**VB.Net Notepad**

目录

[前言简介 1](#_Toc466884752)

[一、 面向对象 1](#_Toc466884753)

[通用规则 2](#_Toc466884754)

[一、 快捷键 2](#_Toc466884755)

[二、 进制表示 2](#_Toc466884756)

[三、 加载窗体 2](#_Toc466884757)

[四、 环境变量 2](#_Toc466884758)

[1. 路径 2](#_Toc466884759)

[2. 获取用户名 2](#_Toc466884760)

[3. 当前程序路径名称 3](#_Toc466884761)

[五、 Spy++查找句柄 3](#_Toc466884762)

[1. 模拟键盘输入 3](#_Toc466884763)

[六、 进程和线程 4](#_Toc466884764)

[1. 线程和进程 4](#_Toc466884765)

[七、 VS.net平台框架 5](#_Toc466884766)

[1. .Net Framework框架 5](#_Toc466884767)

[2. 编译平台 6](#_Toc466884768)

[3. 托管代码 6](#_Toc466884769)

[八、 调试 7](#_Toc466884770)

[1. 调试快捷键 7](#_Toc466884771)

[2. 附加到进程 7](#_Toc466884772)

[3. 程序调试模式 8](#_Toc466884773)

[语法规则 9](#_Toc466884774)

[一、关键字 9](#_Toc466884775)

[二、设置字体 9](#_Toc466884776)

[数据类型 10](#_Toc466884777)

[一、数值类型 10](#_Toc466884778)

[1、unsigned 和 signed 数值类型 10](#_Toc466884779)

[2、非整数 数值类型 10](#_Toc466884780)

[3、System 11](#_Toc466884781)

[4、数值变量类型符 11](#_Toc466884782)

[5、文字式数值类型符 11](#_Toc466884783)

[6、字节（Byte）数据类型 11](#_Toc466884784)

[二、字符类型 11](#_Toc466884785)

[1、 Char类型 11](#_Toc466884786)

[2、 String类型 11](#_Toc466884787)

[三、日期类型 12](#_Toc466884788)

[四、布尔类型 12](#_Toc466884789)

[五、对象类型 12](#_Toc466884790)

[六、自定义结构类型 12](#_Toc466884791)

[七、数据类型转换 13](#_Toc466884792)

[1.取整函数 13](#_Toc466884793)

[2.CType、DirectCast、TryCast(被转换数据，数据类型X） 13](#_Toc466884794)

[3.Parse和ToString方法 13](#_Toc466884795)

[4.Convert类 13](#_Toc466884796)

[八、判断变量数据类型 14](#_Toc466884797)

[字符字母 15](#_Toc466884798)

[一、字符处理 15](#_Toc466884799)

[二、汉字首字母 15](#_Toc466884800)

[日期时间 19](#_Toc466884801)

[一、当前时间 19](#_Toc466884802)

[1、DateTime 19](#_Toc466884803)

[2、Now 19](#_Toc466884804)

[3、Today 19](#_Toc466884805)

[4、TimeOfDay 19](#_Toc466884806)

[5、实例 20](#_Toc466884807)

[二、时间间隔 20](#_Toc466884808)

[1、Sleep 20](#_Toc466884809)

[2、Now.Subtract 20](#_Toc466884810)

[3、实例 21](#_Toc466884811)

[键盘鼠标 22](#_Toc466884812)

[一、模拟按键 22](#_Toc466884813)

[1、键盘键与虚拟键码对照表 22](#_Toc466884814)

[2、模拟按键 23](#_Toc466884815)

[二、模拟鼠标 23](#_Toc466884816)

[窗体控件 24](#_Toc466884817)

[一、 窗体 24](#_Toc466884818)

[1. 程序在windows的位置设置 24](#_Toc466884819)

[2. 使用PrintForm打印窗体 24](#_Toc466884820)

[二、 控件 25](#_Toc466884821)

[1. 动态增加控件 25](#_Toc466884822)

[2. 控件的父控件 26](#_Toc466884823)

[文件目录 27](#_Toc466884824)

[一、使用VBScript 27](#_Toc466884825)

[二、使用DirectoryInfo 27](#_Toc466884826)

[三、目录和文件 27](#_Toc466884827)

[四、快捷方式 28](#_Toc466884828)

[1.引用COM组件 28](#_Toc466884829)

[2.CreateObject of WScript.Shell 28](#_Toc466884830)

[五、关联文件打开程序 29](#_Toc466884831)

[读写操作 30](#_Toc466884832)

[一、二进制读写 30](#_Toc466884833)

[1、读操作示例 30](#_Toc466884834)

[类和对象 31](#_Toc466884835)

[一、委托 31](#_Toc466884836)

[1、自定义委托 31](#_Toc466884837)

[2、系统委托 31](#_Toc466884838)

[二、获取活动程序对象 32](#_Toc466884839)

[1.Marshal. GetActiveObject 方法 32](#_Toc466884840)

[2.AutomationFactory. CreateObject 方法 (String) 32](#_Toc466884841)

[通信服务 33](#_Toc466884842)

[一、WCF服务 33](#_Toc466884843)

[二、Windows服务 33](#_Toc466884844)

[三、通信 33](#_Toc466884845)

[句柄进程 34](#_Toc466884846)

[一、程序句柄相关 34](#_Toc466884847)

[1、本程序句柄 34](#_Toc466884848)

[二、获取系统句柄 34](#_Toc466884849)

[1、输入焦点（光标位置）对象句柄获得 34](#_Toc466884850)

[三、进程ID 34](#_Toc466884851)

[1、隐藏应用程序 34](#_Toc466884852)

[2、获取进程 34](#_Toc466884853)

[编程钩子 36](#_Toc466884854)

[前言 36](#_Toc466884855)

[一、钩子的机制及类型 36](#_Toc466884856)

[二、vb编程中钩子的实现 37](#_Toc466884857)

[1钩子函数(hook function)的格式 37](#_Toc466884858)

[2钩子的安装及执行 37](#_Toc466884859)

[3钩子安装应注意的问题 38](#_Toc466884860)

[三、实例--键盘消息的拦截 39](#_Toc466884861)

[四、总结 40](#_Toc466884862)

[数据系统 41](#_Toc466884863)

[一、Access数据库 41](#_Toc466884864)

[1 ADO.Net对象 41](#_Toc466884865)

[2连接数据库 41](#_Toc466884866)

[3删除数据 41](#_Toc466884867)

[4修改数据 42](#_Toc466884868)

[5增加数据 42](#_Toc466884869)

[并行编程 44](#_Toc466884870)

[外部调用 45](#_Toc466884871)

[1. Shell 方法 45](#_Toc466884872)

[2. Process类 45](#_Toc466884873)

# 前言简介

## 面向对象

目前主流编程语言都是面向对象编程的。而面向对象的关键是要有对象、类。

.NET FrameWork框架提供的类库对于编程人员而言是极大便利，将我们所需要的功能全部封装起来。这单单对于编程是很便利的，但是想要深入了解编程原理、计算机底层结构，它将成为了一种障碍。这些涉及计算机底层操作的设计都已经封装到相应类库中，方便使用，但却没办法去查看它，或者正是由于这种便利性，程序员已经不太愿意去研究它的实现机理而只需要了解其功能。

正是面向对象编程语言的这种特性，以后编程也许会越来越脱离计算机物理层而成为一项单独的技巧，程序员只需将心思单独放在实现程序的业务功能上。

（注：笔者个人见解）

## 面向服务

# 通用规则

## 快捷键

F1…………帮助

F2…………对象浏览器

Alt+F11……Vba编辑器

## 进制表示

BIN 二进制（B）

OCT 八进制（O）

DEC 十进制（D）

HEX 十六进制（H）

## 加载窗体

窗体先加载Load事件 再加载shown事件（所以在load事件中使用Me.Hide() 无效）

Me.WindowState = FormWindowState.Minimized ''窗体显示格式/可视状态

Me.Visible = False ''程序可见性

Me.ShowInTaskbar = False ''程序在任务栏中的可见性

Me.Hide() ''隐藏程序，在任务管理器中不可见（不能隐藏进程）

## 环境变量

简介：System.Environment不可继承，实例化该对象无意义，可直接使用。

### 路径

IO.File.Exists(path) = True ''查看路径是否存在

Dim deskTopStr As String ''桌面路径

deskTopStr = System.Environment.GetFolderPath(Environment.SpecialFolder.Desktop)

Dim path As String

path = Application.StartupPath ''程序当前路径

path = path.Substring(0, path.LastIndexOf("\")) ''上级目录

### 获取用户名

Dim sad As New System.Security.Principal.WindowsPrincipal (System.Security.Principal.WindowsIdentity.GetCurrent)

Dim nId As String

nId = sad.Identity.Name ''获取用户名

nId = System.Environment.GetEnvironmentVariable("username") ''环境变量

### 当前程序路径名称

''当前程序目录

System.Environment.CurrentDirectory/My.Application.Info.DirectoryPath

''当前程序名称

Process.GetCurrentProcess.ProcessName

''当前程序内部名称和描述

My.Application.Info.AssemblyName //My.Application.Info.Description

## Spy++查找句柄

### 模拟键盘输入

' Get a handle to an application window.

Declare Auto Function FindWindow Lib "USER32.DLL" ( \_

ByVal lpClassName As String, \_

ByVal lpWindowName As String) As IntPtr

' Activate an application window.

Declare Auto Function SetForegroundWindow Lib "USER32.DLL" \_

(ByVal hWnd As IntPtr) As Boolean

' Send a series of key presses to the Calculator application.

Private Sub button1\_Click(ByVal sender As Object, \_

* ByVal e As EventArgs) Handles button1.Click
* ' Get a handle to the Calculator application. The window class
* ' and window name were obtained using the Spy++ tool.
* Dim calculatorHandle As IntPtr = FindWindow("CalcFrame", "Calculator")
* ' Verify that Calculator is a running process.
* If calculatorHandle = IntPtr.Zero Then
* MsgBox("Calculator is not running.")
* Return
* End If
* ' Make Calculator the foreground application and send it
* ' a set of calculations.
* SetForegroundWindow(calculatorHandle)
* SendKeys.SendWait("111")
* SendKeys.SendWait("\*")
* SendKeys.SendWait("11")
* SendKeys.SendWait("=")

End Sub

## 进程和线程

### 线程和进程

在计算机科学中，“线程”和“进程”是两个相关的概念。 二者都表示必须按特定顺序执行的指令序列。 但是不同线程或进程中的指令可以并行执行。

进程存在于操作系统内，并对应于用户可看作为程序或应用程序的事物。 另一方面，线程存在于进程内。 因此，线程有时也称作“轻量进程”。 每个进程都由一个或多个线程组成。

多个进程的存在使得计算机能够一次执行多个任务。 而多个线程的存在使得进程能够分解工作以便并行执行。 在多处理器计算机上，进程或线程可以在不同的处理器中运行。 这使得真正的并行处理成为可能。

并不总是能够完全并行处理。 有时候必须要同步线程。 一个线程可能必须等待另一个线程的结果，或者一个线程可能需要独占访问另一个线程正在使用的资源。 同步问题是多线程应用程序中出现 Bug 的一个常见原因。 有时候线程可能最终等待的是永远不会变得可用的资源。 这导致了一种称为“死锁”的状况。

Visual Studio 调试器为调试线程和进程提供了功能强大但易于使用的工具。

## VS.net平台框架

### .Net Framework框架

**.Net Framework**

### 编译平台

AnyCpu、x64、x86默认的三个平台。

相同编译平台的dll才能被调用，否则会报错。

（相关内容引申：32位Dll不能被64位的程序调用）

### 托管代码

托管代码 (managed code) :由公共语言运行库环境（而不是直接由操作系统）执行的代码。托管代码应用程序可以获得公共语言运行库服务，例如自动垃圾回收、运行库类型检查和安全支持等。这些服务帮助提供独立于平台和语言的、统一的托管代码应用程序行为。

非托管代码(Unmanaged Code):在公共语言运行库环境的外部，由操作系统直接执行的代码。非托管代码必须提供自己的垃圾回收、类型检查、安全支持等服务；它与托管代码不同，后者从公共语言运行库中获得这些服务。非托管代码的英文名是Unmanaged Code ，它是在公共语言运行库环境的外部，由操作系统直接执行的代码。

托管和非托管是修饰内存的。托管的意思，你不用直接操作内存，你需要的时候跟我说。我替你申请，然后给你用，你用完可以告诉我，我帮你释放，如果你忙，忘记告诉我了，我也会在定期去帮你释放的。 这就是托管，你打交道的不是直接的内存，而是.net clr。

非托管的意思就是你要自己负责管理内存，这里所说的内存管理。实际上只是堆上的内存管理，栈内存和以前的一样，函数退出则释放，heap上的内存，非托管内存需要自己分配，调用构造函数(c需要，c++里用new替代这部操作了)，使用完毕后，需要自己释放这个内存，如果你不小心，把指向内存的指针弄丢了，就造成内存泄露了，这个内存泄露在你程序退出之前是无法弥补的，所以要小心。(virtualMalloc的情况不在此描述了。因为分配的也不是heap上的内存，属于扩展内存空间，题外话了。)

简单说，托管的意思是托管内存，但多一层必然会慢，这个么，微软早考虑好了，硬件升级，大家掏钱就能解决的问题不是问题。带来的好处是托管内存不会有泄漏的危险。

托管代码被编译成MSIL后在.net framework下运行，同操作系统底层的交互都交给.net framework来处理。非托管代码就是脱离了.net framework的管制，直接同底层API打交道，自己管理自己的内存和安全机制等东西。而托管代码就不管这些，全都由.net framework去完成 vb.net，C#等写的程序是托管程序，VC++可以写托管程序，如果用到了内存管理，则只能编译为非托管程序

这些东西MSDN上写得很清楚,如果真想深入了解的话不妨好好看看，或者也可以看看C#、.net框架设计或者Effective C#之类的书籍。都是较好的介绍。

http://bbs.csdn.net/topics/330164764

## 调试

### 调试快捷键

F8 逐语句

F10 逐过程

F5 继续

F9 断点

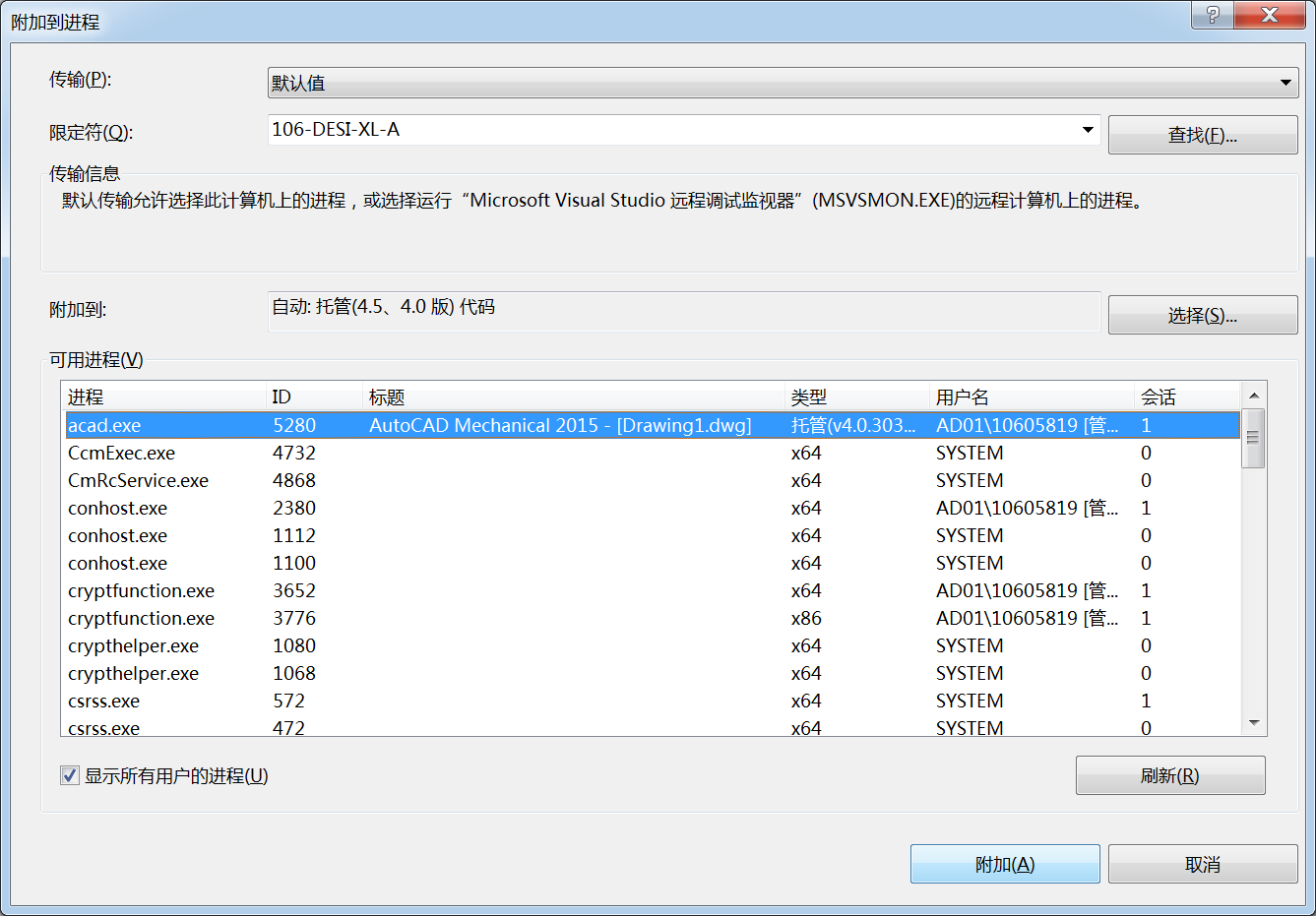
Ctrl+Break 中断

Ctrl+Alt+Break 停止调试

### 附加到进程

步骤：a)打开程序/服务（例cad.exe）；

b)VS工具—附加到进程—选择”cad.exe”（见下图）；

 b)在cad命令行中输入NetLoad—加载项目生成的dll；

c)注：所加载的dll应该与源代码保持一致（断点无要求）；

注：从调试服务中引申过来，其它程序也应有借鉴意义

### 程序调试模式

Dim ss As System.Reflection.Assembly = System.Reflection.Assembly.LoadFile(System.Environment.CurrentDirectory & "\" & Process.GetCurrentProcess.ProcessName & ".exe")

Dim temp() As DebuggableAttribute = ss.GetCustomAttributes(GetType(DebuggableAttribute), True)

If temp(0).IsJITTrackingEnabled = True Then

MessageBox.Show("IsJITTrackingEnabled=True")

Debug.Print("122")

End If

If temp(0).IsJITOptimizerDisabled = True Then

MessageBox.Show("IsJITOptimizerDisabled=True")

Debug.Print("122")

End If

# 语法规则

## 一、关键字

Andalso 和 orelse替代（and 和 or）

## 二、设置字体

Dim strName As String ''字体Name

strName = "宋体"

Dim myFontFamily As System.Drawing.FontFamily

myFontFamily = New FontFamily(strName)

Dim sngSize = 14 ''字体Size

Dim myFontStyle As System.Drawing.FontStyle ''字体Underline、Strikeout、Bold、Italic

Dim bolUnder = True

Dim bolStrike = True

Dim bolBold = True

Dim bolItalic = True

If bolUnder Then

myFontStyle = myFontStyle Or System.Drawing.FontStyle.Underline

End If

If bolStrike Then

myFontStyle = myFontStyle Or System.Drawing.FontStyle.Strikeout

End If

If bolBold Then

myFontStyle = myFontStyle Or System.Drawing.FontStyle.Bold

End If

If bolItalic Then

myFontStyle = myFontStyle Or System.Drawing.FontStyle.Italic

End If

Dim myColor As System.Drawing.Color ''字体Color

Dim intColorR = 0

Dim intColorG = 255

Dim intColorB = 120

myColor = System.Drawing.Color.FromArgb(RGB(intColorR, intColorG, intColorB))

'myColor = RGB(intColorR, intColorG, intColorB)

myFont = New System.Drawing.Font(myFontFamily, sngSize, myFontStyle, System.Drawing.GraphicsUnit.Point)

Label1.Font = myFont

Label1.ForeColor = myColor

# 数据类型

(2012-07-30 10:53:08)

[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)转载▼于锋学习娱乐网

|  |  |
| --- | --- |
| 标签：  [数据类型](http://search.sina.com.cn/?c=blog&q=%CA%FD%BE%DD%C0%E0%D0%CD&by=tag)  [变量](http://search.sina.com.cn/?c=blog&q=%B1%E4%C1%BF&by=tag)  [转换函数](http://search.sina.com.cn/?c=blog&q=%D7%AA%BB%BB%BA%AF%CA%FD&by=tag)  [vb](http://search.sina.com.cn/?c=blog&q=vb&by=tag)  [it](http://search.sina.com.cn/?c=blog&q=it&by=tag) | 分类： [VisualBasic.Net](http://blog.sina.com.cn/s/articlelist_2296297277_3_1.html) |

VB.NET基本数据类型是由系统定义的数据类型，包括数值数据类型、字节型、逻辑型、日期型、字符型、字符串类型和对象类型。

## 一、数值类型

数值数据类型分别是：Integer、Long、Single、Double和Decimal，它们的表数范围、所占字节数如表3－1所列。

表3－1  VB ．NET的数值数据类型

类型 数值范围 字节数

Integer －2147483648 ～ 2147483647 4

Long 9223372036854775808 ～ 9223372036854775807 8

Short －32768 ～ 32767 2

Single －3．402823E38 ～－1．401298E－451．041298E－45 ～ 3．402823E38 4

Double －1．7976313486231E308 ～ 4．94065645841247E－324

4．94065645841247E－324 ～ 1．79769313486231E308 8

Decimal －7922816251426433759354 ～ 79228162514264337593543950335 16

绝对值最小的非0数为：＋/－0．000000000000000000000000001 12

### 1、unsigned 和 signed 数值类型

    sign 指的是 +/- 正负符号，unsigned 数值类型即无正负符号的数值类型，也就是正值数值类型；signed 数值类型即带正负符号的数值类型，可能是正值也可能是负值，但其绝对值范围约为对应 unsigned 数值类型的一半。

如：SByte (signed Byte)占用1个字节，取值范围为 -128 到 127，而 Byte 取值范围为0到255。

    Short 占用2个字节，取值范围为 -32768 到 32767，而 UShort (unsigned Short) 取值范围为0到65535。

    Integer 占用4个字节，取值范围为 -2147483648 到 2147483647，而 UInteger (unsigned Integer) 取值范围为0到

4294967295。

    Long 占用8个字节，取值范围为 －9223372036854775808 到 9223372036854775808，而 ULong (unsigned Integer) 取值范围为0到 18446744073709551615。

### 2、非整数 数值类型

非整数类型包括Decimal（128位，16字节)定点数、Single（32位，4字节)单精度浮点数和 Double（64位，8字节)双精度浮点数。浮点数也就是实数。定点数取值不允许有舍入的误差，而浮点数允许有舍入的误差。可以用Rnd函数表示一个随意的>=0而且小于1的Single数，比如表示一个1-10之间的整数：var = Int(Rnd()\*10+1)

### 3、System

命名空间还提供了以位为标志的整型数值类型：Int16、Int32、Int64、UInt16、UInt32、UInt64。在VB6中，一个Long类型数值占用32位（4字节)，而在Visual Basic .NET 中一个Integer类型数值就占用32位,Long类型数值占用64位（8字节）。Byte 8位,1个字节;Short 16位,2个字节。计算机处理4字节数值更高效一些，所以我们一般常声明整数用Integer，声明小数用Single。

### 4、数值变量类型符

附加在变量名称后面，等于 As + 相应的类型名。如： Dim zhengshu% (= Dim zhengshu As Integer)

  % -- Integer, ! -- Single, # -- Double, & -- Long, @ -- Decimal

### 5、文字式数值类型符

附加在数字赋值后面。如：j = 12S (= Dim j As Short : j = 12)

S -- Short, US -- UShort, I -- Integer, UI -- UInteger, L -- Long, UL -- ULong, D -- Decimal, F -- Single（F=floating point）, R -- Double（R=real)。

另外在数字前加 &O 表示8进制数值，在数字前加 &H 表示16进制数值。可以用 Hex(数值）返回相应16进制表达字符，用 Oct(数值）返回相应8进制表达字符。

### 6、字节（Byte）数据类型

字节类型用以存储二进制数据，占1字节，其数值范围在0～255的整数，是一个无符号的数据类型。在进行ANSI与Unicode格式转换时，若数据是用Byte格式存储的，就可以保留其原值；若数据是用String格式存储的则可能会遭到破坏。在读文件、写文件、调用DIL、调用对象的方法和属性时，VB.NET都会自动在ANSI和Unicode之间进行转换。注意，除了一元减号外，所有对整数起作用的运算符都可用于Byte类型的数据。

## 二、字符类型

字符数据类型，用以处理可打印和显示的字符。字符数据类型包括Char类型和String类型，它们不能和数字类型数据直接进行转换。

### 1、 Char类型

Char类型（单个Unicode字符数据类型，也就是说这种变量只接受一个字符的数据）占用单个两字节（16位）Unicode字符(统一的字符编码标准,采用双字节对字符进行编码，中文就是Unicode字符，而英文则是非Unicode字符)。

字符型数据占2字节，取值范围0～65535，代表一个Unicode字符。Char数据类型与数值类型之间不允许隐式转换，但可使用系统的Asc（）或AscW（）将Char数据显示转换为数值数据。

### 2、 String类型

String类型（也称为字符串数据类型）为0个或多个双字节Unicode字符(最多有近20亿个),初始值为null。

String类型用于保存字符串数据，一个字符占1字节，一个汉字占2字节，字符串最大长度可达20亿（231）个Unicode字符。字符串前后要加上" "。例如："Good morning ！"、"程序"。

## 三、日期类型

Date 日期数据类型（64位）表示日期时间,范围从公元1年1月1日0：0：00到公元9999年12月31日11：59：59。注意：Date类型必须用“#”括起来。Date类型中可以只有日期，也可以只有时间，日期用/号分割，时间用:号分割。Date类型是以数字存储的，所以可以用DateDiff计算时间间隔，用DayOfYear方法计算该日期为一年中的第几天。我们可以用其ToOADate方法返回Date类型实例的等效OLE自动化日期（Double类型）。

Date类型用来保存日期和时间数据，占8字节，取值范围为1－1－0001 0：00：00～12－31－9999 23：59：59的日期时间，即日期范围为公元1年1月1日～9999年12月31日，时间范围为0：00：00～23：59：59。Date数据要求在日期时间值前后加上“#”，日期时间值的格式为m－d－yyyy hh ： mm ： ss（月－日－年时：分：秒）。例如：#12－3－2003 10：20：18#。

## 四、布尔类型

Boolean 布尔数据类型（32位）占用2个字节，只有两个值：真（True）和假（False），经常被用来表示逻辑判断的结果。

当将其他数据类型转换为逻辑数据时，非0转换为True，0转换为False。

## 五、对象类型

Object对象数据类型为一个32位地址（占4字节），相当于VB6中的Variant类型，它指向应用程序或其他应用程序中的一个对象。如果数据为Object类型，则它可以指向任何应用程序可处理的对象。它可以接收任何类型数据。当其中包含值类型时，Object类型数据被作为值类型处理；当其中包含引用类型时，Object类型数据被作为引用类型处理。

## 六、自定义结构类型

在VB6中用户自定义类型的关键字是Type,在vb.net中大部分用Structure,可以包含有一个或多个不同种类的数据类型，视自定义类型的成员来决定存放的数据，关键字为System．ValueType。尽管结构中的数据可以单独被访问，但是这些数据仍被认为是一个集合整体。

自定义结构类型的定义以Structure关键字开始，以End Structure关键字结束，结构中的元素可以是任意的数据类型的组合，包括其他结构。结构一旦定义出来后就可以被用作变量声明、参数传递以及函数的返回值等用途，下面是一个结构的语法：

如下例定义一个Employee结构：

Structure Employee

Public FirstName As String

Public FamilyName As String

Public Extension As Long

End Structure

用户自定义数据类型占用内存空间是其包含的所有数据类型所占用内存空间的总和。

## 七、数据类型转换

CBool,CByte,CSByte,CChar,CStr,CDate,CDble,CSng,CDec,CInt,CUInt,CShort,CUShort,CLng,CULng,

CObj

### 1.取整函数

（1）整型转换函数是将小数部分四舍五入掉，使被转换的数值转变为其最接近的整数值。注意：如果小数部分正好是0.5，CInt和CLng会四舍五入到最接近的偶数值。如：CInt(0.5)结果为0，CLng(4.5)结果为4，CInt(7.5)结果为8，CLng(14.7)结果为15。

（2）Fix和Int函数不四舍五入，直接删除小数部分，取得整数值部分。如：Fix(-9.9)结果为-9,Fix(-9.2)结果为-9,Fix(-0.9)结果为0。但Int函数在取负数小数的整数时，会取比当前数值小的整数部分，如：Int(-9.9)结果为-10，(-9.2)结果为-10。所以，当负数小数部分大于0.5时，CInt相当于Int。

（3）CInt返回值为Integer类型，CLng返回值为Long类型；而Fix和Int返回值与被转换的数值类型一致。

### 2.CType、DirectCast、TryCast(被转换数据，数据类型X）

三个函数均返回数据类型X，但DirectCast和TryCast被转换数据必须继承自或实现数据类型X，否则DirectCast会发生InvalidCastException错误，而TryCast会返回Nothing。例如：

Dim q As Object = 2.37  
Dim i As Integer = CType(q, Integer)  
' 以下转换会发生错误  
Dim j As Integer = DirectCast(q, Integer)  
Dim f As New System.Windows.Forms.Form  
Dim c As System.Windows.Forms.Control  
' 以下转换成功  
c = DirectCast(f, System.Windows.Forms.Control)

### 3.Parse和ToString方法

Parse是将字符数据转换为其他类型；ToString是将其他类型转换为字符数据。

基础数据类型（除String外）.Parse(字符数据）--〉 Parse函数将字符串数据转换为该基础数据类型值，此函数在对字符数据进行转型时比CType,CDate等要快。例：

Dim rq as Date

rq = Date.Parse("3-4-1999)

变量名.ToString() --〉将变量的值转换为文本。ToString可以带有一个格式化文本的字符参数，如：

Dim s As Double,str As String

s = 455454454

str = s.ToString("0.00")

str显示结果为455454454.00

### 4.Convert类

Convert类提供了许多方法，用来将一个值转换为另一种类型的值。但是在把数据转换为整型数值时，它使用的是.ToInt16 .ToInt32 .ToInt64 .ToUInt16 .ToUInt32 .ToUInt64方法，而不用 .ToInt。如：

Dim i As Integer = Convert.ToInt64("23")

Dim s As Boolen = Convert.ToBoolen(3)

## 八、判断变量数据类型

VarType（数据X)可以返回一个Integer类型的数值，该数值代表数据X的数据类型。所以判断一个数值的数据类型，可以通过VarType来判断，如：

Dim lx as Integer,dbl As Double，sz As Object

sz = 6.0

lx = VarType(v)

If lx = VarType(dbl) then

MessageBox.Show(sz.ToString & "是Double类型数值")

Else

MessageBox.Show(sz.ToString & "不是Double类型数值")

End If

# 字符字母

## 一、字符处理

string path=@"c:\test\a.txt";

string ChangeExtension=System.IO.Path.ChangeExtension(path,".old");//更改路径字符串的扩展名。

string CombinePath=System.IO.Path.Combine(@"c:\","b.txt");//合并两个路径字符串。

string DirectoryName=System.IO.Path.GetDirectoryName(path);//返回指定路径字符串的目录信息。

string Extension=System.IO.Path.GetExtension(path);//返回指定的路径字符串的扩展名。

string FileName=System.IO.Path.GetFileName(path);//返回指定路径字符串的文件名和扩展名。

string FileNameWithoutExtension=System.IO.Path.GetFileNameWithoutExtension(path); //返回不具有扩展名的指定路径字符串的文件名。

string FullPath=System.IO.Path.GetFullPath(path);//返回指定路径字符串的绝对路径。

string PathRoot=System.IO.Path.GetPathRoot(path);//获取指定路径的根目录信息。

string TempFileName=System.IO.Path.GetTempFileName();//返回唯一临时文件名并在磁盘上通过该名称创建零字节文件。

string TempPath=System.IO.Path.GetTempPath();//返回当前系统的临时文件夹的路径。

string HasExtension=System.IO.Path.HasExtension(path).ToString();//确定路径是否包括文件扩展名。

string IsPathRooted=System.IO.Path.IsPathRooted(path).ToString();//获取一个值，该值指示指定的路径字符串是包含绝对路径信息还是包含相对路径信息。

## 二、汉字首字母

xulanb的BLOG:http://blog.sina.com.cn/s/blog\_591f16290100gagq.html

获得汉字字符串拼音首字母 VB.net版(2010-01-18 15:54:31)

对于琤（AC62）无用。

'返回给定字符串的首字母

Private Function IndexCode(ByVal IndexTxt As String) As String

Dim i As Integer

For i = 1 To IndexTxt.Length

IndexCode = IndexCode & GetOneIndex(Mid(IndexTxt, i, 1))

Next

End Function

'得到单个字符的首字母

Private Function GetOneIndex(ByVal OneIndexTxt As String) As String

If Asc(OneIndexTxt) >= 0 And Asc(OneIndexTxt) < 256 Then

GetOneIndex = OneIndexTxt

Else

GetOneIndex = GetX(CInt(Format((Asc(OneIndexTxt) + 65536) \ 256 - 160, "00") & Format((Asc(OneIndexTxt) + 65536) Mod 256 - 160, "00")))

End If

End Function

'根据区位得到首字母

Private Function GetX(ByVal GBCode As Integer) As String

'判断一级汉字

If GBCode >= 1601 And GBCode < 1637 Then GetX = "A"

If GBCode >= 1637 And GBCode < 1833 Then GetX = "B"

If GBCode >= 1833 And GBCode < 2078 Then GetX = "C"

If GBCode >= 2078 And GBCode < 2274 Then GetX = "D"

If GBCode >= 2274 And GBCode < 2302 Then GetX = "E"

If GBCode >= 2302 And GBCode < 2433 Then GetX = "F"

If GBCode >= 2433 And GBCode < 2594 Then GetX = "G"

If GBCode >= 2594 And GBCode < 2787 Then GetX = "H"

If GBCode >= 2787 And GBCode < 3106 Then GetX = "J"

If GBCode >= 3106 And GBCode < 3212 Then GetX = "K"

If GBCode >= 3212 And GBCode < 3472 Then GetX = "L"

If GBCode >= 3472 And GBCode < 3635 Then GetX = "M"

If GBCode >= 3635 And GBCode < 3722 Then GetX = "N"

If GBCode >= 3722 And GBCode < 3730 Then GetX = "O"

If GBCode >= 3730 And GBCode < 3858 Then GetX = "P"

If GBCode >= 3858 And GBCode < 4027 Then GetX = "Q"

If GBCode >= 4027 And GBCode < 4086 Then GetX = "R"

If GBCode >= 4086 And GBCode < 4390 Then GetX = "S"

If GBCode >= 4390 And GBCode < 4558 Then GetX = "T"

If GBCode >= 4558 And GBCode < 4684 Then GetX = "W"

If GBCode >= 4684 And GBCode < 4925 Then GetX = "X"

If GBCode >= 4925 And GBCode < 5249 Then GetX = "Y"

If GBCode >= 5249 And GBCode <= 5589 Then GetX = "Z"

'判断二级汉字

If GBCode >= 5601 And GBCode <= 8794 Then

Dim CodeData As String

CodeData = "cjwgnspgcenegypbtwxzdxykygtpjnmjqmbsgzscyjsyyfpggbzgydywjkgaljswkbjqhyjwpdzlsgmrybywwccgznkydgttngjeyekzydcjnmcylqlypyqbqrpzslwbdgkjfyxjwcltbncxjjjjcxdtqsqzycdxxhgckbphffsspybgmxjbbyglbhlssmzmpjhsojnghdzcdklgjhsgqzhxqgkezzwymcscjnyetxadzpmdssmzjjqjyzcjjfwqjbdzbjgdnzcbwhgxhqkmwfbpbqdtjjzkqhylcgxfptyjyyzpsjlfchmqshgmmxsxjpkdcmbbqbefsjwhwwgckpylqbgldlcctnmaeddksjngkcsgxlhzaybdbtsdkdylhgymylcxpycjndqjwxqxfyyfjlejbzrwccqhqcsbzkymgplbmcrqcflnymyqmsqtrbcjthztqfrxchxmcjcjlxqgjmshzkbswxemdlckfsydsglycjjssjnqbjctyhbftdcyjdgwyghqfrxwckqkxebpdjpxjqsrmebwgjlbjslyysmdxlclqkxlhtjrjjmbjhxhwywcbhtrxxglhjhfbmgykldyxzpplggpmtcbbajjzyljtyanjgbjflqgdzyqcaxbkclecjsznslyzhlxlzcghbxzhznytdsbcjkdlzayffydlabbgqszkggldndnyskjshdlxxbcghxyggdjmmzngmmccgwzszxsjbznmlzdthcqydbdllscddnlkjyhjsycjlkohqasdhnhcsgaehdaashtcplcpqybsdmpjlpcjaqlcdhjjasprchngjnlhlyyqyhwzpnccgwwmzffjqqqqxxaclbhkdjxdgmmydjxzllsygxgkjrywzwyclzmcsjzldbndcfcxyhlschycjqppqagmnyxpfrkssbjlyxyjjglnscmhcwwmnzjjlhmhchsyppttxrycsxbyhcsmxjsxnbwgpxxtaybgajcxlypdccwqocwkccsbnhcpdyznbcyytyckskybsqkkytqqxfcwchcwkelcqbsqyjqcclmthsywhmktlkjlychwheqjhtjhppqpqscfymmcmgbmhglgsllysdllljpchmjhwljcyhzjxhdxjlhxrswlwzjcbxmhzqxsdzpmgfcsglsdymjshxpjxomyqknmyblrthbcftpmgyxlchlhlzylxgsssscclsldclepbhshxyyfhbmgdfycnjqwlqhjjcywjztejjdhfblqxtqkwhdchqxagtlxljxmsljhdzkzjecxjcjnmbbjcsfywkbjzghysdcpqyrsljpclpwxsdwejbjcbcnaytmgmbapclyqbclzxcbnmsggfnzjjbzsfqyndxhpcqkzczwalsbccjxpozgwkybsgxfcfcdkhjbstlqfsgdslqwzkxtmhsbgzhjcrglyjbpmljsxlcjqqhzmjczydjwbmjklddpmjegxyhylxhlqyqhkycwcjmyhxnatjhyccxzpcqlbzwwwtwbqcmlbmynjcccxbbsnzzljpljxyztzlgcldcklyrzzgqtgjhhgjljaxfgfjzslcfdqzlclgjdjcsnclljpjqdcclcjxmyzftsxgcgsbrzxjqqcczhgyjdjqqlzxjyldlbcyamcstylbdjbyregklzdzhldszchznwczcllwjqjjjkdgjcolbbzppglghtgzcygezmycnqcycyhbhgxkamtxyxnbskyzzgjzlqjdfcjxdygjqjjpmgwgjjjpkjsbgbmmcjssclpqpdxcdyykypcjddyygywchjrtgcnyqldkljczzgzccjgdyksgpzmdlcphnjafyzdjcnmwescsglbtzcgmsdllyxqsxsbljsbbsgghfjlwpmzjnlyywdqshzxtyywhmcyhywdbxbtlmswyyfsbjcbdxxlhjhfpsxzqhfzmqcztqcxzxrdkdjhnnyzqqfnqdmmgnydxmjgdhcdycbffallztdltfkmxqzdngeqdbdczjdxbzgsqqddjcmbkxffxmkdmcsychzcmljdjynhprsjmkmpcklgdbqtfzswtfgglyplljzhgjjgypzltcsmcnbtjbhfkdhbyzgkpbbymtdlsxsbnpdkleycjnycdykzddhqgsdzsctarlltkzlgecllkjljjaqnbdggghfjtzqjsecshalqfmmgjnlyjbbtmlycxdcjpldlpcqdhsycbzsckbzmsljflhrbjsnbrgjhxpdgdjybzgdlgcsezgxlblgyxtwmabchecmwyjyzlljjshlgndjlslygkdzpzxjyyzlpcxszfgwyydlyhcljscmbjhblyjlycblydpdqysxktbytdkdxjypcnrjmfdjgklccjbctbjddbblblcdqrppxjcglzcshltoljnmdddlngkaqakgjgyhheznmshrphqqjchgmfprxcjgdychghlyrzqlcngjnzsqdkqjymszswlcfqjqxgbggxmdjwlmcrnfkkfsyyljbmqammmycctbshcptxxzzsmphfshmclmldjfyqxsdyjdjjzzhqpdszglssjbckbxyqzjsgpsxjzqznqtbdkwxjkhhgflbcsmdldgdzdblzkycqnncsybzbfglzzxswmsccmqnjqsbdqsjtxxmbldxcclzshzcxrqjgjylxzfjphymzqqydfqjjlcznzjcdgzygcdxmzysctlkphtxhtlbjxjlxscdqccbbqjfqzfsltjbtkqbsxjjljchczdbzjdczjccprnlqcgpfczlclcxzdmxmphgsgzgszzqjxlwtjpfsyaslcjbtckwcwmytcsjjljcqlwzmalbxyfbpnlschtgjwejjxxglljstgshjqlzfkcgnndszfdeqfhbsaqdgylbxmmygszldydjmjjrgbjgkgdhgkblgkbdmbylxwcxyttybkmrjjzxqjbhlmhmjjzmqasldcyxyqdlqcafywyxqhz"

GetX = Mid(CodeData, (Microsoft.VisualBasic.Left(CStr(GBCode), 2) - 56) \* 94 + (Microsoft.VisualBasic.Right(CStr(GBCode), 2)), 1)

End If

End Function

# 日期时间

简介：DateTime和Date数据结构貌似是一样的。System.DateTime可不用实例化而直接使用。

## 一、当前时间

### 1、DateTime

Public Structure **Date**

**System** 的成员

**摘要:**表示时间上的一刻，通常以日期和当天的时间表示。

### 2、Now

Public Shared ReadOnly Property **Now** As **Date**

**System**.**Date** 的成员

**摘要:**获取一个 System.DateTime 对象，该对象设置为此计算机上的当前日期和时间，表示为本地时间。

**返回值:**

其值为当前日期和时间的对象。

### 3、Today

Public Shared ReadOnly Property **Today** As **Date**

**System**.**Date** 的成员

**摘要:**获取当前日期。

**返回值:**一个对象，设置为当天日期，其时间组成部分设置为 00:00:00。

### 4、TimeOfDay

Public ReadOnly Property **TimeOfDay** As **System**.**TimeSpan**

**System**.**Date** 的成员

**摘要:**获取此实例的当天的时间。

**返回值:**一个时间间隔，它表示当天自午夜以来已经过时间的部分。

### 5、实例

My.Settings.Day1 = Now.Date （获取日期Now换成Today一样）

My.Settings.Timer1 = Now.TimeOfDay （获取时间Now换成Today一样）

## 二、时间间隔

### 1、Sleep

Public Shared Sub **Sleep**(*millisecondsTimeout* As **Integer**)

**System**.**Threading**.**Thread** 的成员

**摘要:**

将当前线程挂起指定的时间。

**参数:**

*millisecondsTimeout*: 线程被阻塞的毫秒数。 指定零 (0) 以指示应挂起此线程以使其他等待线程能够执行。 指定 System.Threading.Timeout.Infinite 以无限期阻止线程。

**异常:**

*System.ArgumentOutOfRangeException*: 超时值为负且不等于 System.Threading.Timeout.Infinite。

### 2、Now.Subtract

Public Function **Subtract**(*value* As **Date**) As **System**.**TimeSpan**

**System**.**Date** 的成员

**摘要:**

从此实例中减去指定的日期和时间。

**参数:**

*value*: 要减去的日期和时间值。

**返回值:**

一个时间间隔，它等于此实例所表示的日期和时间减去 value 所表示的日期和时间。

**异常:**

*System.ArgumentOutOfRangeException*: 结果小于 System.DateTime.MinValue 或大于 System.DateTime.MaxValue。

### 3、实例

Dim start As Date = Now

System.Threading.Thread.Sleep(5000) '程序暂停5s（单位毫秒）

Debug.Print(Now.Subtract(start).TotalMilliseconds) ‘现在时间Now和原来时间start比较

# 键盘鼠标

## 一、模拟按键

### 1、键盘键与虚拟键码对照表

按键 值 按键 值 按键 值 按键 值

(小键盘 VK)

A 65 0 96 F1 112 Backspace 8

B 66 1 97 F2 113 Tab 9

C 67 2 98 F3 114 Clear 12

D 68 3 99 F4 115 Enter 13

E 69 4 100 F5 116 Shift 16

F 70 5 101 F6 117 Control 17

G 71 6 102 F7 118 Alt 18

H 72 7 103 F8 119 Caps Lock 20

I 73 8 104 F9 120 Esc 27

J 74 9 105 F10 121 Spacebar 32

K 75 “\*” 106 F11 122 Page Up 33

L 76 “+” 107 F12 123 Page Down 34

M 77 Enter 108 End 35

N 78 “-“ 109 Home 36

O 79 “.” 110 Left Arrow 37

P 80 “/” 111 Up Arrow 38

Q 81 Right Arrow 39

R 82 Down Arrow 40

S 83 Insert 45

T 84 Delete 46

U 85 Help 47

V 86 Num Lock 144

W 87

X 88

Y 89

Z 90

0 48

1 49

2 50

3 51

4 52

5 53

6 54

7 55

8 56

9 57

### 2、模拟按键

最简单System.Windows.Forms.SendKeys.Send(Chr(65)) 输入“A”

（详情参见SendKeys. Send 方法帮助）注：不引发按键事件（钩子不能捕捉）

其他可用API

## 二、模拟鼠标

# 窗体控件

## 窗体

### 程序在windows的位置设置

通过在程序关闭事件中将程序的最后位置保存到My.Settings中，下次打开时加载该位置

Private Sub Form1\_FormClosed(sender As Object, e As FormClosedEventArgs) Handles Me.FormClosed

My.Settings.LastLocation = Me.Location

My.Settings.Save()

End Sub

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

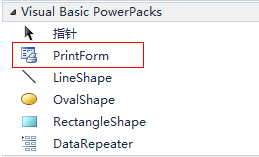
Me.Location = My.Settings.LastLocation

Debug.Print(Me.Location.X & "//" & Me.Location.Y)

End Sub

### 使用PrintForm打印窗体

在VS20\*\*中内置了了一个PowerPacks的扩展包,可以直接打印窗体内容。先从"工具箱"中拖一个PrintForm到设计好的窗体中（可以使用代码调用该类Dim shd As New Microsoft.VisualBasic.PowerPacks.Printing.PrintForm）

[](http://photo.blog.sina.com.cn/showpic.html#blogid=559541b40101iqap&url=http://album.sina.com.cn/pic/001zaET2ty6GycvgyTX85)

然后在窗体里加了个打印的按钮,对应代码如下

Dim printFormObj As New Microsoft.VisualBasic.PowerPacks.Printing.PrintForm ''打印窗体对象

Dim pSettings As New Printing.PrinterSettings

With pSettings ''打印设置

Dim myMag As New System.Drawing.Printing.Margins ''页边距设置信息是存放在这个Margins类型的对象中的

With myMag ''分别设置上下左右边距,单位貌似是毫米吧,带求证

myMag.Left = 1

myMag.Top = 1

myMag.Right = 1

myMag.Bottom = 1

End With

Dim myPaper As New System.Drawing.Printing.PaperSize ''设置纸张大小

For Each Item As System.Drawing.Printing.PaperSize In .PaperSizes

'Debug.Print(Item.PaperName)

If Item.PaperName.Equals("A4") = True Then

myPaper = Item

Exit For

End If

Next

Dim myPrinter As String = Nothing ''设置打印机

For Each myPrinter In Printing.PrinterSettings.InstalledPrinters

Debug.Print(myPrinter)

If myPrinter.Equals("\\10.33.12.92\pdfFactory") = True Then

Exit For

End If

Next

.DefaultPageSettings.Margins = myMag

.DefaultPageSettings.PaperSize = myPaper

.DefaultPageSettings.PrinterSettings.PrinterName = myPrinter

If .DefaultPageSettings.PrinterSettings.IsValid = True Then

''确定打印机是否有效

End If

End With

printFormObj.PrinterSettings = pSettings

printFormObj.Form = Me

Debug.Print(printFormObj.PrinterSettings.PaperSizes.Item(0).Kind) ''打印纸张代号

printFormObj.PrintAction = Printing.PrintAction.PrintToPreview ''设置打印预览

printFormObj.Print()

## 控件

### 动态增加控件

Public WithEvents monCal As System.Windows.Forms.MonthCalendar ''设置一个控件变量动态调用该控件

Private Sub DataGridViewMM\_CellClick(sender As Object, e As DataGridViewCellEventArgs) Handles DataGridView1.CellClick, DataGridView2.CellClick, DataGridView3.CellClick, DataGridView4.CellClick

If e.ColumnIndex = 4 OrElse e.ColumnIndex = 5 Then

dataViewMM = sender : colMM = e.ColumnIndex : rowMM = e.RowIndex

Select Case CInt(Microsoft.VisualBasic.Right(dataViewMM.Name, 1))

Case 1 : tabPageMM = Me.TabPage1

Case 2 : tabPageMM = Me.TabPage2

Case 3 : tabPageMM = Me.TabPage3

Case 4 : tabPageMM = Me.TabPage4

End Select

monCal = New System.Windows.Forms.MonthCalendar

Dim conTemp As System.Windows.Forms.Control

tabPageMM.Controls.Add(monCal)

conTemp = tabPageMM.Controls(tabPageMM.Controls.Count - 1)

tabPageMM.Controls.Remove(sender) ''更改控件的顺序

tabPageMM.Controls.Add(sender)

conTemp.Left = 0 : conTemp.Top = 0

monCal.Focus()

End If

End Sub

### 控件的父控件

Public dataViewMM As System.Windows.Forms.DataGridView

Public tabPageMM As System.Windows.Forms.TabPage

tabPageMM = dataViewMM.Controls.Owner

# 文件目录

**VB中也有相关的内容。**

## 一、使用VBScript

Dim objFs, objFolder, objFiles

Dim strDirSvr As String = "C:\Users\10605819\Desktop\Data"

objFs = Nothing

objFolder = Nothing

objFs = CreateObject("Scripting.FileSystemObject")

objFolder = objFs.GetFolder(strDirSvr)

objFiles = objFolder.Files

For Each FCnt In objFiles

Debug.Print(FCnt.Path)

Next FCnt

‘’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’

## 二、使用DirectoryInfo

Dim dCfiles As New System.IO.DirectoryInfo("D:\")

Dim fFile As System.IO.FileInfo

For Each fFile In dCfiles.GetFiles

Debug.Print(fFile.FullName)

Next

‘’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’

## 三、目录和文件

''创建目录、文件及文件写入

Private Sub DirectoryFile()

Dim path As String = "D:\temp"

If System.IO.Directory.Exists(path) = False Then

' Create a directory.

Directory.CreateDirectory(path)

End If

path = "D:\temp\MyTest.txt"

If File.Exists(path) = False Then

' Create a file to write to.

Using sw As StreamWriter = File.CreateText(path)

sw.WriteLine("Hello")

sw.WriteLine("And")

sw.WriteLine("Welcome")

sw.Flush()

End Using

End If

' Open the file to read from.

Using sr As StreamReader = File.OpenText(path)

Do While sr.Peek() >= 0

Console.WriteLine(sr.ReadLine())

Loop

End Using

End Sub

## 四、快捷方式

### 1.引用COM组件

调用Windows系统内部COM组件（Windows Script Host Object Model）（项目工程-引用）。

Vb.Net项目文件夹中需要Interop.IWshRuntimeLibrary.dll文件（添加引用自动复制）

''创建文件lnk（快捷方式）

Private Sub Btn\_lnk\_Click(sender As Object, e As EventArgs) Handles Btn\_lnk.Click

Dim wsh As New IWshRuntimeLibrary.IWshShell\_Class

Dim desktop As String = wsh.SpecialFolders.Item("Desktop") ''桌面路径

Dim theink As IWshRuntimeLibrary.IWshShortcut = wsh.CreateShortcut(desktop & "\Test01.lnk")

With theink

.Arguments = "/?" ''传递参数

.Description = "Test面板说明"

''.IconLocation="\*\*\*.ico" ’’’’图标

.TargetPath = "D:\02Program Files\VB.net\VB.Net学习"

.WindowStyle = 1

.WorkingDirectory = "D:\"

.Save()

End With

Debug.Print("已创建！！")

End Sub

### 2.CreateObject of WScript.Shell

''CreateObject of IWshRuntimeLibrary---lnk（快捷方式）（不需引用COM组件）

Private Sub IwshCreateObject(ByVal lnkName As String, ByVal theTargetPath As String, Optional ByVal theDescription As String = "描述说明")

Dim wsh As Object = CreateObject("WScript.Shell")

Dim desktop As String = wsh.SpecialFolders.Item("Desktop") ''桌面路径

Dim theink As Object = wsh.CreateShortcut(desktop & "\" & lnkName & ".lnk")

With theink

.Arguments = "/?" ''传递参数

.Description = theDescription

''.IconLocation="\*\*\*.ico" ''图标

.TargetPath = theTargetPath

.WindowStyle = 1

.WorkingDirectory = ""

.Save()

End With

Debug.Print("已创建！！")

End Sub

## 五、关联文件打开程序

将所用关联的文件右键选打开方式—指定以程序A方式打开。（可以使用代码将指定的文件类型和程序A相关联）

在程序A的加载窗体（或主程序）中加入以下代码：

Private Sub FileDoubleOpen\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Dim MyFile As String = "" '用以保存所打开的文件路径

Dim Parameters() As String '用以保存数组参数

Dim i As Integer

Parameters = System.Environment.GetCommandLineArgs()

i = Parameters.GetUpperBound(0)

If i > 0 Then

MessageBox.Show("数量：" & i, "Count", MessageBoxButtons.OK)

MyFile = Parameters(0) ''指定程序的路径

MessageBox.Show(MyFile, "Atten00", MessageBoxButtons.OK)

MyFile = Parameters(1) ''打开文件的路径

MessageBox.Show(MyFile, "Atten11", MessageBoxButtons.OK)

'PictureBox1.ImageLocation = MyFileName

End If

End Sub

# 读写操作

## 一、二进制读写

System.IO 名称空间下的各种类

### 1、读操作示例

'从文件指定位置读取数据

dim fn as new io.filestream("e:\123.mp3", IO.FileMode.Open)

dim fr as new io.binaryreader(fn)

fr.BaseStream.Position = 1000 '从1000字节处开始读取文件、即将文件流的指针移动到1000字节处，然后再根据自己的要求操作即可。

dim data() as byte= fr.ReadBytes(2000) '读取2000个字节至data()数组

fr.close

fn.close

# 类和对象

## 一、委托

### 1、自定义委托

Module ModuleDelegate

''声明委托——f(\*\*,\*\*) As \*\*

Public Delegate Function f(ByVal in1 As Integer, ByVal in2 As Integer) As Integer

''和声明委托相同签名的函数——f(\*\*,\*\*) As \*\*

Public Function AcceptDelegateSum(ByVal temp1 As Integer, ByVal temp2 As Integer) As Integer

Return temp1 + temp2

End Function

''和声明委托相同签名的函数——f(\*\*,\*\*) As \*\*

Public Function AcceptDelegateMax(ByVal temp1 As Integer, ByVal temp2 As Integer) As Integer

If temp1 > temp2 Then Return temp1 Else Return temp2

End Function

''使用委托数据类型作为参数

Public Function ParamaterDelegate(ByVal tempDelegate As f) As Integer

Return tempDelegate.Invoke(5, 26)

End Function

''使用委托

Public Sub UseDelegate()

''求最大值

Debug.Print(ParamaterDelegate(AddressOf AcceptDelegateMax))

''求和值

Debug.Print(ParamaterDelegate(AddressOf AcceptDelegateSum))

End Sub

End Module

### 2、系统委托

Action 委托

'声明 Public Delegate Sub Action

Func(Of TResult) 委托 '声明

Public Delegate Function Func(Of Out TResult) As TResult

例子：（使用了Lambda 表达式）

Private funTest As Func(Of Integer, String) = Function(nt As Integer) As String

Return "123"

End Function

Private baseMeth As Action(Of testBaseClass) = AddressOf testParam

Private deribeMeth As Action(Of testChildClass, Integer, Integer) = Sub(iin As testChildClass, ns As Integer, d As Integer)

iin.Print()

End Sub

Private Sub testParam(ByVal param As testBaseClass)

param.Print()

End Sub

## 二、获取活动程序对象

VB常用方法Interaction. GetObject 方法[Microsoft.VisualBasic]

### 1.Marshal. GetActiveObject 方法

从运行对象表 (ROT) 获取指定对象的运行实例。

**命名空间：**   System.Runtime.InteropServices   
**程序集：**mscorlib（在 mscorlib.dll 中）

### 2.AutomationFactory. CreateObject 方法 (String)

激活具有指定的编程标识符 (ProgID) 的已注册自动化服务器并返回对它的引用。

**命名空间：**   System.Runtime.InteropServices.Automation   
**程序集：**System.Windows（在 System.Windows.dll 中）

# 通信服务

(WCF/Windows)

## 一、WCF服务

面向服务的体系结构

## 二、Windows服务

## 三、通信

使用System.Net.Sockets.Socket类

# 句柄进程

## 一、程序句柄相关

### 1、本程序句柄

Me.Handle（只读属性原VB Me.hWnd）

## 二、获取系统句柄

### 1、输入焦点（光标位置）对象句柄获得

‘’申明API

Private Declare Function GetFocus Lib "user32" Alias "GetFocus" () As Integer

Private Declare Function AttachThreadInput Lib "user32" Alias "AttachThreadInput" (ByVal idAttach As Integer, ByVal idAttachTo As Integer, ByVal fAttach As Integer) As Integer

Private Declare Function GetCurrentThreadId Lib "kernel32" Alias "GetCurrentThreadId" () As Integer

Private Declare Function GetWindowThreadProcessId Lib "user32" Alias "GetWindowThreadProcessId" (ByVal hwnd As Integer, lpdwProcessId As Integer) As Integer

Private Declare Function GetForegroundWindow Lib "user32" Alias "GetForegroundWindow" () As Integer

‘’调用API

Dim hWndT As Integer

AttachThreadInput(GetCurrentThreadId, GetWindowThreadProcessId(GetForegroundWindow, 0), True)

hWndT = GetFocus()

Debug.Print("Hwnd:" & hWndT) ‘’hWndI即为光标位置句柄

## 三、进程ID

### 1、隐藏应用程序

''''任务管理器的应用程序窗口不显示，只能在进程中找

### 2、获取进程

Dim pro As System.Diagnostics.Process ''进程管理

Debug.Print(pro.ProcessName)

For Each pro In Process.GetProcesses

If pro.ProcessName = "EXCEL" Then

pro.WaitForExit()

pro.Kill()

End If

Next

# 编程钩子

VB编程中钩子的实现及应用-.NET教程,VB.Net语言

## 前言

windows系统中钩子具有相当强大的功能，通过这种技术可以对几乎所有的windows 系统中的消息进行拦截、监视、处理。这种技术可以广泛应用于各种软件，尤其是需要有监控、自动记录等对系统进行监测功能的软件。本文针对这个专题进行了探 讨，希望可以为读者朋友们起到抛砖引玉的作用。

## 一、钩子的机制及类型

windows的应用程序都是基于消息驱动的，应用程序的操作都依赖于它所得到的消息的类型及内容。钩子与dos中断截获处理机制有类似之 处。钩子(hook)是windows消息处理机制的一个平台，通过安装各种钩子，应用程序可以在上面设置子程序以监视指定窗口的某种消息，并且当消息到 达目标窗口之前处理它。

在windows中，钩子有两种，一种是系统钩子(remotehook)，它对消息的监视是整个系统范围，另一种是线程钩子 (localhook)，它的拦截范围只有进程内部的消息。对于系统钩子，其钩子函数（hookfunction）应在windows系统的动态链接库 (dll)中实现，而对于线程钩子来说,钩子函数可以在dll之中实现，也可以在相应的应用程序之中实现。这是因为当开发人员创建一个钩子 时，windows先在系统内存中创建一个数据结构，该数据结构包含了钩子的相关信息，然后把该结构体加到已经存在的钩子链表中去，并且新的钩子将排在老 的钩子的前面。当一个事件发生时，如果安装的是一个局部钩子，当前进程中的钩子函数将被调用。如果是一个远程钩子，系统就必须把钩子函数插入到其它进程的 地址空间，要做到这一点就要求钩子函数必须在一个动态链接库中，所以如果想要使用远程钩子，就必须把该钩子函数放到动态链接库中去。对于钩子所监视的消息 类型来说，windws一共提供了如下几种类型：如表1所示:

表一、windows消息类型

消息类型常量标识 //值 //消息类型 //适用范围

wh\_callwndproc //4 //发给窗口的消息 //线程或系统

wh\_callwndprocret //12 //窗口返回的消息 //线程或系统

wh\_cbt //5 //窗口变化、焦点设定等消息 //线程或系统

wh\_debug //9 //是否执行其它hook的hook //线程或系统

wh\_foregroundidle //11 //前台程序空闲 //线程或系统

wh\_getmessage //3 //投放至消息队列中的消息 //线程或系统

wh\_journalplayback //1 //将所记载的消息进行回放 //系统

wh\_journalrecord //0 //监视并记录输入消息 //系统

wh\_keyboard //2 //键盘消息 //线程或系统

wh\_mouse //7 //鼠标消息 //线程或系统

wh\_msgfilter //-1 //菜单滚动条、对话框消息 //线程或系统

wh\_shell //10 //外壳程序的消息 //线程或系统

wh\_sysmsgfilter //6 //所有线程的菜单滚动条、对话框消息 //系统

## 二、vb编程中钩子的实现

### 1钩子函数(hook function)的格式

hook function实际上是一个函数，如果是系统钩子，该函数必须放在动态链接库中。该函数有一定的参数格式，在vb中如下:

private function hookfunc(byval ncode as long，byval wparam as long，byval lparam as long)as long

其中，ncode代表是什么情况之下所产生的钩子，随钩子的不同而有不同组的可能值；参数wparam，lparam传回值包括了所监视到的消息内容，它随hook所监视消息的种类和ncode的值不同而不同。对于用vb所设置的钩子函数,一般的框架形式如下:

private function hookfunc(byval ncode as long，byval wparam as long，byval lparam as long)as long

select case of ncode

case ncode<0:hookfunc=callnexthookex(hhookfunc，ncode，wparam，lparam)

case值1:处理过程1:hookfunc=x1

case2:处理过程2:hookfunc=x1

……

end select

end function

函数的传回值，如果消息要被处理，则传0，否则传1，吃掉消息。

### 2钩子的安装及执行

钩子的安装要用到几个api函数:可以使用api函数setwindowshookex()把一个应用程序定义的钩子子程安装到钩子链表中。setwindowshookex()函数的声明如下:

declare function setwindowshookex lib "user32" alias "setwindowshookexa"(byval idhook as long，byval lpfn as long，byval hmod as long，byval dwthreadid as long)as long

idhook值为它处理的消息类型；lpfn值为钩子子程序的地址指针。如果dwthreadid参数为0或是一个由别的进程创建的线程的标 识，lpfn必须指向dll中的钩子子程。除此以外，lpfn可以指向当前进程的一段钩子子程代码。hmod值为应用程序的句柄，标识包含lpfn所指的 子程的dll。如果dwthreadid标识当前进程创建的一个线程，而且子程代码位于当前进程，hmod必须为0。dwthreadid值为与安装的钩 子子程相关联的线程的标识符，如果为0，钩子子程与所有的线程关联。钩子安装成功则返回钩子子程的句柄，失败返回0。

另外，一般应在钩子子程中调用callnexthookex()函数以执行钩子链表所指的下一个钩子子程，否则安装了别的钩子的应用程序就会收不到钩子通知，从而产生错误的结果。callnexthookex()函数的声明如下:

declare function callnexthookex lib"user32" alias "callnexthookex"(byval hhook as long，byval ncode as lonog， byval wparam as long，lparam as any)as long

hhook值是setwindowshookex()的传回值，ncode、wparam、lparam则是hook函数中的三个参数。在程 序终止之前，必须调用unhookwindowshookex()函数释放与钩子关联的系统资源。unhookwindowsex()函数声明如下:

declare function unhook windowshookex lib "user32" alias "unhook windowshookex(byval hhook as long)as long

hhook为安装钩子时的返回值，即钩子子程的句柄。

### 3钩子安装应注意的问题

lpfn参数是一个hookfunc的地址,vb规定必须将hookfunc代码放到标准的.bas模块 中，并以"address of hookfunc"传入，而不可以将其放到类模块中，也不能将其附加到窗体上。而对于remotehook来说，hookfunc应包含在动态链接库中， 因此如果在vb中使用remotehook，则还要用到getmodulehandle()、getprocaddress()两个api函数,它们的声 明如下:

declare function getmodulehandle lib"kernel32" alias "getmodulehandlea"（byval lpmodulename as string）as long

declare function getprocaddress lib "kernel32" alias "getprocaddress"（byval hmodule as long，byval lpprocname as string）as long

hmod值是含钩子过程的模块名柄，如果是localhook，该值可以是null(vb中传0)，而如果是remotehook，则可以使用getmodulehandle("名称.dll")来传入。

## 三、实例--键盘消息的拦截

在程序开发时常用的有对输入消息进行监视的键盘钩子，对于所监视到的消息应进行处理，下面对键盘钩子参数的具体内容组成进行说明:

如果有键盘消息(wm\_keyup或wm\_keydown)将被处理时，则系统调用键盘钩子。

ncode为hc\_action或hc\_noremove，若小于0，则要求处理函数向下传递该消息。

wparam表示按键键码常数，ａ键到ｚ键与其ascii码的相应值ａ到ｚ是一致的，例如按c键，则wparam值为67。

lparam与wm\_keydown同，占四个字节，其包括的内容较多,其二进制结构如下:

0 1 …… 14 15|16……22 23|24…… 28 29 30 31

0-15位(key repeat count)，键码重复次数。16-23位(scan code)，按键的扫描码。24位(extended\_key flag)，扩展键(功能键、数字小键盘上的键)标志，为1则是扩展键,否则为0。25-28位被保留。29位(context code)，状态描述码，alt键被按下则为1，否则为0。30位(previouskey\_stateflag)指定先前的键状态，如果消息被发出之前 键处于按下状态，则为1；键处于释放状态则为0。31位(transiton\_stateflag)状态转换标志，如果键是被按下值为1，如果键被放开值 为0。

本例中的钩子用来监视并记录应用程序中的按键信息。在程序中，alt+f4组合键被屏蔽。下面是部分代码：

Public hhook as long

private sub form\_load()′程序启动时安装钩子

hhook=setwindowshookex(2，address of mykbhook，0，app.threadid)

End sub

′具体的钩子程序，本例中该过程被包含在module1中

public function mykbhook(byval ncode as long，byval wparam as long，byval lparam as long)as long

if ncode>=0 then

open "c:\keyfile.txt" for append as #1 将键盘的操作记录在keyfile.txt文件之中

记录所操作的键、操作时间、日期操作时的按键状态，用16进制记录

write #1，wparam，hex(lparam)，date，time

close #1

mykbhook=0 表示要处理这个消息

屏蔽ａｌｔ+ｆ4组合键

if wparam=115 and(lparam and&h20000000)<>0 then

if(lparam and &hc000000)=0 then 是否进行alt+f4操作

myhbhook=1 钩子吃掉这个消息

end if

end if

end if

call callnexthookex(hhook，ncode，wparam，lparam)将消息传给下一个钩子

end function

程序退出时卸载钩子

private sub form\_unload(cancel as interger)

call unhook windowshookex(hhook)

end sub

## 四、总结

钩子处理程序是windows高级编程技术，一般程序员都使用vc++等程序设计工具实现，本文表明，对于vb来说，虽然很多人认为是非专业 的设计工具，但实现钩子这样的高级技术也是非常方便的。另外在使用钩子时应注意到，钩子虽然功能比较强,但如果使用不当将会严重影响系统的效率，所以要尽 量避免使用系统钩子,并且在不用钩子时，应将钩子及时卸载。

文章整理：西部数码--专业提供域名注册、虚拟主机服务

http://www.west263.com

以上信息与文章正文是不可分割的一部分,如果您要转载本文章,请保留以上信息，谢谢!

# 数据系统

## 一、Access数据库

### 1 ADO.Net对象

以下这些变量在本章的例子中通用。

imports System.data.oledb /\*引用命名空间\*/

Public adoConn As System.Data.OleDb.OleDbConnection

Public adoComm As System.Data.OleDb.OleDbCommand

Public adoRst As System.Data.OleDb.OleDbDataAdapter

Public adoset As System.Data.DataSet

Public sConnectString As String

以下示例数据库名称与字段

数据库名称：form\_info

数据库字段：form\_no,product\_no,drawing\_no,designer,date1,num1,num2,num3。

注：

1、VB.net 中将Date作为关键字，数据库中如将Date作为一个字段时，在修改Date字段则会产生语法错误。

2、数据库中重点内容是查询数据库方法。

### 2连接数据库

#Region "连接数据库"

Private Sub Button2\_Click(sender As Object, e As EventArgs) Handles Button2.Click

Dim sqlstr As String

sConnectString = "provider=microsoft.jet.oledb.4.0,data source=Access数据库文件位置 " '/\*定义连接字符串\*/

adoConn = New System.Data.OleDb.OleDbConnection(cnstr) '/\*定义连接\*/

adoConn.Open() '/\*打开连接\*/

sqlstr = "select... from... 或其他sql"

adoComm = New System.Data.OleDb.OleDbCommand(sqlstr, adoConn)

'/\*以上是建立执行语句\*/

adoComm.ExecuteNonQuery() '不返回结果

adoComm.ExecuteScalar() '返回单个结果

adoComm.ExecuteReader() '返回结果集

'/\*以上是执行sql语句的方法\*/

adoConn.Close() '/\*关闭连接\*/

End Sub

#End Region

你要做用户验证的话，先根据用户名查询对应的密码，再判断用户输入的密码和查询结果是不是相同就可以了用cmd.executescalar() 这个方法

### 3删除数据

#Region "删除记录"

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

Dim coMstr As String = "delete form\_no,product\_no,drawing\_no,designer,date from form\_info where form\_no like '%" & "设05-10-1" & "%'"

adoConn.Open()

adoComm = New System.Data.OleDb.OleDbCommand(coMstr, adoConn)

adoRst.DeleteCommand = adoComm

adoRst.DeleteCommand.ExecuteNonQuery()

End Sub

#End Region

### 4修改数据

#Region "修改数据"

Private Sub Button2\_Click(sender As Object, e As EventArgs) Handles Button2.Click

adoConn.Open()

Dim strsql As String

strsql = "select \* from form\_info where form\_no = '设05-10-2'" '"

Dim adoComm As New System.Data.OleDb.OleDbCommand(strsql, adoConn)

Dim dr As System.Data.OleDb.OleDbDataReader

dr = adoComm.ExecuteReader

If dr.Read Then

dr.Close()

Dim coMstr As String = "UPDATE form\_info SET form\_no = '456' where form\_no = '设05-10-2'"

adoComm = New System.Data.OleDb.OleDbCommand(coMstr, adoConn)

adoRst.UpdateCommand = adoComm

adoRst.UpdateCommand.ExecuteNonQuery() '老是这里提示错误 不明白之前要用 dr.Close()

Else

dr.Close()

Debug.Print("无数据")

End If

End Sub

#End Region

### 5增加数据

#Region "增加记录"

Private Sub Button3\_Click(sender As Object, e As EventArgs) Handles Button3.Click

Dim coMstr As String = "INSERT INTO form\_info(form\_no,num1,num2,num3) VALUES('032',5,8,8)"

adoConn.Open()

adoComm = New System.Data.OleDb.OleDbCommand(coMstr, adoConn)

adoRst.InsertCommand = adoComm

adoRst.InsertCommand.ExecuteNonQuery()

End Sub

#End Region

# 并行编程

.Net Framework 4引入了新的任务并行库(taskPrallelLibrary,TPL)。TPL的诞生源于多核处理器。 TPL最重要的名称空间是System.Threading.Tasks，其中最主要的类是Task类、Parallel静态类。

Parallel静态类的提供方法：

System.Threading.Tasks.Parallel.For()

System.Threading.Tasks.Parallel.ForEach()

System.Threading.Tasks.Parallel.Invoke()

# 外部调用

调用外部程序，传入参数。

### Shell 方法

打开记事本：

Shell "notepad.exe d:\111.txt"

打开图片

Shell "rundll32.exe c:\WINDOWS\System32\shimgvw.dll,ImageView\_Fullscreen " & "C:\Users\Public\Pictures\Sample Pictures\Hydrangeas.jpg"

### Process类

System.Diagnostics. Process