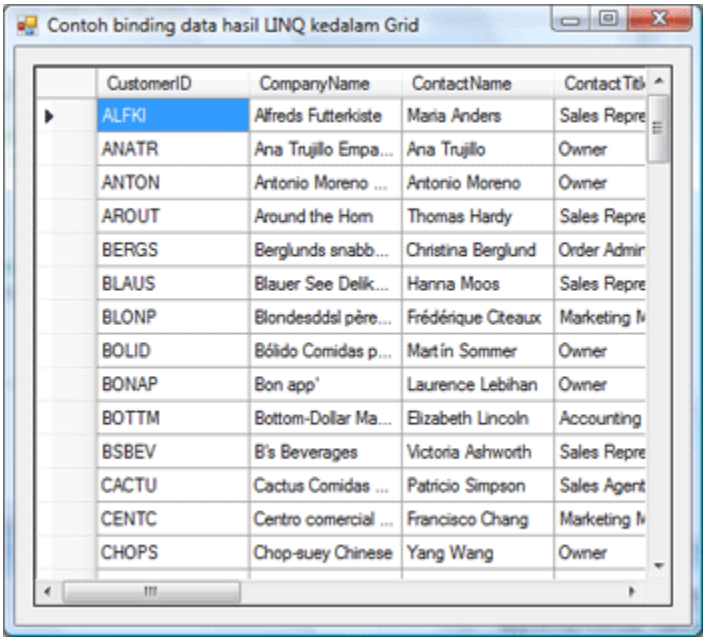
Kemudian pada double click area kosong pada Form yang ada agar kita masuk kedalam layar code editor, tambahkan script berikut pada bagian atas script setelah deklarasi class form1.

```
Public dc As New CustomerDataContext()
```

Lalu tambahkan proses berikut kedalam event Load dari form yang kita gunakan.

```
'-- Binding DataContext Customers kedalam Grid ---
oGrid.DataSource = dc.customers
```

Setelah selesai jalankan project tersebut dengan menekan tombol (F5), maka hasil pembacaan data LINQ telah ditambahkan kedalam Grid yang kita miliki seperti gambar berikut:



Gambar 29
(Hasil BINDING Data)

Dengan proses diatas kita telah melakukan BINDING DataContext kedalam datasource dari object Grid, semua data yang ada pada table customers akan ditampilkan pada Grid, namun jika kita ingin melakukan filter dan sedikit manipulasi urutan index terhadap data yang akan ditampilkan kita perlu mengunakan proses LINQ seperti berikut:

```
'-- Proses pembacaan data dengan LINQ ---
Dim Customers = From data In dc.Customers _
                Where data.Country = "UK" _
                Order By data.CompanyName _
                Select data


'-- Binding data LINQ Customers kedalam Grid ---
oGrid.DataSource = Customers
```

Contoh diatas kita lakukan Filter data untuk customers yang berasal dari “UK” dan kemudian melakukan pengurutan data berdasarkan CompanyName, pada bagian terakhir kita BINDING data hasil proses LINQ kedalam DataSource object Grid.

Untuk BINDING data ini kita dapat melakukan kepada semua object yang memiliki/mendukung DataSource type didalamnya, kita dapat parsing data hasil LINQ kedalam ListBox, ComboBox dsb.

Sekian pembahasan kita tentang berkenalan dengan LINQ pada visual basic, belum semua kemapuan dari LINQ kita bahas disini. Mudah-mudahan tutorial singkat ini memberikan cukup gambaran pengunaan LINQ.

Selamat mencoba,
Happy coding.

Referensi

- MSDN, http://msdn.microsoft.com/vbasic
- VB Team Blog, http://blogs.msdn.com/vbteam/
- Beth Messi, http://blogs.msdn.com/bethmessi/
- Perjalanan ke desa LINQ, http://geeks.netindonesia.net/files/
- LINQ pocket reference, O'Reily

TOOLS

- Visual studio 2008 Express Edition, http://www.microsoft.com/express
- Ms SQL Server Express edition, http://www.microsoft.com/express
- LINQPad, http://www.lingpad.net

Biografi Penulis

Jerry Peter Saerang. Lahir di jakarta 15 Januari 1979. Menyelesaikan studi S1 di Universitas Gunadarma, Jakarta. Dan saat ini dalam proses menyelesaikan Thesis pada program Studi S2 di Universitas Gunadarma.
Saat ini bekerja sebagai System Analyst & Software Developer di sebuah perusahaan Farmasi Nasional.

Informasi lebih lanjut tentang penulis bisa di dapat melalui:
Email: jerry.peter@gmail.com
Blog: www.ruangkecil.or.id