Linq for VB はものすごい

えムナウ（児玉宏之）
http://mnow.jp/
http://mnow.wankuma.com/
http://blogs.wankuma.com/mnow/
http://www.ailight.jp/blog/mnow/

アジェンダ

- はじめに
- Linq をおさらいしてみる
- Linq 関係で VB のたりないところ
- Linq for VB のすごいところ
- まとめ

はじめに

- 私がわんくま勉強会で Linq についてしゃべるのも、もう3回目になりました。
- 今回は VB Day とのことなので VB にからんだ Linq を見ていきたいと思います。
- 本日初めて参加される方もいらっしゃるようなので、まずはおさらいから、次に Linq に関係する VB で改善すべき点、Linq for VB のいい点を解説したいと思います。

Linq をおさらいしてみる

- Linq は .Net Framework上の「言語に統合されたクエリ」(Language-INtegrated Query)の拡張セットを指すコードネームの名称です。

- こんな感じです。
  - from から先に書く
  - 型推論が働くので入力が簡単
  - SQL文とは順序が逆なので注意

Linq をおさらいしてみる

- Linq to Object
  - **IEnumerable<T> ベースの情報ソースにクエリを適用**

```
Dim al = New Collection
al.Add(New With {.Name = "hnaka", .ZipCode = "553-0001", .Prefecture = "大阪府"})
al.Add(New With {.Name = "hkodama", .ZipCode = "168-0064", .Prefecture = "東京都"})

Dim accounts = From a In al _
               Where a.ZipCode = "168-0064" _
               Select New With {a.Name, a.ZipCode}

For Each account In accounts
    Console.WriteLine(account.Name & "(" & account.ZipCode & ")")
Next
```

Linq をおさらいしてみる

- Linq to DataSet
  - 従来のADO.NETのDataSetの情報ソースにクエリを適用

```
Using ta As New AcountDataSetTableAdapters.AccountTableAdapter
        Dim dt As New AcountDataSet.AccountDataTable
        ta.Fill(dt)

        Dim accounts = From a In dt.AsEnumerable() _
                    Where a.ZipCode = "168-0064" _
                    Select New With {a.Name, a.ZipCode}

        For Each account In accounts
           Console.WriteLine(account.Name & "(" & account.ZipCode & ")")
        Next
End Using
```

Linq をおさらいしてみる

- Linq to SQL
  - SQLサーバーのデータベースの情報ソースにクエリを適用

```
Using db As New AccountDataContext
        Dim accounts = From a In db.Account _
                       Where a.ZipCode = "168-0064" _
                       Select New With {a.Name, a.ZipCode}

        For Each account In accounts
            Console.WriteLine(account.Name & "(" & account.ZipCode & ")")
        Next
End Using
```

Linq をおさらいしてみる

- Linq to Entities
  - Entity Framework を通した概念エンティティの情報ソースにクエリを適用

```
Using db As New DemoEntities
        Dim accounts = From a In db.AccountModel設定 _
                       Where a.ZipCode = "168-0064" _
                       Select New With {a.Name, a.ZipCode}

        For Each account In accounts
            Console.WriteLine(account.Name & "(" & account.ZipCode & ")")
        Next
End Using
```

Linq をおさらいしてみる

- Linq to XML
  - XMLのXelementの情報ソースにクエリを適用

```
Dim xml = <Accounts>
              <Account Name="nNaka" ZipCode="553-0001" Prefecture="大阪府"/>
              <Account Name="hkodama" ZipCode="168-0064" Prefecture="東京都"/>
          </Accounts>
    Dim accounts = From a In xml.Elements _
                   Where a.@ZipCode = "168-0064" _
                   Select New With {a.@Name, a.@ZipCode}

    For Each account In accounts
        Console.WriteLine(account.Name & "(" & account.ZipCode & ")")
    Next
```

Linq をおさらいしてみる

- 対象となるデータは以下のようなものです。

名称	対象
Linq to Objects	IEnumerable<T>
Linq to DataSet	ADO.NETのDataSet
Linq to SQL	SQLサーバーのデータベース
Linq to Entities	Entity Framework を通した概念エンティティ
Linq to XML	XMLのXElement

Linq をおさらいしてみる

- IQueryable(Of (T)) インターフェイスを実装すれば独自の Linq to XXX も作れます。
  - LINQ to WebQueries
  - LINQ to Amazon
  - LINQ to RDF Files
  - LINQ to MySQL
  - LINQ to NHibernate
  - LINQ to LDAP
  - LINQ to Flickr
  - LINQ to Google Desktop
  - LINQ to SharePoint
  - LINQ to Streams (SLinq, Streaming LINQ)
  - LINQ to Expressions

http://oakleafblog.blogspot.com/2007/03/third-party-linq-providers.html

Linq をおさらいしてみる

- Linq to Object
  - Linq to Object は、IEnumerable<T> によるパイプライン
  - Linq to Dataset や Linq to XML も同じ
  - 便利に使うことができる反面、メソッドの内部で使用する foreach の回数に注意を払わないと、効率が悪い場合もある
  - データ量や参照回数が多くなきゃ大丈夫
  - 同じ Linq の foreach を何回もやるんであれば ToArray

Linq をおさらいしてみる

- Linq to SQL
  - 母体になる DataContext は Dataset より進化しています。
  - Row は INotifyPropertyChanging や INotifyPropertyChanged を実装したObjectです。
  - 当然 Insert Update Delete ストアド も使えます。
  - あらかじめ必要な partial method が仕込まれています。
  - 同時実行制御で競合の解決がサポートされています。

Linq をおさらいしてみる

- Linq to Entities
  - 母体になる ObjectContext は Dataset より進化しています。
  - EntityObjectはINotifyPropertyChanging や INotifyPropertyChanged を実装してます。
  - 当然 Insert Update Delete ストアド も使えます。
  - あらかじめ必要な partial method が仕込まれています。
  - SQLより高次元な概念クエリを発行できます。

Linq 関係で VB のたりないところ

- イテレータ や Yield がない
  - VB9 の段階では、IEnumerable を返す、イテレータに使用する Yield がありません。
  - イテレータで遅延解析する機構を使えませんので、Extension Method とかで、コレクションを返す方法が現在は適当です。

Linq 関係で VB のたりないところ

```
Module Module1
    Sub Main()
        Dim al = New String() {"hnaka", "hkodama", "ttakahagi"}

        Dim accounts = al.MyExtension

        For Each account In accounts
            Console.WriteLine(account)
        Next

    End Sub
```

```
<System.Runtime.CompilerServices.Extension()> _
    Public Function MyExtension
            (ByVal values As IEnumerable(Of String))
             As IEnumerable(Of String)

        Dim results As New List(Of String)
        For Each value In values
            If value.StartsWith("h") Then
                results.Add(value)
            End If
        Next

        Return results
    End Function
End Module
```

Linq 関係で VB のたりないところ

- 安心してください次のバージョンでは入るかも知れません。

http://www.panopticoncentral.net/archive/2008/08/08/24155.aspx

```
Function FromTo(ByVal low As Integer, ByVal high As Integer) As
                IEnumerable(Of Integer)
    Return Iterator
        If low <= high Then
            Return low
            Return Each FromTo(low + 1, high)
        End If
    End Iterator
End Function
```

Linq 関係で VB のたりないところ

- コレクション初期化子 がない
  - 今はコレクションを定義して、ひとつひとつ Add していくしか方法はなさそうです。

```
Dim al = New Collection
al.Add(New With {.Name = "hnaka", .ZipCode = "553-0001", .Prefecture = "大阪府"})
al.Add(New With {.Name = "hkodama", .ZipCode = "168-0064", .Prefecture = "東京都"})
```

Linq 関係で VB のたりないところ

- 安心してください、複雑なこともできるようにしようと、次バージョンに回したようです。

http://www.infoq.com/jp/news/2008/05/VB-Collections

```
Dim x = {1, 2, 3, 4}
Dim y = {{"a", 1}, {"b", 2}, {"c", 3}, {"d", 4}}
Dim z = {{1, 2}, {3, 4}, {5, 6}, {7, 8}}

Dim x = {1, 2, 3, 4}                    'List(Of Int32)
Dim x() = {1, 2, 3, 4}                  'array of Int32
Dim d = {1:“Hello”, 2:“World” }         ‘ディクショナリー
```

Linq 関係で VB のたりないところ

- やっぱり __ がうざったい
  - とにかく毎行の最後には _ を入れないといけない。
  - Linq は特に複数行に書くことが多いので非常に気になります。
  - できているクエリを修正しようとして、途中で行を追加して、_ を忘れたりすると、次の行のエラーを勝手に修復しようとして () とかを余分に作ったりします。

Linq 関係で VB のたりないところ

- 安心してください次のバージョンでは、_ は不要になるかも知れません。

http://www.panopticoncentral.net/archive/2008/02/27/22910.aspx

```
a = b +
   c

Console.WriteLine(
"{0} {1}",
FirstName,
LastName )

Dim ys = From x In xs
Where x > 5
Select ten = x * 10,
 twenty = x * 20,
 thirty = x * 30
```

Linq for VB のすごいところ

- C# は、メソッドベースでしかサポートされていない機能もありますが、Visual Basic ではクエリ式の構文が用意されています。
- いっぱいあるので組み合わせて使うととても便利です
- 集計用の構文も作られていてとても便利です。

Linq for VB のすごいところ

- IEnumerable で定義されているメソッドを言語でほとんど組み込まれている。
  - C# では「言語に統合されたクエリ」といっても、統合されたのは有名どころだけ。
  - VB ではかなり頑張って実装されました。
  - 欲を言えば、Union とかSQL文として定義されているものはすべて入っているとよかった。

Linq for VB のすごいところ

IEnumerableメソッド	C# のクエリ式の構文	Visual Basic のクエリ式の構文
Cast	from type i in numbers	From … As …
GroupBy	group … by	Group … By … Into …
	group … by … into …	
GroupJoin	join … in … on … equals … into …	Group Join … In … On …
Join	join … in … on … equals …	From x In , y In Where x.a = y.a
		Join … [As …]In … On …
	let … = …	Let … = …
OrderBy	orderby …	Order By …
OrderByDescending	orderby … descending	Order By … Descending
Select	select	Select
SelectMany	複数の from 句。	複数の From 句。
ThenBy	orderby …, …	Order By …, …
ThenByDescending	orderby …, … descending	Order By …, … Descending
Where	where	Where

Linq for VB のすごいところ

IEnumerableメソッド	C# のクエリ式の構文	Visual Basic のクエリ式の構文
All	該当なし	Aggregate … In … Into All(…)
Any	該当なし	Aggregate … In … Into Any()
Average	該当なし	Aggregate … In … Into Average()
Count	該当なし	Aggregate … In … Into Count()
Distinct	該当なし	Distinct
LongCount	該当なし	Aggregate … In … Into LongCount()
Max	該当なし	Aggregate … In … Into Max()
Min	該当なし	Aggregate … In … Into Min()
Skip	該当なし	Skip
SkipWhile	該当なし	Skip While
Sum	該当なし	Aggregate … In … Into Sum()
Take	該当なし	Take
TakeWhile	該当なし	Take While

Linq for VB のすごいところ

Enumerableメソッド	機能
Concat	2つのIEnumerableの連結
Contains	要素がIEnumerableに格納されているかどうかを判断
Except	1つめのIEnumerableから2つめのIEnumerableの要素を削除
Intersect	2つのIEnumerableの積集合
OfType<Type>	Typeで指定された型の物だけ抜き出す
Range	指定範囲の整数のIEnumerableを作成
Repeat	一つの要素を繰り返し作成
Reverse	IEnumerableの要素の順番を反転
SequenceEqual	2つのIEnumerableが等しいか比較
Union	2つのIEnumerableの和集合

Linq for VB のすごいところ

Enumerableメソッド	機能
DefaultIfEmpty	IEnumerableが空でなければそのまま、空ならデフォルト値
ElementAt	インデックス位置にある要素
ElementAtOrDefault	インデックス位置にある要素、空ならデフォルト値
Empty	空のIEnumerable
First	先頭の要素
FirstOrDefault	先頭の要素、空ならデフォルト値
Last	最後の要素
LastOrDefault	最後の要素、空ならデフォルト値
Single	要素が一つか確認し取り出す
SingleOrDefault	要素が一つか確認し取り出す、空ならデフォルト値
ToArray	Arrayに変換する
ToDictionary	Dictionaryに変換する
ToList	Listに変換する
ToLookup	Lookupに変換する

Linq for VB のすごいところ

- 集計するのに便利な Aggregate
  - クエリに集計関数を含めることができ、複数の集計関数を一度にかける。
  - 集計関数の結果は、クエリ結果の中にクエリ結果型のフィールドとして格納されます。
  - C# は結果を一つしかもらえないので、複数は書けない。
  - Aggregate の中には Where や Order By などのクエリも書けます。

Linq for VB のすごいところ

- こんなのが定義できてしまう

```
Dim point() As Double = {10.2, 12.5, 8.9, 13.8, 15.9}

Dim countQuery = Aggregate p In point _
                 Order By p _
                 Skip 1 Take point.Length - 2 _
                 Into CountValue = Count(), SumValue = Sum(), _
                    AverageValue = Average(), MaxValue = Max(), MinValue = Min()

Console.WriteLine("Count=" + countQuery.CountValue.ToString())
Console.WriteLine("Sum=" + countQuery.SumValue.ToString())
Console.WriteLine("Average=" + countQuery.AverageValue.ToString())
Console.WriteLine("Max=" + countQuery.MaxValue.ToString())
Console.WriteLine("Mix=" + countQuery.MinValue.ToString())
```

Linq for VB のすごいところ

- さらにうれしいのは Distinct

```
Dim text() As String = {"A penny saved is a penny earned.", _
        "The early bird catches the worm.", _
        "The pen is mightier than the sword.", _
        "My name is M-now."}

    Dim earlyBirdQuery = From sentence In text _
        Let words = sentence.Split(" "c, "."c) _
        From word In words _
        Where Not String.IsNullOrEmpty(word) _
        Let w = word.ToLower() _
        Order By w _
        Select w Distinct

    For Each v In earlyBirdQuery
        Console.WriteLine(v)
    Next
```

Linq for VB のすごいところ

- XML だって代入文で __ なしに書けてしまう。

```
Dim xml = <Accounts>
   <Account Name="nNaka" ZipCode="553-0001" Prefecture="大阪府"/>
   <Account Name="hkodama" ZipCode="168-0064" Prefecture="東京都"/>
                </Accounts>
Dim accounts = From a In xml.Elements _
                    Where a.@ZipCode = "168-0064" _
                    Select New With {a.@Name, a.@ZipCode}

For Each account In accounts
          Console.WriteLine(account.Name & "(" & account.ZipCode & ")")
Next
```

まとめ

- C# で出来て VB で出来ないことは、どんどん減っていきます。
- いっぽう C# で出来ないことは あまり予約後を増やしたくないせいか減りません。（動的言語サポートはやりそうですが）

http://blogs.msdn.com/charlie/archive/2008/01/25/future-focus.aspx

- 業務で利用する言語としては VB の方が便利なのかもしれないと思う今日この頃です。