您的潜力，我们的动力

Microsoft
微软（中国）有限公司

Visual Basic 2005开发技巧系列课程

Visual Basic 2005 泛型编程介绍

讲师：施凡

msdn

MSDN Webcasts

摘要

- 泛型的引入和意义所在
- VB泛型的语法
- .NET Framework 2.0泛型API简介
- 总结

泛型之前的时代

- 对一种类型编写的算法不能用于其他类型

```
Function Algorithm1(x As Integer) As Integer
Dim s As String
Algorithm1(s) ‘ 编译错误
```

- 使用Object可以包容一切类型，但具体类型要到编译时确定，缺乏类型安全性

```
Dim a As New ArrayList()
a.Add(123)
a.Add(Date.Now) ‘无类型检查
Dim b As Integer = CInt(a(0)) ‘需要运行时类型转换
Dim c As Integer = CInt(a(1)) ‘潜在运行时错误
```

泛型给出的解决方案

- 利用类型参数，让多种类型共享代码

```
Function Algorithm1(Of T)(x As T) As T
Dim s As String = Algorithm1(Of String)("s")
Dim i As Integer = Algorithm1(Of Integer)(1)
```

- 提供强类型支持

```
Dim gl As New List(Of Integer)
gl.Add(1)
gl.Add(Date.Now) '编译错误
Dim i As Integer = gl(0) '无需类型转换
```

泛型的含义

- 泛型（Generic）主要指将类型参数化。
- 泛型类型（Generic Type）和泛型方法（Generic Method）不基于具体类型，而是基于类型参数编写代码
- 通过赋予类型参数具体的类型，已建造类型及已建造方法自动用具体赋予的类型实参代替类型参数以完成代码
- 建造的过程是在JIT编译时进行

泛型的优越之处

- 重用针对特定类型的代码
- 确保类型安全性，提供编译时类型检查和IDE智能感知特性
- 减少装箱/拆箱和类型转换操作带来的性能损耗
- 是未来Visual Basic新特性的基础

DEMO

- 编译时类型检查
- 智能感知支持
- 性能对比

可声明为泛型的语言元素

- 可以将以下类型声明为泛型类型
  - 类（Class）
  - 接口（Interface）
  - 结构（Structure）
  - 委托（Delegate）
- 可以将方法成员声明为泛型方法
  - 子程序（Sub）
  - 函数（Function）
- 其他类型和成员不能声明为泛型

类型参数与类型实参

- Of关键字同时用于声明类型参数与指定类型实参
-

```
Public Class GenericType(Of T)
Public Delegate Function GenericDelegate(Of T1, T2)(arg As T1)As T2
Public Sub GenericMethod(Of T)(obj As T)
```

-

```
Dim x As GenericType(Of Integer) '用具体类型作为类型实参
GenericMethod(Of IComparable)(t) '用接口作为类型实参
```

-

```
Public Class GenericType2(Of T2) 'T2对GenericType2来说是类型参数
    Inherits GenericType(Of T2) '而对GenericType来说是类型实参
End Class
```

泛型类型和已建造类型


```
Public Class A(Of T1, T2) ‘泛型类型
Public Class B(Of T)
    ‘下面用到的2个A的已建造类型是开类型
    Dim a1 As A(Of T, Integer)
    Dim a2 As A(Of T, T)
    ‘下面用到的A的已建造类型是闭类型
    Dim a3 As A(Of String , Object)
End Class
```

```
Dim objlist As List(Of Object)
objlist = New List(Of Integer) ‘错误
虽然Integer是Object的子类，但List(Of Integer)
不是List(Of Object)的子类
Dim l As List(Of Integer)
t = GetType(List(Of String))
l2 = CType(l1, List(Of Object))
```

- 定义有 体和委托称为
- 赋予类 类型就展开成 d Type）
- 已建造 种。开类型的 类型参数，闭 实类型
- 一个泛 间没有继承关系。

泛型方法和类型实参的自动推定

- 为方法指定类型参数，就定义了泛型方法

```
Public Sub Swap(Of T)(ByRef a As T, ByRef b Abs T)
    Dim c As T
    c = a
    a = b
    b = c
End Sub
```

- 调用方法时，如果编译器可以推定类型实参的类型，就不必用Of语句指定

```
Dim x, y As Integer
Swap(Of Integer)(x, y) ‘可以
Swap(x, y) ‘也可以，类型实参自动推定
Dim u As Double
Swap(x, u) ‘错误，无法推定类型实参
```

约束

- 约束的目的是减少类型参数的取值范围
- 约束的类型：基类/接口约束、New约束、值类型约束和引用类型约束
- 每一种约束都会给类型参数带来相应的负作用
- 约束的通用语法：

Declaration (**Of** *TypeParam1* **As** *Constraints*, *TypeParams2…*)

基类/接口约束

- 作用：让类型参数的取值范围限定为某个类的子类或实现某接口的所有类型中
- 副作用：该类型参数的变量可以使用约束基类或接口的成员
-

```
Class SomeType1(Of T As Exception) ‘基类约束
Class SomeType2(Of T As IComparable) ‘接口约束

Dim x As SomeType1(Of Integer) ‘错误，Integer不是Exception的子类
Dim y As SomeType2(Of String) ‘正确，String实现了IComparable

Sub SomeMethod1(Of T1 As IDisposable, T2)(o1 As T1, o2 As T2)
    o1.Dispose()  ‘正确，T1约束了IDisposable
    o2.Dispose()  ‘错误，T2没有约束，因此只包含System.Object的成员
End Sub
```

  - System.MulticastDelegate

New约束

- 作用：将类型参数限定为带有一个公共无参数构造器的类型
- 副作用：可以在代码中创建类型参数的实例

```
Class SomeType(Of T As New) ‘用New关键字作约束
    Dim obj As T
    Public Sub Test()
        obj = New T() ‘约束了New的类型参数，可以创建实例
    End Sub
End Class

Dim x As SomeType(Of Integer) ‘可以，Int32有公共无参数构造器
Dim y As SomeType(Of String) ‘错误，String没有公共无参数构造器
```

值类型约束和引用类型约束

- 作用：将类型参数限定为值类型（Nullable除外）或引用类型
- 副作用：在代码中，可将类型参数当作值类型或引用类型看待

```
‘用Structure关键字作约束值类型, Class约束引用类型
Class SomeType(Of V As Structure, R As Class)
    Dim val As V, ref As R
    Public Sub Test(o As Object)
        ref = TryCast(o, R) ‘可以，R被当作引用类型
        val = New V() ‘值类型一定可以用New初始化
    End Sub
End Class
```

多约束

- 要进行多种约束，只需要把约束的内容用大括号{}括起来，用逗号分隔

```
'T只能取值为含有公共无参数构造器、实现了IComparable的引用类型
Class SomeType(Of T As {New, Class, IComparable})

'T必须同时实现IList和IDisposable
Function SomeMethod(Of T As {IList, IDisposable})(obj As T)
```

关于约束使用的建议

- 约束是.NET泛型解决类型安全性问题提出的一种妥协方案。
- 尽管副作用可以给约束带来更丰富的操作，但会减少泛型类型参数的取值范围，因此建议少用约束。
- 通过一些设计可以避免使用约束。

.NET Framework 2.0泛型API简介

- Generic Nullable
- 公共泛型接口
- 公共泛型委托
- System.Collections.Generic 泛型集合

Generic Nullable

- 以前版本中，值类型不能取值为空
- 但在数据库等应作用中，这种需求很多
- 使用System.Generic Nullable，值类型也可以等于空值

‘Generic Nullable的定义：
Public Structure **Nullable**(Of T As Structure)

Generic Nullable的用法

- 可将原类型的数值直接转换成相应的Nullable类型
- 用HasValue属性判断是否为空
- 从Value属性可得到得到原类型的数值

```
Dim ni As Nullalbe(Of Integer)
ni = 12 ‘可直接将相应类型转化为Nullable
If ni.HasValue() Then Console.WriteLine(ni.Value)

Ni = Nothing ‘可赋值为Nothing来获得空值
```

泛型接口

- 许多常用接口提供了泛型版本，便于让类型提供强类型的实现

非泛型版本	泛型版本
IComparable	IComparable(Of T)
无	IEquatable(Of T)
ICollection	ICollection(Of T)
IEnumerable	IEnumerable(Of T)
IDictionary	IDictionary(Of TKey, TValue)

泛型接口

- 实现泛型接口可以提供强类型的方法

```
Public Class MyClass
    Implements IComparable(Of MyClass)

    private m_AField As Integer
    Public Function CompareTo(other As MyClass) As Integer _
        Implements IComparable(Of MyClass).CompareTo
        ‘无需类型检查和转换，代码更直接
        Return Me.m_AField.CompareTo(other.m_AField)
    End Function
End Class

‘自己也可以创建泛型接口，便于实现类型安全的代码
```

泛型委托

- 系统新提供的四个泛型委托，用于抽象四种不同任务的函数
- 只要函数的签名的类型和类型实参符合，就可以赋值给委托的对象

```
Public Delegate Sub Action(Of T)(obj As T)
Public Delegate Function Predicate(Of T)(obj As T) As Boolean
Public Delegate Function Converter(Of TInput, TOutput) _
    (input As TInput) As TOutput
Public Delegate Function Comparison(Of T)(x As T, y As T) As Integer
```

泛型EventHandler

- 所有EventHandler都有相似的签名，但因为EventArgs的类型不同，而无法重用同一个委托。
- 有了泛型，这个问题就有了优美的解决方法

```
Public Delegate Sub EventHandler(Of TEventArgs As EventArgs) _
    (sender As Object, e As TEventArgs)

‘需要自定义事件时
Public Class MyEventArgs
    Inherits EventArgs
    ….
End Class

Public Event MyEvent As EventHandler(Of MyEventArgs)
```

泛型集合

- 提供强类型的List, Queue, Stack和Dictionary等常用数据结构
- 在值类型上有更好的性能
- 使用更方便

```
Dim l As List(Of Integer)
Dim d As Dictionary(Of String, Integer)
Dim s As Stack(Of Double)
Dim q As Queue(Of TreeNode)

此外还提供自定义强类型集合的Collection(Of T)
Public Class MyCollection : Inherits Collection(Of MyCustomType)
```

泛型List

- 取代ArrayList的地位；充分利用了所有泛型特性的典范

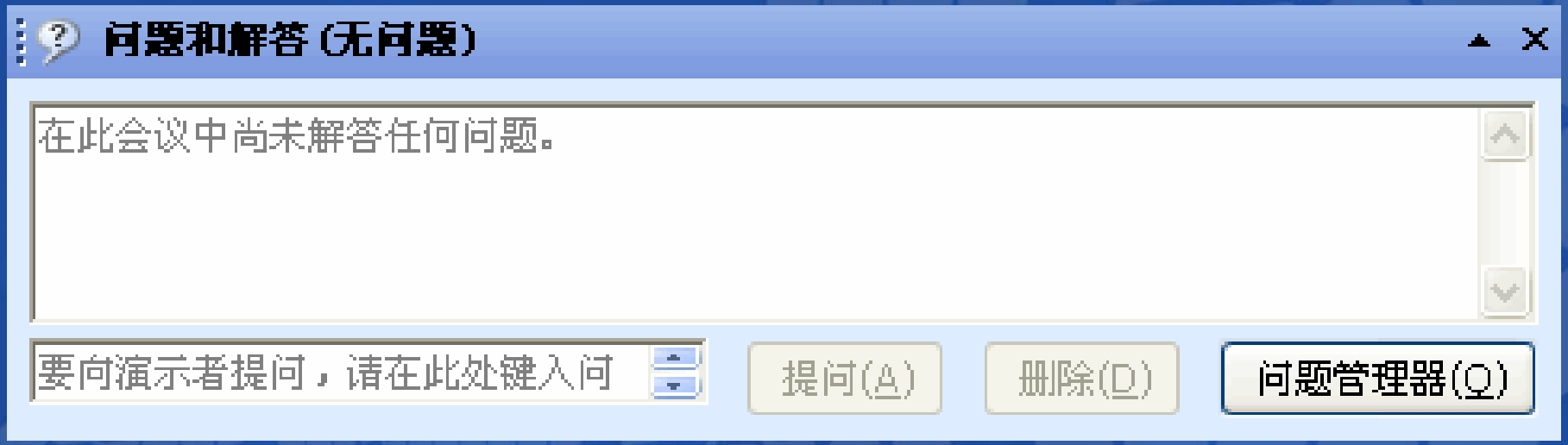
```
Dim l1 As New List(Of Integer)

‘强类型的方法
l1.Add(123)

‘利用泛型委托完成优美的类型转换
Dim l2 As List(Of String)
l2 = l1.ConvertAll(Of String)(AddressOf Convert.ToString)

‘ 与“VBF算符”结合
Dim l3 As List(Of Integer)
l3 = l1.FindAll(GreaterThan(50) And Not EqualTo(100))
```

总结

- 泛型给.NET开发带来了新的内容。
- 泛型可实现类型安全的代码，可减少装箱和类型转换带来的性能损耗。
- 泛型类、接口和委托的引入，使得抽象能力更进一步加强。
- 泛型方法为类型无关的算法提供可能，类型实参的自动推定让代码保持简洁。

Q&A

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。




您的潜力，我们的动力
Microsoft
微软（中国）有限公司
Microsoft
msdn
MSDN Webcasts