

您的潜力，我们的动力

Microsoft
微软(中国)有限公司

Visual Basic 9.0前瞻

语言集成的查询支持和动态编程

讲师：施凡

摘要

- Visual Basic 9.0 与LINQ简介
- 为LINQ而新增的语法结构
- 为动态编程新增的语法特性
- 总结与前景分析
- 技术资源

Visual Basic 9.0 为未来而创新

- Visual Basic .NET 2002 – 7.0
- Visual Basic .NET 2003 – 7.1
- Visual Basic 2005 – 8.0
- Visual Basic “Orcas” – 9.0
- Visual Basic “Hawaii” – ??

LINQ

- Language **I**Ntegrated **Q**uery
- 语言集成的查询
- 两个重要组成部分：DLinq和XLinq

DLINQ

- 数据编程目前遇到的问题：
 - 书写太多代码
 - 弱类型
 - 离线数据缺乏有效的查询手段
- DLinQ带给你：
 - 强类型的远程/离线查询运算符
 - 数据对象自动持久化
- Visual Basic 带给你：
 - 完全与Visual Basic集成的类似SQL查询语法

XLINQ

- XML编程模型中存在的问题：
 - 不直观
 - 相对容易读取，但不易生成XML
 - 需要用不同的工具（XSLT, XPath, XQuery）查询和变换XML
- XLinQ带给你：
 - 与XML树状结构一样的对象模型
 - 用统一而简单的方式访问和创建XML
- Visual Basic 带给你：
 - 深度支持XML，全行业最紧密的语言与XML集成

LINQ示意图

您的潜力，我们的动力

Microsoft
微软(中国)有限公司

C#

VB

其他...

查询操作符模式支持

标准查询操作符

DLinq
(ADO.NET)

XLinq
(System.Xml)



对象

SQL

WinFS

DB2

Oracle

```
<book>
  <title/>
  <author/>
  <year/>
  <price/>
</book>
```

XML

为LINQ而新增的语法结构

- 局部变量类型推测
- 对象初始化和匿名类型
- 嵌套函数和Lambda函数
- 扩展方法
- 查询包含
- XML字面量
- XML后期绑定

局部变量类型推测

- 编译器根据初始化表达式推测变量的类型

```
Dim i As Integer = 1
Dim s As String = "abc"
Dim a As Integer() = {1, 2, 3}

' Function GetList() As List(Of Integer)
Dim l As List(Of Integer) = GetList()

' Function GetList() As Dictionary(Of String, String)
Dim d As Dictionary(Of String, String) = GetDict()
```

```
Dim i = 1
Dim s = "abc"
Dim a = {1, 2, 3}
Dim l = GetList()
Dim d = GetDict()
```

局部变量类型推测

- 局部变量类型推测为强类型特性
- 变量类型一旦确定，就不能更改

```
Dim i = 100 ' 推测为Integer  
i = Date.Now ' 编译错误：无法将Date转换为Integer
```

- 推测循环变量

```
For Each Dim item In MyList  
    ' item为MyList的元素类型  
    ' 有全面的IDE智能感知功能  
Next
```

类型初始化器

- 以前，除非有合适的构造函数，否则只能分开定义对象和初始化

```
Dim emp As New Employee  
With emp  
    .FirstName = "Harry"  
    .LastName = "Potter"  
    .Age = 15  
End With
```

- 新增对象初始化语法

```
Dim emp = New Employee { FirstName := "Harry", _  
    LastName := "Potter", Age := 15 }  
Dim emp = New Employee("Harry", "Potter") { Age := 15 }
```

匿名类型

- 需要临时类型时，无须定义，直接用对象初始化的语法生成

```
Dim name = New { FirstName := "Harry", LastName := "Potter" }  
' name的类型为 { FirstName As String, LastName As String }
```

- 需配合局部变量初始化使用

您的潜力，我们的动力

Microsoft
微软(中国)有限公司

DEMO

- 局部变量类型推测
- 类型初始化器

嵌套函数

- Visual Basic 9.0最重要的新增特性
- 可在方法的内部嵌套定义函数或子程序

```
Private Sub MyMethod()  
    Function NestedMethod(x As Integer) As Boolean  
        Return x > 100  
    End Function  
  
    MsgBox("Greater than 100? " & NestedMethod(101))  
  
End Sub
```

嵌套函数的Closure特性

- 嵌套定义的函数可以访问其上下文的变量

```
Private Sub MyMethod()  
    Dim i = 100  
  
    Function NestedMethod(x As Integer) As Boolean  
        Return x > i  
    End Function  
  
    MsgBox("Greater than " & i & "? " & NestedMethod(101))  
  
End Sub
```

- 甚至可以修改这些变量的值

嵌套函数的意义所在

- 延迟计算

```
Delegate Function Func(Of T, R)(arg As T)As R
Dim myArr As Int32() = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

Public Function GreaterThan(val As Int32)As Func(Of Int32, Boolean)
    Function Greater(param As Int32)As Boolean
        Return param > val
    End Function
    Return AddressOf Greater
End Function

Public Function TrueForAll(arr As Int32(), f As Func(Of Int32, Boolean))As Boolean
    For Each i As Int32 In arr
        TrueForAll = TrueForAll AndAlso f(i)
    Next
End Function

Public Sub Test()
    Console.WriteLine(TrueForAll(myArr, GreaterThan(5)))
End Sub
```

Lambda函数

- 若嵌套函数为一个简单表达式
- 若仅需要取得嵌套函数的委托
- 使用Lambda定义匿名嵌套函数！

```
Public Function GreaterThan(val As Int32)As Func(Of Int32, Boolean)  
    Function Greater(param As Int32)As Boolean  
        Return param > val  
    End Function  
    Return AddressOf Greater  
End Function
```



```
Public Function GreaterThan(val As Int32)As Func(Of Int32, Boolean)  
    Return (param) param > val  
End Function
```

Linq定义的泛型Func系列委托

- 用于即时创建所需类型的函数委托

```
Delegate Function Func(Of T0, R) _  
    (a0 As T0)As R
```

```
Delegate Function Func(Of T0, T1, R) _  
    (a0 As T0, a1 As T1)As R
```

```
Delegate Function Func(Of T0, T1, T2, R) _  
    (a0 As T0, a1 As T1, a2 As T2)As R
```

```
Delegate Function Func(Of T0, T1, T2, T3, R) _  
    (a0 As T0, a1 As T1, a2 As T2, a3 As T3)As R
```

扩展方法

- 一种将第一个参数转化为操作对象的机制
- 可为任何类型（包括接口）扩展方法

```
<Extension> _  
Public Module Extension  
    <Extension> _  
    Public Function IsNumeric(str As String) As Boolean  
        Return Strings.IsNumeric(str)  
    End Function  
  
End Module
```

```
Dim mystr = "1234"  
Dim isnumber = mystr.IsNumeric()  
' 采用 参数1. 扩展方法名（其他参数） 的语法调用
```

扩展方法

- 扩展方法的限制
 - 原类型的方法优先
 - 不能是虚方法
 - 使用前必须导入
- 扩展方法的意义
 - 让顺序操作的书写顺序与运行顺序一致化
 - 更高抽象性

序列语法的次序

```
<Extension> Function Where(Of T)(emu As IEnumerable(Of T), _  
    filter As Func(Of T, Boolean)) As IEnumerable(Of T)
```

```
<Extension> Function OrderBy(Of T, U)(emu As IEnumerable(Of T), _  
    sortby As Func(Of T, U)) As IEnumerable(Of T)
```

```
<Extension> Function Select(Of T, S)(emu As IEnumerable(Of T), _  
    selection As Func(Of T, S)) As IEnumerable(Of S)
```

‘ 传统调用

```
Dim result = _  
    Select(OrderBy(Where(employees, filter1), sortby1), selection1)
```

‘ 扩展方法调用

```
Dim result = _  
    employees.Where(filter1).OrderBy(sortby1).Select(selection1)
```

查询序列操作符

- 采用扩展方法，将各种查询操作一一封装

映射	Select
限定	Where
检测	Any, All
联合	Join
分组	GroupBy
聚集	Count, Sum, Min, Max, Average
分割	Take
集合	Distinct, Union, Intersect, Except
排序	OrderBy, ThenBy

使用查询序列操作符进行查询

- 用Lambda函数或嵌套函数组成查询过程
- 用匿名类型作字段映射
- 扩展方法组成查询序列操作符

```
Dim myEmployees = GetEmployees() ' 填充数据
```

```
Dim result = myEmployees _  
    .Where((e) e.FirstName = "Harry" ) _  
    .OrderBy((e) e.Age ) _  
    .Select((e) New {Name := e.First & e.LastName, Age := e.Age })
```

查询包含

- Visual Basic 9.0引入更接近SQL的语法包装整个查询序列操作

‘ 查询序列语法

```
Dim result = myEmployees _  
    .Where((e) e.FirstName = "Harry" ) _  
    .OrderBy((e) e.Age ) _  
    .Select((e) New {Name := e.First & e.LastName, Age := e.Age })
```

‘ 查询包含语法

```
Dim result = _  
    Select New {Name := e.FirstName & e.LastName, Age := e.Age} _  
    From e In myEmployees _  
    Where e.FirstName = "Harry" OrderBy e.Age
```

您的潜力，我们的动力

Microsoft[®]
微软(中国)有限公司

DEMO

- 查询包含
- 查询本地对象

XML字面量

- Visual Basic 9.0引入全新的XML字面量，可将XML结构完整地引入VB

' XML字面量

```
Dim xbooks = _  
    <Books>  
        <Book author="Robbi ns">Debuggi ng Appl i cati ons</Book>  
        <Book author="Howard">Wri ti ng Secure Code</Book>  
    </Books>
```

- 遇到最外层的关闭标记前，无须续行符

洞和代码嵌入

- 使用“洞”可以动态确定标记和属性的名字
- 使用</>来关闭用洞定义的标记

```
Dim rootNode = "Books"

Dim xbooks = _
    <(rootNode)>
        <Book author="Robbi ns">Debuggi ng Appl i cati ons</Book>
        <Book author="Howard">Wri ti ng Secure Code</Book>
    </>
```

```
Dim attrName = "author"
Dim attrVal ue = "Robbi ns"

Dim xbook = <Book (attrName)=(attrVal ue)>Debuggi ng Apps</Book>
```

洞和代码嵌入

- 使用<%= %>标记来代码嵌入
- 可嵌入文本或另一段XML

```
Dim attrValue = "Robbins"  
Dim title = "Debugging Applications"  
Dim xbook = <Book author=(attrValue)><%= title %></Book>  
  
Dim rootNode = "Books"  
Dim xbooks = <(rootNode)><%= xbook %></>
```

洞和代码嵌入

- 甚至可以嵌入查询包含语句

```
Dim books As New List(Of Book)
books.Add(New Book {Author:="Robbins", Title:="Debugging Apps"})
books.Add(New Book {Author:="Howard", Title:="Writing Sec Code"})
books.Add(...)

Dim rootName = "Books"

Dim xbooks = _
    <(rootName)>
        <%= Select
            <Book author=(b.Author)><%= b.Title %></Book>
            From b In books Where b.Author = "Robbins" %>
        </>
```

XML后期绑定

- Visual Basic支持用标记和属性的名字直接作为XElement的属性进行访问

```
Dim xbooks = _  
    <Books>  
        <Book author="Robbins">Debugging Applications</Book>  
        <Book author="Howard">Writing Secure Code</Book>  
    </Books>
```

```
Dim titles = xbooks.Book  
Dim authors = xbooks.Book.@author
```

XML后期绑定

- XML还支持以...符号进行纵向搜索

```
Dim x = <_Root>  
    <A>  
        <B>some words</B>  
    </A>  
</Root>
```

```
Dim findb_1 = x.B ' 找不到，因为. 为横向搜索  
Dim findb_2 = x...B ' 纵向搜索
```

您的潜力，我们的动力

Microsoft
微软(中国)有限公司

DEMO

- XML字面量
- XML后期绑定

Visual Basic 动态编程概述

- 什么是动态编程？
 - 不在编译时限定
 - 按运行时类型操作
 - 执行动作可动态变更
 - Static typing when possible; dynamic typing when needed
- Visual Basic 2005所支持的动态特性
 - 后期绑定
 - 类型自动转换

您的潜力，我们的动力

Microsoft[®]
微软(中国)有限公司

Visual Basic 9.0新增动态特性

- 动态标识符
- 动态接口（鸭子类型）

动态标识符

- Visual Basic 9.0允许你用字符串表示对象属性、方法的名字

```
Dim o As Object = New Button()
```

```
‘ 后期绑定语法
```

```
o.Name = "Button1"
```

```
o.Text = "Hello World"
```

```
‘ 动态标识符语法
```

```
o.("Name") = "Button1"
```

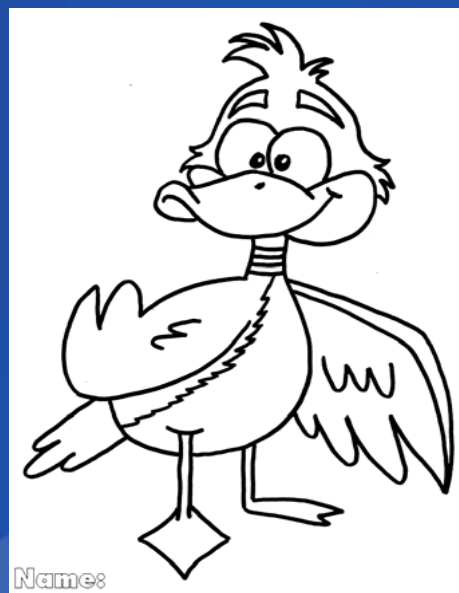
```
o.("Text") = "Hello World"
```

```
‘ 到运行时再决定属性的名字
```

```
o.(TextBox1.Text) = "Hello"
```

动态接口（鸭子类型）

- 如何判断一只动物是鸭子？



静态语言：如果它声明自己是鸭子，那就判定它是鸭子

动态语言：如果它走路像鸭子，说话也像鸭子，那它就是鸭子

动态接口（鸭子类型）

- 动态接口与普通接口不同，对象无须声明实现动态接口
- 只要定义有动态接口所需的成员，就可以应用于动态接口

```
Dynamic Interface IHasName  
    Property Name As String  
End Interface
```

```
Dim n As IHasName = New Button() ' Button有Name属性，可以！  
n.Name = "Button1"
```

```
n = New {Name := "Harry Potter"} ' 有Name属性，可以！  
n.Name = "The Goblet of Fire"
```

```
N = New ArrayList() ' 错误！ArrayList没有Name属性
```

动态接口与后期绑定的不同

- 后期绑定仅仅按名称工作，而动态接口则检查整个签名
- 编译器无法为后期绑定提供智能感知支持，但动态接口可以

您的潜力，我们的动力

Microsoft[®]
微软(中国)有限公司

DEMO

- 动态标识符

总结与前景分析


- Visual Basic 9.0在语法层面充分更新
- 减少程序员记忆负担
- 更少代码完成更多工作
- 更高抽象，更多动态
- 将成为XML开发最得力的助手，可替代XSLT，XPath，XQuery的大部分功能

资源

- Visual Basic 下一代主页
- <http://msdn.microsoft.com/vbasic/future/>
- 您可以找到
 - Linq概述
 - XLinq和DLinq概述
 - Visual Basic 9.0 Linq 预览版下载



Question & Answer

如需提出问题，请单击“提问”按钮并在随后显示的浮动面板中输入问题内容。一旦完成问题输入后，请单击“提问”按钮。

 **问题和解答 (无问题)** ▲ ×

在此会议中尚未解答任何问题。

要向演示者提问，请在此处键入问

提问(A)

删除(D)

问题管理器(Q)

您的潜力，我们的动力

Microsoft®
微软(中国)有限公司

Microsoft®

msdn


MSDN Webcasts