



Делегаты

Последнее обновление: 30.10.2015



Наряду со свойствами и методами классы и интерфейсы могут иметь делегаты и события. Делегаты представляют такие объекты, которые указывают на другие методы. При этом делегаты и методы, на которые ссылаются делегаты, должны иметь те же параметры и тот же тип возвращаемого значения. Создадим два делегата:

```
1 Delegate Function Operation(x As Integer, y As Integer) As Integer
2
3 Delegate Sub GetMessage()
```

Первый делегат у нас ссылается на функцию, которая в качестве параметров принимает два значения типа Integer и возвращает некоторое число. Второй делегат у нас ссылается на процедуру без параметров. Чтобы использовать делегат, нам надо создать его объект с помощью конструктора, в который мы передаем адрес метода, вызываемого делегатом. Чтобы вызвать делегат, надо использовать его метод `Invoke`. Кроме того, делегаты могут выполняться в асинхронном режиме, при этом нам не надо создавать второй поток, нам надо лишь вместо метода `Invoke` использовать пару методов **`BeginInvoke/EndInvoke`**.

```
1 Public Delegate Sub GetMessage()
2
3 Sub Main()
4
5     Dim del As GetMessage
6     If Date.Now.Hour < 12 Then
7         del = New GetMessage(AddressOf GoodMorning)
8     Else
9         del = New GetMessage(AddressOf GoodEvening)
10    End If
11    del.Invoke()
12
13    Console.ReadLine()
14 End Sub
15
16 Sub GoodMorning()
```

```
17 Console.WriteLine("Good Morning")
18 End Sub
19
20 Sub GoodEvening()
21 Console.WriteLine("Good Evening")
22 End Sub
```

В данном случае мы в зависимости от времени передаем в делегат адрес определенного метода (с помощью ключевого слова **AddressOf** и выводим сообщение.

Теперь посмотрим на примере другого делегата:

```
1 Delegate Function Operation(x As Integer, y As Integer) As Integer
2
3 Sub Main()
4
5     Dim op As New Operation(AddressOf Add)
6     Console.WriteLine(op.Invoke(4, 5))
7
8     Console.ReadLine()
9 End Sub
10
11 Function Add(x As Integer, y As Integer) As Integer
12     Return x + y
13 End Function
14
15 Function Multiply(x As Integer, y As Integer) As Integer
16     Return x * y
17 End Function
```

Так как второй делегат ссылается на функции с двумя параметрами, то при вызове делегата мы должны передать в метод Invoke два значения.

Как и любой объект, делегат можно использовать в качестве параметра метода:

```
1 Public Delegate Sub GetMessage()
2
3 Sub Main()
4
5     If Date.Now.Hour < 12 Then
6         Show_Message(AddressOf GoodMorning)
7     Else
8         Show_Message(AddressOf GoodEvening)
9     End If
10
11     Console.ReadLine()
12 End Sub
13
14 Private Sub Show_Message(_del As GetMessage)
15     _del.Invoke()
```

```
16 End Sub
17
18 Sub GoodMorning()
19     Console.WriteLine("Good Morning")
20 End Sub
21
22 Sub GoodEvening()
23     Console.WriteLine("Good Evening")
24 End Sub
```

Однако, эти примеры не могут показать всю мощь делегатов, так как мы вполне спокойно могли обойтись и без них, вызвав напрямую методы. А наиболее сильная сторона делегатов состоит в том, что они служат в качестве методов обратного вызова, уведомляя другие объекты о произошедших событиях. Итак, вернемся к нашим классам, описывающим клиента банка, которые мы разработали в предыдущих главах (в данном случае классы `Employee` и `Manager` опущены, так как они нам не понадобятся):

```
1 Public MustInherit Class Person
2
3     Public Property FirstName() As String
4     Public Property LastName() As String
5     Public MustOverride Sub Display()
6
7     Public Sub New(fName As String, lName As String)
8         FirstName = fName
9         LastName = lName
10    End Sub
11
12 End Class
13
14 Public Class Client
15     Inherits Person
16     Implements IAccount
17
18     'Переменная для хранения суммы
19     Dim _sum As Integer
20     'Переменная для хранения процента
21     Dim _procentage As Integer
22
23     Public Property Bank As String
24
25     'Текущая сумма на счете
26     ReadOnly Property CurentSum() As Integer Implements IAccount.CurentSum
27         Get
28             Return _sum
29         End Get
30     End Property
31     'Метод для добавления денег на счет
32     Sub Put(sum As Integer) Implements IAccount.Put
```

```

33         _sum += sum
34     End Sub
35     'Метод для снятия денег со счета
36     Sub Withdraw(sum As Integer) Implements IAccount.Withdraw
37         If sum <= CurentSum Then
38             _sum -= sum
39         End If
40     End Sub
41     'Процент начислений
42     ReadOnly Property Procentage() As Integer Implements IAccount.Procentage
43     Get
44         Return _procentage
45     End Get
46 End Property
47
48 Public Overrides Sub Display()
49     Console.WriteLine(FirstName & " " & LastName & " has an account in b
50 End Sub
51
52 Public Sub New(fName As String, lName As String, _bank As String, _sum As Integer)
53     MyBase.New(fName, lName)
54     Bank = _bank
55     Me._sum = _sum
56 End Sub
57
58 End Class
59
60 Public Interface IAccount
61     ReadOnly Property CurentSum() As Integer
62     Sub Put(sum As Integer)
63     Sub Withdraw(sum As Integer)
64     ReadOnly Property Procentage() As Integer
65 End Interface

```

Допустим, в случае вывода денег с помощью метода `Withdraw` нам надо как-то уведомлять об этом самого клиента и, может быть, другие объекты. Для этого создадим делегат `AccountStateHandler`. Чтобы использовать делегат, нам надо создать переменную этого делегата, а затем присвоить ему метод, который будет вызываться делегатом. Итак, добавим в класс `Client` следующие строки:

```

1 Public Class Client
2     Inherits Person
3     Implements IAccount
4
5     'Объявляем делегат
6     Public Delegate Sub AccountStateHandler(message As String)
7     'Создаем переменную делегата
8     Dim del As AccountStateHandler
9

```

```

10     'Регистрируем делегат
11     Public Sub RegisterHandler(_del As AccountStateHandler)
12         del = _del
13     End Sub
14
15     'Здесь остальной код

```

Здесь все понятно. Сначала создаем делегат, который будет указывать на метод с параметром message типа String. Затем создаем переменную делегата. И в конце создаем метод, в котором будет происходить присваивание делегату ссылки на метод. Теперь изменим метод **Withdraw** следующим образом:

```

1 Sub Withdraw(sum As Integer) Implements IAccount.Withdraw
2     If sum <= CurentSum Then
3         _sum -= sum
4         If del IsNot Nothing Then
5             del("Сумма " & sum & " снята со счета")
6         End If
7     Else
8         If del IsNot Nothing Then
9             del("Недостаточно денег на счете")
10        End If
11    End If
12 End Sub

```

Теперь в главной программе протестируем работу делегата:

```

1 Module Module1
2
3     Sub Main()
4         'Создаем нового клиента
5         Dim client1 As New Client("John", "Thompson", "City Bank", 200)
6         'Добавляем в делегат ссылку на метод Show_Message
7         client1.RegisterHandler(New Client.AccountStateHandler(AddressOf Show_Message))
8         'Два раза подряд пытаемся снять деньги
9         client1.Withdraw(100)
10        client1.Withdraw(150)
11
12        Console.ReadLine()
13    End Sub
14
15    Private Sub Show_Message(message As String)
16        Console.WriteLine(message)
17    End Sub
18
19 End Module

```

Запустив программу, мы получим два разных сообщения, которые мы передали в коде класса Client:

Сумма 150 снята со счета
Недостаточно денег на счете

Хотя в примере наш делегат принимал адрес на один метод, в действительности он может указывать сразу на несколько методов. Кроме того, при необходимости мы можем удалить ссылки на адреса определенных методов, чтобы они не вызывались при вызове делегата. Итак, изменим в классе `Client` метод **RegisterHandler** и добавим новый метод **UnregisterHandler**, который будет удалять методы из списка методов делегата:

```

1 Public Sub RegisterHandler(_del As AccountStateHandler)
2     Dim mainDel As [Delegate] = System.Delegate.Combine(_del, del)
3     del = CType(mainDel, AccountStateHandler)
4 End Sub
5
6 Public Sub UnregisterHandler(_del As AccountStateHandler)
7     Dim mainDel As [Delegate] = System.Delegate.Remove(del, _del)
8     del = CType(mainDel, AccountStateHandler)
9 End Sub

```

В первом методе метод **Combine** объединяет делегаты `_del` и `del` в один, который потом присваивается переменной `del`. Во втором методе метод **Remove** возвращает делегат, из списка вызовов которого удален делегат `_del`. Теперь перейдем к основной программе:

```

1 Sub Main()
2
3     Dim client1 As New Client("John", "Thompson", "City Bank", 200)
4
5     client1.RegisterHandler(New Client.AccountStateHandler(AddressOf Show_Me
6     client1.RegisterHandler(New Client.AccountStateHandler(AddressOf Color_M
7     client1.Withdraw(100)
8     client1.Withdraw(150)
9     'Удаляем делегат
10    client1.UnregisterHandler(New Client.AccountStateHandler(AddressOf Color
11    client1.Withdraw(50)
12
13    Console.ReadLine()
14 End Sub
15
16 Private Sub Show_Message(message As String)
17     Console.WriteLine(message)
18 End Sub
19
20 Private Sub Color_Message(message As String)
21     'Устанавливаем красный цвет символов
22     Console.ForegroundColor = ConsoleColor.Red
23     Console.WriteLine(message)

```

```
24 'Сбрасываем настройки цвета
25 Console.ResetColor()
26 End Sub
```

В целях тестирования мы создали еще один метод - `Color_Message`, который выводит то же самое сообщение только красным цветом. В строке `client1.UnregisterHandler(New Client.AccountStateHandler(AddressOf Color_Message))` мы удаляем этот метод из списка вызовов делегата, поэтому этот метод больше не будет срабатывать. Консольный вывод будет иметь следующую форму:

```
Сумма 150 снята со счета
Сумма 150 снята со счета
Недостаточно денег на счете
Недостаточно денег на счете
Сумма 50 снята со счета
```

[Назад](#) [Содержание](#) [Вперед](#)



ТАКЖЕ НА METANIT.COM

Подключение к SQLite

5 месяцев назад · 1 коммен...

Библиотека sqlite3, подключение к базе данных SQLite в ...

Параметры строки запроса

5 месяцев назад · 1 коммен...

Параметры строки запроса query string в приложении Blazor на ...

Взаимодействие с кодом Python

5 месяцев назад · 4 коммен...

Взаимодействие с кодом Python в программе на языке Си, установка ...

Отправка серверу

5 меся...

Отправка серверу помощи

1 Комментарий

1 Войти ▾

Присоединиться к обсуждению...

ВОЙТИ С ПОМОЩЬЮ

ИЛИ ЧЕРЕЗ DISQUS ?

Имя



Поделиться

Лучшие Новые Старые



Валентин Коптев



6 лет назад

Почему при создании проекта на vb 2010
System.Windows.Controls.FlowDocumentScrollView
вызывает исключение?

Помощь сайту

YooMoney:
410011174743222

Qiwi:
qiwi.com/n/METANIT

Перевод на карту
Номер карты:
4048415020898850

[Вконтакте](#) | [Телеграм](#) | [Twitter](#) | [Помощь сайту](#)

Контакты для связи: metanit22@mail.ru

Copyright © metanit.com, 2023. Все права защищены.