LINQ

Language-Integrated Query (LINQ)

- LINQ es un conjunto de características que agrega **capacidades de consulta** eficaces a VB.net y C#.
- Facilita la consulta **a cualquier origen de datos** *(bases de datos, documentos XML, colecciones, conjuntos de datos y entidades de ADO.net…)*
- Emplea una **sintaxis unificada** independientemente del origen de datos a consultar.

Consultas Linq

- Expresiones que recuperan datos de un origen de datos *(bases de datos, documentos XML, colecciones, conjuntos de datos y entidades de ADO.net…)*
- Emplean los mismos patrones de codificación para consultar y transformar cualquier origen de datos para el que esté disponible un proveedor *(en lugar de utilizar SQL para consultar bases de datos, o Xquery para consultar XML)*

Consultas Linq

- Fases de una operación de consulta
  - 1. Obtener el origen de los datos.
  - 2. Crear la consulta.
  - 3. Ejecutar la consulta.

```
' Data source.
Dim numbers() As Integer = {0, 1, 2, 3, 4, 5, 6}

' Query creation.
Dim evensQuery = From num In numbers
                 Where num Mod 2 = 0
                 Select num

' Query execution.
For Each number In evensQuery
    Console.Write(number & " ")
Next
```

Salida:

```
0 2 4 6
```

Consultas Linq

Fase1. Obtener el origen de los datos

- Ej. el origen de datos es un array

```
' Data source.
Dim numbers() As Integer = {0, 1, 2, 3, 4, 5, 6}
```

- Ej. el origen de datos es un documento XML

```
' Create a data source from an XML document.
Dim contacts = XElement.Load("c:\myContactList.xml")
```

- Ej. el origen de datos es una tabla de una base de datos relacional

```
' Create a data source from a SQL table.
Dim db As New DataContext("C:\Northwind\Northwnd.mdf")
Dim customers As Table(Of Customer) = db.GetTable(Of Customer)
```

Consultas Linq

Fase2. Crear / definir la consulta *(ej. sobre array)*

```
' Query creation.
Dim evensQuery = From num In numbers
                 Where num Mod 2 = 0
                 Select num
```

- **From** especifica el origen *(number array)* y una variable de iteración *(num)*
- **Where** especifica el filtrado
- **Select** especifica forma y contenido de los elementos devueltos

Ojo!! La variable de consulta no realiza acción alguna ni devuelve ningún dato. Únicamente almacena la definición de la consulta. *En el ejemplo de la diapositiva 4 es el bucle* ***For Each*** *el que ejecuta la consulta.*

Consultas Linq

Fase2. Crear /definir la consulta *(ej. sobre xml)*

```
Dim homePhone = From phone In contacts...<phone>
                Select phone.Value
```

- **From** especifica el origen *(elementos <phone> descendientes de contacts)* y una variable de iteración *(phone)*
- **Where** especifica el filtrado
- **Select** especifica forma y contenido de los elementos devueltos

Ojo!! La variable de consulta no realiza acción alguna ni devuelve ningún dato. Únicamente almacena la definición de la consulta. En el ejemplo de la diapositiva 4 es el bucle ***For Each*** el que ejecuta la consulta.

Consultas Linq

Fase2. Crear /definir la consulta *(ej. origen de datos es una tabla)*

- **From** especifica el origen *(tabla customers)* y una variable de iteración *(cust)*
- **Where** especifica el filtrado
- **Select** especifica forma y contenido de los elementos devueltos

```
Dim queryResults = From cust In customers
                   Where cust.Country = "Canada"
                   Select cust.CompanyName, cust.Country
```

Ojo!! La variable de consulta no realiza acción alguna ni devuelve ningún dato. Únicamente almacena la definición de la consulta. En el ejemplo de la diapositiva 4 es el bucle ***For Each*** el que ejecuta la consulta.

Consultas Linq

Fase3. Ejecutar la consulta *(For Each)*

- las definiciones anteriores han creado la consulta, pero no se ha ejecutado

```
' Data source.
Dim numbers() As Integer = {0, 1, 2, 3, 4, 5, 6}

' Query creation.
Dim evensQuery = From num In numbers
                 Where num Mod 2 = 0
                 Select num

' Query execution.
For Each number In evensQuery
    Console.Write(number & " ")
Next
```

Consultas Linq

Fase3. Ejecutar la consulta ***(For Each)***

- La consulta se define una vez, pero se puede ejecutar tantas veces como sea preciso

---------->

Consultas Linq

```
Dim numberArray() = {0, 1, 2, 3, 4, 5, 6}

Dim evensQuery2 = From num In numberArray
                  Where num Mod 2 = 0
                  Select num

Console.WriteLine("Evens in original array:")
For Each number In evensQuery2
    Console.Write("  " & number)
Next
Console.WriteLine()

' Change a few array elements.
numberArray(1) = 10
numberArray(4) = 22
numberArray(6) = 8

' Run the same query again.
Console.WriteLine(vbCrLf & "Evens in changed array:")
For Each number In evensQuery2
    Console.Write("  " & number)
Next
Console.WriteLine()
```

LINQ TO XML

Sintaxis literal XML en Visual Basic

- ***LINQ to XML***: API de programación XML en memoria diseñada específicamente para aprovechar Language-Integrated Query (LINQ).
- Los ***literales XML*** nos permiten incluir XML directamente en nuestro código facilitando la creación de fragmentos, elementos y documentos XML
- Permite representar ***objetos XML***

Sintaxis literal XML en Visual Basic

Objeto ***XElement*** definido mediante un literal XML:

```
Dim contact1 As XElement =
    <contact>
      <name>Patrick Hines</name>
      <phone type="home">206-555-0144</phone>
      <phone type="work">425-555-0145</phone>
    </contact>
```

Sintaxis literal XML en Visual Basic

Objeto ***XDocument*** definido mediante un literal XML:

```
Dim contactDoc As XDocument =
    <?xml version="1.0"?>
    <contact>
      <name>Patrick Hines</name>
      <phone type="home">206-555-0144</phone>
      <phone type="work">425-555-0145</phone>
    </contact>
```

Expresiones incrustadas

Sintaxis: ***<%= expression %>***

```
Dim isbnNumber As String = "12345"
Dim modifiedDate As String = "3/5/2006"
Dim book As XElement =
    <book category="fiction" isbn=<%= isbnNumber %>>
        <modifiedDate><%= modifiedDate %></modifiedDate>
    </book>
```

cuando se ejecuta este código, el valor de book es:

```
<book category="fiction" isbn="12345">
  <modifiedDate>3/5/2006</modifiedDate>
</book>
```

Expresiones incrustadas

- expresión incrustada en un nombre de elemento:

```
Dim elementName As String = "contact"
Dim contact1 As XElement = <<%= elementName %>/>
```

- expresión incrustada en el contenido de un elemento:

```
Dim contactName As String = "Patrick Hines"
Dim contact2 As XElement =
  <contact><%= contactName %></contact>
```

Expresiones incrustadas

- expresión incrustada en un nombre de atributo:

```
Dim phoneType As String = "home"
Dim contact3 As XElement =
  <contact <%= phoneType %>="206-555-0144"/>
```

- expresión incrustada en un valor de atributo:

```
Dim phoneNumber As String = "206-555-0144"
Dim contact4 As XElement =
  <contact home=<%= phoneNumber %>/>
```

Expresiones incrustadas

- expresión incrustada en un atributo de elemento

```
Dim phoneAttribute As XAttribute =
  New XAttribute(XName.Get(phoneType), phoneNumber)
Dim contact5 As XElement =
  <contact <%= phoneAttribute %>/>
```

- expresión incrustada en un elemento raíz de documento

```
Dim document As XDocument =
  <?xml version="1.0"?><%= contact1 %>
```

Acceder a XML- propiedades de eje

Propiedades del eje XML: *permiten acceder a nodos secundarios, nodos descendientes y atributos de un literal XML.*

Descripción de propiedad	Ejemplo	Descripción
eje hijo	`contact.<phone>`	Obtiene todos los elementos `phone` que son elementos secundarios del elemento `contact`
eje de atributo	`phone.@type`	Obtiene todos los atributos `type` del elemento phone.
eje descendiente	`contacts...<name>`	Obtiene todos los elementos `name` del elemento `contacts`, independientemente de cuán profundo en la jerarquía ocurran.

Acceder a XML- propiedades de eje

Descripción de propiedad	Ejemplo	Descripción
indexador de extensión	`contacts...<name>(0)`	Obtiene el primer elemento `name` de la secuencia.
valor	`contacts...<name>.Value`	Obtiene la representación de cadena del primer objeto en la secuencia, o `Nothing` si la secuencia está vacía.

Métodos de XDocument

*XDocument.**Save**(string)*
serializa XDocument en un archivo, sobreescribiendo un archivo existente, si existe

```
Dim doc As XDocument = _
    <?xml version="1.0" encoding="utf-8"?>
        <Root><Child>content</Child></Root>

doc.Save("Root.xml")
```

*XDocument.**Load**(string)*
crea un nuevo XDocument a partir de un archivo

```
Dim books =
    XDocument.Load(My.Application.Info.DirectoryPath &
                   "\..\..\Data\books.xml")
Console.WriteLine(books)
```

Propiedad Root de XDocument

*XDocument.**Root***
obtiene el elemento raíz del árbol XML del documento

```
Dim doc As XDocument = _
    <?xml version="1.0" encoding="utf-8" standalone="yes"?>
    <!--This is a comment.-->
    <Pubs>
        <Book>
            <Title>Artifacts of Roman Civilization</Title>
            <Author>Moreno, Jordao</Author>
        </Book>
        <Book>
            <Title>Midieval Tools and Implements</Title>
            <Author>Gazit, Inbar</Author>
        </Book>
        <!--This is another comment.-->
    </Pubs>

Console.WriteLine(doc.Root.Name.ToString())
```

```
Pubs
```

Propiedad Object.value

*Object.**value***
proporciona acceso al valor del **primer** elemento de una colección de objetos XElement

```
Dim contact As XElement =
    <contact>
        <name>Patrick Hines</name>
        <phone type="home">206-555-0144</phone>
        <phone type="work">425-555-0145</phone>
    </contact>

Console.WriteLine("Phone number: " & contact.<phone>.Value)
```

```
Phone number: 206-555-0144
```

Propiedad Object.value

```
Dim contact As XElement =
    <contact>
      <name>Patrick Hines</name>
      <phone type="home">206-555-0144</phone>
      <phone type="work">425-555-0145</phone>
    </contact>


Dim types = contact.<phone>.Attributes("type")

For Each attr In types
  Console.WriteLine(attr.Value)
Next
```

```
home
work
```