События

Последнее обновление: 30.10.2015

Хотя наши делегаты прекрасно выполняют возложенную на них функцию, но в языке VB.NET для этих же целей предусмотрены более простые и удобные конструкции под названием **события**. Итак, продолжим работу с классом `Client` из прошлой главы и изменим его следующим образом:

```
Public Class Client
    Inherits Person
    Implements IAccount

    'Объявляем делегат
    Public Delegate Sub AccountStateHandler(message As String)
    'Событие, возникающее при выводе денег
    Public Event Withdrowed As AccountStateHandler
    'Событие возникающее при добавление на счет
    Public Event Added(sum As Integer)

    'Переменная для хранения суммы
    Dim _sum As Integer
    'Переменная для хранения процента
    Dim _procentage As Integer

    Public Property Bank As String

    'Текущая сумма на счете
    ReadOnly Property CurentSum() As Integer Implements IAccount.CurentSum
        Get
            Return _sum
        End Get
    End Property
    'Метод для добавления денег на счет
    Sub Put(sum As Integer) Implements IAccount.Put
        _sum += sum
        RaiseEvent Added(sum)
    End Sub
    'Метод для снятия денег со счета
    Sub Withdraw(sum As Integer) Implements IAccount.Withdraw
```

```
        If sum <= CurentSum Then
            _sum -= sum
            RaiseEvent Withdrowed("Сумма " & sum & " снята со счета")
        Else
            RaiseEvent Withdrowed("Недостаточно денег на счете")
        End If
    End Sub
    'Процент начислений
    ReadOnly Property Procentage() As Integer Implements IAccount.Procentage
        Get
            Return _procentage
        End Get
    End Property

    Public Overrides Sub Display()
        Console.WriteLine(FirstName & " " & LastName & " has an account in b
    End Sub

    Public Sub New(fName As String, lName As String, _bank As String, _sum A
        MyBase.New(fName, lName)
        Bank = _bank
        Me._sum = _sum
    End Sub

End Class
```

Здесь мы сделали несколько изменений по сравнению с предыдущей версией класса. Во-первых, мы убрали переменную делегата и методы регистрации и отмены регистрации метода делегата. Во-вторых. мы добавили два события. События объявляются с помощью ключевого слова **Event**. Поскольку первое событие **Withdrowed** объявлено как экземпляр делегата **AccountStateHandler**, то для его обработки потребуется метод, принимающий строку в качестве параметра. Для второго события **Added** делегат не задан, поэтому мы указываем параметры в объявлении события. Поэтому метод, обрабатывающий данное событие, должен будет принимать в качестве параметра значение типа Integer. Вместо вызова делегатов мы устанавливаем вызовы событий с помощью ключевого слова **RaiseEvent**, передавая в события значения для параметров.

Теперь обработаем события в основной программе:

```
Sub Main()

    Dim client1 As New Client("John", "Thompson", "City Bank", 200)

    'Добавляем обработчики события
    AddHandler client1.Withdrowed, New Client.AccountStateHandler(AddressOf
    AddHandler client1.Added, AddressOf Show_Message

```

```
    client1.Withdraw(100)
    'Удаляем обработчик события
    RemoveHandler client1.Withdrowed, AddressOf Color_Message

    client1.Withdraw(50)
    client1.Put(150)

    Console.ReadLine()
End Sub

Private Sub Show_Message(sum As Integer)
    Console.WriteLine("На счет поступило {0} $", sum)
End Sub

Private Sub Color_Message(message As String)
    'Устанавливаем красный цвет символов
    Console.ForegroundColor = ConsoleColor.Red
    Console.WriteLine(message)
    'Сбрасываем настройки цвета
    Console.ResetColor()
End Sub
```

Добавление обработчиков события происходит с помощью ключевого слова **AddHandler**, после которого идет имя события, которое будет обрабатываться. Далее мы можем указать делегат, либо использовать сокращенную запись добавления обработчика события. После ключевого слова **AddressOf** мы указываем метод, который и будет обрабатывать событие. Удаление обработчика события происходит подобным образом с помощью ключевого слова **RemoveHandler**. После удаления обработчика данное событие не будет обрабатываться методом `Color_Message`.

Существует еще один способ обработки события - с помощью ключевого слова **Handles**:

```
Module Module1

    Dim WithEvents client2 As New Client("Gomer", "Simpson", "City Bank", 20

    Private Sub Color_Message2(message As String) Handles client2.Withdrowed
        'Устанавливаем красный цвет символов
        Console.ForegroundColor = ConsoleColor.Red
        Console.WriteLine(message)
        'Сбрасываем настройки цвета
        Console.ResetColor()
    End Sub
End Module
```

В этом случае мы объявляем переменную, у которой будем обрабатывать события, с помощью ключевого слова **WithEvent**, которое идет сразу после модификатора доступа. При чем так объявить переменную мы можем только на уровне модуля или класса, но не

на уровне процедуры или функции. После объявления с помощью **WithEvent** мы можем обрабатывать событие, написав в объявлении метода после параметров ключевое слово **Handles**, после которого следует событие объекта. Обратите внимание. что такое объявление обработчика события доступно только для одиночного объекта, то есть мы не можем объявить целый массив следующим образом:

```
Dim WithEvents clients(4) As Client
```

В этом случае нам придется добавлять обработчики через **AddHandler**.

Говоря о событиях, нельзя не затронуть еще одну тему. Как правило, одним из наиболее распространенных вопросов новичков, состоит в том, а что такое параметр `e` в обработчике кнопке (`Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles Button1.Click`), да и в других событиях. Ответ на этот вопрос, как правило, следующий: `e` является объектом класса `EventArgs`, который содержит все данные события. Добавим и в нашу программу подобный класс:

```
Public Class AccountEventArgs
    Inherits EventArgs
    'Сообщение
    Public ReadOnly message As String
    'Сумма, на которую изменился счет
    Public ReadOnly sum As Integer
    Sub New(_message As String, _sum As Integer)
        message = _message
        sum = _sum
    End Sub
End Class
```

Здесь все просто - класс наследуется от класса **EventArgs**. Этот класс имеет два поля: одно для выводимого сообщения, и другое для хранения величины, на которую изменился счет. С учетом этого класса изменим класс `Client`:

```
Public Class Client
    Inherits Person
    Implements IAccount

    'Объявляем делегат
    Public Delegate Sub AccountStateHandler(sender As Object, e As AccountEv
    'Событие, возникающее при выводе денег
    Public Event Withdrowed As AccountStateHandler
    'Событие возникающее при добавление на счет
    Public Event Added(sender As Object, e As AccountEventArgs)

    'Переменная для хранения суммы
    Dim _sum As Integer
    'Переменная для хранения процента
    Dim _procentage As Integer
```

```
16
17      Public Property Bank As String
18
19      'Текущая сумма на счете
20      ReadOnly Property CurentSum() As Integer Implements IAccount.CurentSum
21          Get
22              Return _sum
23          End Get
24      End Property
25      'Метод для добавления денег на счет
26      Sub Put(sum As Integer) Implements IAccount.Put
27          _sum += sum
28          RaiseEvent Added(Me, New AccountEventArgs("Деньги добавлены на счет"
29      End Sub
30      'Метод для снятия денег со счета
31      Sub Withdraw(sum As Integer) Implements IAccount.Withdraw
32          If sum <= CurentSum Then
33              _sum -= sum
34              RaiseEvent Withdrowed(Me, New AccountEventArgs("Деньги сняты со
35          Else
36              RaiseEvent Withdrowed(Me, New AccountEventArgs("Недостаточно ден
37          End If
38      End Sub
39      'Процент начислений
40      ReadOnly Property Procentage() As Integer Implements IAccount.Procentage
41          Get
42              Return _procentage
43          End Get
44      End Property
45
46      Public Overrides Sub Display()
47          Console.WriteLine(FirstName & " " & LastName & " has an account in b
48      End Sub
49
50      Public Sub New(fName As String, lName As String, _bank As String, _sum A
51          MyBase.New(fName, lName)
52          Bank = _bank
53          Me._sum = _sum
54      End Sub
55
56  End Class
```

Вместо прежних значений теперь события будут в качестве параметров принимать объект, вызвавший событие, и объект **AccountEventArgs**, хранящий информацию о событии. Поскольку событие вызываем сам объект Client, то в качестве параметра `sender` мы передаем ключевое слово **Me**, указывающее на текущий объект. Теперь изменим основную программу:

```
1   Sub Main()
```

```
2
3       Dim client1 As New Client("John", "Thompson", "City Bank", 200)
4
5       AddHandler client1.Withdrowed, New Client.AccountStateHandler(AddressOf
6       AddHandler client1.Added, AddressOf Show_Message
7
8       client1.Withdraw(100)
9       client1.Withdraw(150)
10      client1.Put(150)
11
12      Console.ReadLine()
13  End Sub
14
15  Private Sub Show_Message(sender As Object, e As AccountEventArgs)
16      Console.WriteLine(e.message)
17      Console.WriteLine("На счет добавлено {0} $", e.sum)
18  End Sub
19
20  Private Sub Color_Message(sender As Object, e As AccountEventArgs)
21      'Устанавливаем красный цвет символов
22      Console.ForegroundColor = ConsoleColor.Red
23      Console.WriteLine("Была проведена попытка снять со счета {0} $", e.sum)
24      Console.WriteLine(e.message)
25      'Сбрасываем настройки цвета
26      Console.ResetColor()
27  End Sub
```

Теперь если мы запустим программу. мы получим следующий вывод:

```
Была проведена попытка снять со счета 100 $
Деньги сняты со счета
Была проведена попытка снять со счета 150 $
Недостаточно денег на счете. Операция недействительна
Деньги добавлены на счет
На счет добавлено 150 $
```

Назад **Содержание** **Вперед**

Контакты для связи: metanit22@mail.ru