Обобщенные типы

Последнее обновление: 30.10.2015

Отличительной особенностью .NET 2.0 от первой версии платформы стала поддержка обобщенных типов (generics), равно как и обобщенных методов и делегатов. Чтобы разобраться в особенности данного нововведения, сначала посмотрим на проблему, которая могла возникнуть до появления обобщенных типов. Посмотрим на примере:

```
Dim x As Integer = 44
Dim s As String = "hello"
Dim ar As New ArrayList()
'Упаковка значения x в тип Object
ar.Add(x)
ar.Add(s)
'Распаковка в значение типа Integer первого элемента коллекции
Dim y As Integer = ar(0)

Console.WriteLine(y)
```

В данном примере мы используем класс **ArrayList**, который представляет коллекцию объектов. Чтобы поместить объект в коллекцию, мы используем метод **Add**, а чтобы получить, указываем индекс элемента в скобках - `ar(0)`. Этот класс содержит коллекцию значений типа Object, а это значит, что в вызовах `ar.Add(x)` и `ar.Add(s)` значения переменных `x` и `s` сначала будут "упакованы" в значения типа Object, потом при получении элементов из коллекции - наоборот, "распакованы" в нужный тип. Упаковка и распаковка (boxing и unboxing) ведут к снижению производительности, так как системе надо осуществить необходимые преобразования.

Кроме того, существует другая проблема - проблема безопасности типов. Если мы напишем так, то естественно получим ошибку во время выполнения программы:

```
Dim y As Integer = ar(1) 'Вторым элементом является строка, а не число
```

Эти проблемы были призваны устранить обобщенные типы. Обобщенные типы позволяют указать конкретный тип, который будет использоваться. Например, используем обобщенный вариант класса `ArrayList` - класс `List`:

```
Dim x As Integer = 44
Dim s As String = "hello"
Dim arGen As New List(Of Integer)()
arGen.Add(x)
arGen.Add(s)
```

Так как класс List является обобщенным, то нам нужно задать в выражении **(Of тип)** тип данных, для которого этот класс будет применяться. Далее мы добавляем число и строку в коллекцию. Однако если число будет добавлено в коллекцию arrGen, то на строке `arGen.Add(s)` мы получим ошибку во время компиляции и должны будем удалить эту строку. Таким образом, используя обобщенный вариант класса, мы снижаем время на выполнение и количество потенциальных ошибок.

Создадим обобщенный класс **Transaction**:

```
Public Class Transaction(Of T)
    Dim inAccount As T
    Dim outAccount As T

    'Перевод с одного счета на другой определенный суммы денег
    Sub DoTransaction(sum As Integer)

    End Sub
    Public Sub New(_in As T, _out As T)
        inAccount = _in
        outAccount = _out
    End Sub

End Class
```

С помощью буквы **T** в описании `Public Class Transaction(Of T)` мы указываем, что данный тип будет использоваться этим классом. В классе мы создаем две переменные этого типа, которым присваиваем значения в конструкторе. Причем сейчас нам неизвестно, что это будет за тип. Однако мы предполагаем, что вместо параметра T мы будем использовать интерфейс счета `IAccount`, который мы создали в предыдущих главах. Однако мы не можем знать, какой вид счета в банке в данном случае будет использоваться. Поэтому мы можем установить ограничение в виде типа `IAccount`:

```
Public Class Transaction(Of T As IAccount)
    Dim inAccount As T
    Dim outAccount As T

    'Перевод с одного счета на другой определенный суммы денег
    Sub DoTransaction(sum As Integer)
        'Вычитаем с одного счета
        inAccount.Withdraw(sum)
        'Прибавляем к другому
```

```
        outAccount.Put(sum)
    End Sub
    Public Sub New(_in As T, _out As T)
        inAccount = _in
        outAccount = _out
    End Sub

End Class
```

При этом мы можем задать множество ограничений. В этом случае они перечисляются после ключевого слова **As** в фигурных скобках:

```
Public Class Transaction(Of T As {IAccount, Client})
```

В качестве ограничения могут выступать как конкретные классы, так и интерфейсы. Кроме того, можно указать ограничение, чтобы использовались только структуры:

```
Public Class Transaction(Of T As Structure)
```

или классы:

```
Public Class Transaction(Of T As Class)
```

А также можно задать в качестве ограничения класс или структуру, которые реализуют конструктор по умолчанию с помощью слова **New**:

```
Public Class Transaction(Of T As {Class, New})
```

Классами и структурами использование обобщений не ограничивается. Мы можем создавать также и обобщенные методы:

```
Sub GetInformation(Of T As Person)(emp As T)
    emp.Display()
End Sub
```

А затем также их использовать:

```
Sub Main()

    Dim emp As New Employee("John", "Thompson", "City Bank")

    GetInformation(Of Employee)(emp)

    Console.ReadLine()
End Sub
```

Назад **Содержание** **Вперед**