Советы тем, кто программирует на Visual Basic

Андрей Колесов
Ольга Павлова

Совет 182. Реализация функции «ожидания» в VB

Здесь мы покажем, как можно реализовать функцию ожидания в VB. Вначале поместите на форму таймер (Timer1) и установите его свойство Interval= 0 и свойство Enabled = False.

Для тестирования процедуры добавьте две метки (Label1 и Label2) и командную кнопку (Command1) к форме. Напишите следующую подпрограмму и код события Timer для таймера:

```
Public Sub Wait(seconds)
    ` включение таймера
    Timer1.Enabled = True
    ` установка интервала для таймера
    Me.Timer1.Interval = 1000 * seconds
    While Me.Timer1.Interval > 0
        DoEvents
    Wend
    ` выключение таймера
    Timer1.Enabled = False
End Sub

Private Sub Timer1_Timer()
    Timer1.Interval = 0
End Sub
```

Теперь можете использовать функцию Wait везде, где требуется какая-либо задержка, например:

```
Private Sub Command1_Click()
    Label1.Caption = Now
    Wait (5)
    Label2.Caption = Now
End Sub
```

Совет 183. Поддержка нумерации версий ваших VB-программ

Нумерация версий программ, созданных на VB, может быть простой, если при создании EXE-модуля использовать функцию Version Numbering. Для этого щелкните кнопку Options диалогового окна Make Project, а затем установите флажок Auto Increment во вкладке Make диалогового окна Project Properties.

Номер версии программы состоит из трех элементов: Major, Minor и Revision. Функция Auto Increment, если она выделена, будет автоматически увеличивать номер Revision на единицу при каждом запуске команды Make Project для конкретного проекта.

Обычно информация о версиях программы используется в форме About. Для этого просто добавьте метку с именем lblVersion и введите следующий код для вашей формы:

```
lblVersion.Caption = "Версия: " & App.Major & _
    "." & App.Minor & "." & App.Revision
```

Если для вашей программы номер Major равен 2, номер Minor — 1 и номер Revision — 12, то метка выведет на экран: “Версия: 2.1.12”.

Совет 184. Создание собственного хранителя экрана

У вас никогда не было желания создать свой собственный хранитель экрана в VB? Если да, то сейчас мы приведем элементарный пример, как это можно сделать. Идея очень проста — в качестве заставки будет выдаваться форма размером с весь экран, на которую вы можете по собственному желанию нанести любые изображения.

Для начала создайте новый проект Standard EXE. Поместите на форму элемент управления Label, содержащий какой-либо текст. Затем добавьте туда элемент управления Timer, для которого установите свойство Interval как 1000 (то есть 1 с). Теперь введите следующий код для вашей формы:

```
Private Sub Form_Click()
    ' хранитель экрана выгружается, если щелкнуть форму
    Unload Me
End Sub

Private Sub Form_Load()
    ' не разрешается загружать несколько
    ' экземпляров хранителя экрана
    If App.PrevInstance Then Unload Me
End Sub

Private Sub Timer1_Timer()
    ' мигание метки каждую секунду
    Label1.Visible = Not (Label1.Visible)
End Sub
```

После этого установите свойство WindowState для формы как Maximized, а свойство Border Style как None. Большинство хранителей экрана занимают полный размер экрана и не имеют строки заголовка.

Выберите команду File|Make EXE File и в появившемся диалоговом окне Make Project щелкните кнопку Options. В диалоговом окне Project Properties в текстовом поле Application Title введите прописными буквами строку SCRNSAVE:. (Например, мы можем назвать

наше приложение SCRNSAVE:TestApp1.) При задании имени исполняемого файла не забудьте поменять расширение: оно должно быть .SCR вместо .EXE. (Для нашего примера назовите исполняемый файл как TestApp1.scr.) Щелкните OK.

Вот и все. Теперь не забудьте поместить SCR-файл в каталог \Windows\System и поменяйте хранитель экрана, как обычно, с помощью Control Panel.

Совет 185. Используйте WithEvents для добавления новых функций к элементу управления

Когда-нибудь вы можете столкнуться с тем, что у стандартных элементов управления отсутствуют какие-либо полезные функции. Для этого можно использовать традиционный способ: написать нужный код для соответствующего события каждого элемента управления. Однако в VB5 и VB6 имеется команда WithEvents, которая предоставляет простое решение с использованием классов.

Предположим, вы хотите вводить в текстовое окно только прописные буквы с тем, чтобы все вводимые строчные буквы автоматически преобразовывались в прописные. Кроме того, вам необходимо, чтобы при попадании курсора мыши внутрь текстового окна на экране появлялась всплывающая подсказка ToolTipText, содержащая координаты курсора внутри данного текстового окна.

Создайте новый проект Standard EXE, разместите на форме четыре элемента управления TextBox с именами Text1, Text2, Text3, Text4 и добавьте модуль класса. Введите следующий код для формы Form1:

```
' общие объявления
Private clsTextBox1 As Class1
Private clsTextBox2 As Class1
Private clsTextBox3 As Class1
Private clsTextBox4 As Class1

Private Sub Form_Load()
    Set clsTextBox1 = New Class1
    Set clsTextBox1.TextBoxCtl = Text1
    Set clsTextBox2 = New Class1
    Set clsTextBox2.TextBoxCtl = Text2
    Set clsTextBox3 = New Class1
    Set clsTextBox3.TextBoxCtl = Text3
    Set clsTextBox4 = New Class1
    Set clsTextBox4.TextBoxCtl = Text4
End Sub

Private Sub Text4_KeyPress(KeyAscii As Integer)
    ' этот код выполняется перед событием KeyPress
    ' для класса Class1
    If KeyAscii = Asc("a") Then
      KeyAscii = Asc("z"): Beep
    End If
End Sub
```

Для класса Class1 введите такой код:

```
Private WithEvents txt As TextBox

Public Property Set TextBoxCtl(OutsideTextBox_
    As TextBox)
    Set txt = OutsideTextBox
End Property

Private Sub txt_KeyPress(KeyAscii As Integer)
    ' преобразует в прописные буквы
    KeyAscii = Asc(UCase$(Chr$(KeyAscii)))
End Sub

Private Sub txt_MouseMove(Button As Integer,
    Shift As Integer, X As Single, Y As Single)
    txt.ToolTipText = "X:" & X & " Y:" & Y
End Sub
```

Используя такую программную конструкцию, вы добавите новые функциональные возможности для всех четырех текстовых полей. Но обратите внимание, что событие KeyPress для элемента управления выполняется раньше, чем происходит обращение к классу. Поэтому в поле TextBox4 вместо «a» будет вводиться «z», которая сразу же преобразуется в «Z».

Совет 186. Изменение набора изображений в элементе управления ImageList, связанном с элементом управления Toolbar

В режиме разработки проекта вам может пригодиться возможность свободно добавлять изображения в элемент управления ImageList, связанный с элементом управления Toolbar, или удалять их оттуда. И поскольку VB не позволяет изменять набор изображений в ImageList, пока он связан с панелью инструментов, мы покажем вам способ, как обойти данное ограничение.

Шаг 1. Заполнение элемента управления ImageList. Поместите элемент управления ImageList на форму. (Если данный компонент не входит в комплект инструментальных средств вашего проекта, то его можно добавить так, как показано в Совете 187.) Щелкните по нему правой кнопкой мыши, а затем выберите команду Properties для открытия диалогового окна Property Pages. Выберите вкладку Images и щелкните кнопку Insert Picture. В диалоговом окне Select Picture найдите изображение, которое хотите добавить в элемент управления ImageList. Присвойте ему уникальное свойство Key. Повторите эти операции, пока не заполните элемент управления ImageList так, как вам хочется.

Шаг 2. Добавление кнопок к панели инструментов. Щелкните правой кнопкой мыши элемент управления Toolbar и затем выберите команду Properties. В раскрывшемся диалоговом окне Property Pages выберите вкладку Buttons. Щелкните кнопку Insert Button и в текстовом поле Key введите уникальное имя, при-

своенное изображению в элементе управления ImageList. Каждая кнопка с изображением должна иметь то же свойство Key, что и соответствующее изображение в компоненте ImageList. Каждая кнопка без изображения, например tbrSeparator или tbrPlaceholder, не должна иметь свойства Key.

Шаг 3. В событии Load для формы установите связь элементов управления ImageList и Toolbar:

```
Set ToolBar1.ImageList = ImageList1
```

Шаг 4. Присвойте изображения кнопкам на панели инструментов:

```
Dim myButton as Variant

For Each myButton in ToolBar1.Buttons
    If myButton.Key <> Empty Then
        myButton.Image = myButton.Key
        ' если значение свойство Key имеет
        ' какой-либо смысл, используйте его
        ' для описания и текста подсказки
        myButton.Description = myButton.Key
        myButton.ToolTipText = myButton.Key
    End If
Next
```

Совет 187. Загрузка элементов управления ActiveX

Для использования элементов управления ActiveX, поставляемых с VB 5.0/6.0, необходимо добавить их к комплекту инструментальных средств Toolbox.

Шаг 1. В меню Project выберите команду Components или щелкните правой кнопкой мыши панель инструментов для вызова диалогового окна Components.

Шаг 2. Элементы, приведенные в этом диалоговом окне, включают все зарегистрированные встроенные объекты, проектировщики и элементы управления ActiveX.

Шаг 3. Установите флажок, находящийся слева от имени элемента управления, который вы хотите добавить.

Шаг 4. Щелкните OK для закрытия диалогового окна Components. Теперь все выбранные элементы управления ActiveX появятся в комплекте инструментальных средств Toolbox.

Такая процедура очень проста, если вы знаете точное имя добавляемого элемента управления. Проблема возникает тогда, когда элемент, приведенный в диалоговом окне Components, содержит несколько компонентов или его название отличается от имени компонента. Здесь вам поможет приведенная ниже таблица с перечнем элементов управления ActiveX, поставляемых с VB 5.0/6.0:

Элемент диалогового окна Components	Имя компонента
Microsoft Windows Common Controls	TabStrip
	Toolbar
	StatusBar
	ProgressBar
	TreeView
	ListView
	ImageList
	Slider
	ImageCombo
Microsoft Windows Common Controls-2	Animation
	UpDown
	MonthView
	DTPicker
	FlatScrollBar
Microsoft Windows Common Controls-3	CoolBar
Microsoft MAPI Controls	MAPISession
	MAPIMessages
Microsoft Data Bound List Controls	DBList
	DBCombo
Microsoft DataList Controls	DataList
	DataCombo
Microsoft Direct Animation Media Controls	PathControl
	StructuredGraphicsControl
	SpriteControl
	SequencerControl
Microsoft Internet Controls	WebBrowser
	ShellFolderViewOC

Подробнее о новых и усовершенствованных элементах управления, поставляемых вместе VB 6.0, см. КомпьютерПресс № 1'99, стр. 154.

Совет 188. Использование типа Date с источником данных ADO

Первое, что приходит в голову при работе с датами в VB, — использовать переменную типа Date. Однако в действительности такой подход оказывается неверным, если вы имеете дело с датами Null, получаемыми из источника данных ADO. Причина заключается в том, что поведение внутреннего типа данных Date отличается от типа данных adDate ADO.

Чтобы увидеть эту разницу между Date и adDate, внимательно изучите следующий код (для VB6 не забудьте установить ссылку Microsoft ActiveX Data Objects Recordset 2.0 Library, но то же самое верно и для Microsoft ActiveX Data Objects 2.0 Library):

```
Private Sub Date_handling()
    Dim lDate As Date
    Dim lDateVar As Variant
    Dim lRs As ADOR.Recordset
    '
    ' инициализация
    Set lRs = New ADOR.Recordset
    lRs.Fields.Append "MyDate", adDate, ,
    adFldIsNullable
    lRs.Open
    lRs.AddNew
    lRs!MyDate = Date
    MsgBox lRs!MyDate
    '
    ' хранение даты
    lDate = lRs!MyDate
    lDateVar = lRs!MyDate
    '
    ' установка даты
    'lDate = Null    ' этот оператор не работает
    lDateVar = Null ' этот оператор будет работать
    '
    lRs!MyDate = lDateVar
    'lRs!Update
    lRs.Close
End Sub
```

Переменная типа String — тоже не лучший вариант, поскольку строковая переменная Null не является представлением даты Null. К сожалению, согласно установке, используемая по умолчанию в DataEnvironment в VB6, при перемещении поля Date с помощью метода drag-and-drop на форму помещается элемент управления TextBox.

Таким образом, чтобы получить правильное внутреннее представление даты, следует использовать переменную типа Variant.

Совет 189. Правильная обработка обратной косой черты при использовании свойства App.Path

Свойство App.Path может использоваться для получения пути к текущему исполняемому файлу приложения. Будьте, однако, осторожны, так как при этом возможен небольшой «ляп». Если приложение выполняется в корневом каталоге, то в конце пути добавляется обратная косая черта. Однако если приложение выполняется в каком-либо другом каталоге, то обратной косой черты на конце не окажется. Использование следующей функции поможет решить данную проблему:

```
Public Function AppPath(sFileName As String) As String
    If Right$(App.Path, 1) = "\" Then
        AppPath = App.Path & sFileName
    Else
        AppPath = App.Path & "\" & sFileName
    End If
End Function
```

Например, AppPath("test.txt") будет правильно добавлять имя файла независимо от того, в каком каталоге находится приложение.

Совет 190. Экспорт содержимого элемента управления Grid в текстовый файл

Здесь приводится подпрограмма, которая используется для экспорта содержимого элементов управления MSGrid или MSFlexGrid в ASCII-файл неограниченного размера. В качестве разделителя вы можете задавать любой нравящийся вам символ. Кроме того, у вас есть возможность указывать символ, в который будет заключаться содержимое каждой ячейки. Используя, например, следующий вызов подпрограммы, вы заключите содержимое ячеек в двойные кавычки:

```
GridExport(grdEmployees, "c:\test.cvs", ",", Chr$(34))
```

А так выглядит сама подпрограмма:

```
Public Sub GridExport(GridToExport As Object, _
    FileName As String, Optional Delimiter As Variant, _
    Optional EncloseStrings As Variant)
    ' GridToExport -- имя экспортируемого элемента
    ' управления MSGrid  или MSFlexGrid;
    ' FileName -- полное имя файла, куда экспортируется
    ' содержимое сетки;
    ' Delimiter -- символ-разделитель (необязательный
    ' параметр, по умолчанию используется Tab);
    ' EncloseStrings -- символ, в который заключается
    ' содержимое каждой ячейки (необязательный
    ' параметр, по умолчанию ничего не используется)
    Dim iNumRows As Integer
    Dim iNumCols As Integer
    Dim iFileNumber As Integer
    '
    If IsMissing(Delimiter) Then
      Delimiter = vbTab
    End If
    If IsMissing(EncloseStrings) Then
      EncloseStrings = ""
    End If
    iFileNumber = FreeFile
    Open FileName For Output As #iFileNumber
      For iNumRows = 0 To GridToExport.Rows - 1
        GridToExport.Row = iNumRows
        For iNumCols = 0 To GridToExport.Cols - 1
          GridToExport.Col = iNumCols
          ' если это не первый столбец, то
          ' перед значением ставится разделитель
          If iNumCols > 0 Then
            Print #iFileNumber, Delimiter;
          End If
          Print #iFileNumber, EncloseStrings & _
                GridToExport.Text & EncloseStrings;
        Next iNumCols
        Print #iFileNumber, ""
      Next iNumRows
    Close #iFileNumber
End Sub
```