DEV342
Visual Basic 2005:
应用程序框架
和高级语言特性

施凡
微软 Visual Basic MV

听起来很熟悉？

- 创建一个组件，可以将事件日志记录到多个事件源
- 创建ini文件或进行复杂的注册表操作保存用户设置
- 一遍又一遍地编写用户身份验证的代码
- 创建强类型的集合包装类型
- 在网上搜索各种代码片断，然后粘贴到你的程序里
- 抱怨窗体设计器生成的代码与自己的代码混在一起

提纲

- 在一个新起点上架构您的应用程序
  - 自定义用户身份验证
  - 使用配置信息
  - 使用应用程序日志
- 自定义IDE的使用体验
  - 插入代码段或扩展代码段库
  - 自定义项目和项目元素模板
- 语言新特性
  - 泛型
  - 编译器警告
  - 附属类型
  - 其它语言特性

演示

自定义用户身份验证

My命名空间层次图

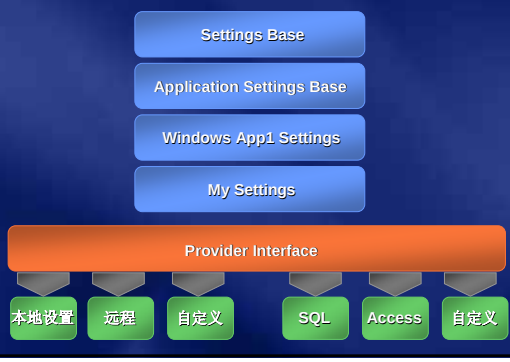
配置信息和用户设置
你现在用的方法……

- 用App.config 保存XML格式的配置信息
  - 某些情况下，读写XML很痛苦
- 用Framework中的类可以读取配置信息
  - 但是不能修改和写入
- 可扩展以支持更多功能
  - 方法还算直观，但也不是特别简单

配置信息和用户设置
我们现在提供的方法……

- Framework类库支持配置信息的读写
- 配置信息的强类型验证
- 有智能感知的帮助
- 支持应用程序级或用户级分别设置
- 在部分信任级别下工作
- 可扩展的Provider和验证模式
- 客户端和Web共享同一基础框架
- 可从项目设计器窗口轻松访问
- 对各种专用配置提供支持
  - 连接字符串，Web服务代理类等.

配置系统的架构



实战设置功能

- 应用程序级设置
- 用户级设置

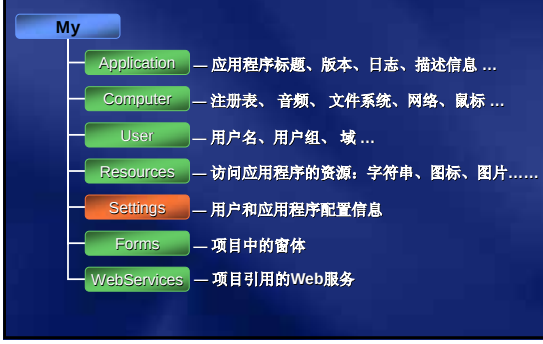
```
Myapp.exe.config

<applicationSettings>
…
</applicationSettings>
```

```
fred.config
ethel.config
gladys.config

<userSettings>
…
</userSettings>
```

My命名空间层次图



实战设置功能

```
My.Settings.UseHighQuality = True
' Settings 在第一次访问的时候自动载入配置数据
```

- 保存

```
My.Settings.UseHighQuality = True
My.Settings.Save()
```

```
Private Sub Settings_SettingChanging(ByVal sender As Object, _
        ByVal e As SettingsArg) Handles MyBase.SettingChanging

    If e.SettingName = "SignatureFile" Then
        If Not My.Computer.FileSystem.FileExists(e.Setting.Value)Then
            ' 取消该事件
        End If
    End If
End Sub
```

演示

使用设置功能

演示

使用应用程序事件日志

可扩展代码段库

- 已内置超过500个代码段！
- 包含多种接口实现和常见设计模式
- 提供标准框架便于自行编写代码段
- 可配置右键菜单的菜单项
- 多个代码段存放路径
- 已结合在各种第三方工具中
  - Refactor!、CodeWise、等等
- 网上还有代码段编辑器
  - 可创建或编辑代码段

演示

编写自定义代码段

语言新特性

- Using语句
- Continue语句
- Global关键字
- 访问器可见性修饰
- 附属类型
- 无符号整数
- 运算符重载
- 泛型
- 编译器警告

Using 语句

Acquire, Execute, Release

- 一种而正确释放对象资源的快速写法
  - 比使用Try、Catch、Finally更易读
- 与Dispose-Finalize模式配套使用

```
‘Using 语句块正确处置对象资源
Using fStr As New FileStream(path, FileMode.Append)

  For i As Integer = 0 To fStr.Length
      fStr.ReadByte()
  Next

  ‘到达块尾部时，文件流自动被关闭
End Using
```

Continue 语句
直接跳到下一次循环

- 让循环逻辑更清楚

```
For j As Integer = 0 to 5000
  While matrix(j) IsNot thisValue
    If matrix(j) Is thatValue

        ‘ 直接跳到下一个j
        Continue For

      End If

      Graph(j)
  End While
Next j
```

Global 关键字
访问根（全局）命名空间

- 解决命名冲突问题
- 代码生成器的更佳选择

```
Namespace HeadTrax
  Class Form1
      Inherits Windows.Forms.Forms

  Sub LastName(nm As String)
    Global.Microsoft.VisualBasic.Left(nm)
  End Sub

  End Class
End Namespace
```

访问器可见性修饰符
让 Get 和 Set 有不同的可见性

- 让所有对字段的访问都经过get和set
- 让字段赋值获得更好的验证

```
Property Salary() As Integer
  Get
      Return mSalary
  End Get

  Private Set( value As Integer)
      If value < 0 Then
        Throw New Exception(“错误”)
      End If
  End Set
End Property
```

附属类型（Partial Types）
同一类型，不同文件

- 设计器生成的代码可以分到另外一个文件
- 将来……

```
Public Class Form1
  Inherits Windows.Forms.Form
      ‘ 你的代码
End Class

Partial Class Form1
      ‘ 设计器的代码
      Sub InitializeComponent()
        ‘ 控件初始化
      End Sub
End Class
```

运算符重载
创建简洁的使用方式

```
Class Addr
  Private mString As String

  Property Value() As String
    Get
      Return mString
    End Get
    Set (value As String)
      If Valid(value) Then
        mString = value
      End If
    End Set

  Shared Operator &(ad1 As Addr, ad2 As Addr) As Addr

    Return New Addr(ad1.Value & ad2.Value)
  End Operator
End Class
```

无符号整型
现已全面支持

- 完全由平台提供
- 主要用于Windows API调用和交互

```
Dim sb As SByte = -4      ‘这个是Byte的有符号版本
Dim us As UShort
Dim ui As UInteger
Dim ul As ULong

‘ VisualBasic 的内置函数现已完全支持
If IsNumeric(uInt) Then
      ‘ 返回True
End If
```

泛型

```
Public Class List(Of TItem)
   Private elements() As TItem
   Private count As Integer

   Public Sub Add(element As TItem)
      If (count = elements.Length) Then _
        Resize(count * 2)
      count += 1
      elements(count) = element
   End Sub

   Default Public Property Indexer(index As Integer) As TItem
      Get : Return elements(index)  : End Get
   End
   Pub
      (
   End
End Cl
```

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

intList.Add(1)          ' 没有装箱
intList.Add(2)          ' 没有装箱
intList.Add("Three")    ' 编译错误

Dim i As Integer = intList(0)    ' 没有类型转换
```

泛型

- 编译时检查
  - 消除运行时错误
- 提高性能
  - 防止装箱和类型转换
- 代码重用
  - 方便创建强类型集合
- 用于构建Framework中常见的数据结构
  - Dictionary、Queue、List、Stack等等.

Visual Basic 编译器警告
对运行时行为的前期警告

- 交叠的Catch块或Case语句
- 递归属性访问
- 未使用的Imports语句
- 未使用的本地变量
- 函数或运算符重载过程未返回值
- 引用类型上可能的空引用
- 分解Option Strict
  - 后期绑定
  - 隐式类型转换
  - 等等

Background Worker

```
Structure BackgroundArgs
    Dim arg1 As Integer
    Dim arg2 As String
End Structure

Private Sub Button1_Click(…) Handles Button1.Click
    Dim args As BackgroundArgs
    BackgroundWorker1.RunWorkerAsync(args)
End Sub

Private Sub BackgroundWorker1_DoWork(…) Handles
BackgroundWorker1.DoWork
    ' 这是一个后台运行的线程
    Dim args As BackgroundArgs
    args = CType(e.Argument, BackgroundArgs)
    ' 现在，在后台线程做点事情
    e.Result = <some object>
End Sub

Private Sub BackgroundWorker1_RunWorkerCompleted(…) Handles
Background…
    result = e.Result
End Sub
```

Visual Basic 2005
专为应用程序框架而设计

- 应用程序架构的新起点
  - 基于可扩展的类库
  - 以最佳用户体验为目标
- 自定义IDE的使用体验
  - 项目和项目元素模板
  - 可扩展代码段库
- 高级语言特性
  - 泛型、附属类型、运算符重载等
  - 帮助设计更出色的类库

社区资源

- 开发中心主页:
  http://msdn.microsoft.com/vbasic
- 实用下载工具主页
  http://msdn.microsoft.com/vbasic/downloads/2005/
  - **Refactor! For VB2005**
  - **Snippet Editor**