Chapter 10: Enum

Section 10.1: GetNames()

Returns the names of constants in the specified Enum as a string array:

```
Module Module1

  Enum Size
    Small
    Medium
    Large
  End Enum

  Sub Main()
    Dim sizes = [Enum].GetNames(GetType(Size))

    For Each size In sizes
      Console.WriteLine(size)
      Next
  End Sub

End Module
```

Output:

> Small
>
> Medium
>
> Large

Section 10.2: HasFlag()

The `HasFlag()` method can be used to check if a flag is set.

```
Module Module1

    <Flags>
    Enum Material
        Wood = 1
        Plastic = 2
        Metal = 4
        Stone = 8
    End Enum

    Sub Main()
        Dim houseMaterials As Material = Material.Wood Or Material.Stone

        If houseMaterials.HasFlag(Material.Stone) Then
            Console.WriteLine("the house is made of stone")
        Else
            Console.WriteLine("the house is not made of stone")
        End If
    End Sub
```

```
End Module
```

For more information about the Flags-attribute and how it should be used see the official Microsoft documentation.

Section 10.3: Enum definition

An enum is a set of logically related constants.

```
Enum Size
    Small
    Medium
    Large
End Enum

Public Sub Order(shirtSize As Size)
    Select Case shirtSize
        Case Size.Small
            ' ...
        Case Size.Medium
            ' ...
        Case Size.Large
            ' ...
    End Select
End Sub
```

Section 10.4: Member initialization

Each of the enum members may be initialized with a value. If a value is not specified for a member, by default it's initialized to 0 (if it's the first member in the member list) or to a value greater by 1 than the value of the preceding member.

```
Module Module1

    Enum Size
        Small
        Medium = 3
        Large
    End Enum

    Sub Main()
        Console.WriteLine(Size.Small)    ' prints 0
        Console.WriteLine(Size.Medium)   ' prints 3
        Console.WriteLine(Size.Large)    ' prints 4

        ' Waits until user presses any key
        Console.ReadKey()
    End Sub

End Module
```

Section 10.5: The Flags attribute

With the **<Flags>** attribute, the enum becomes a set of flags. This attribute enables assigning multiple values to an enum variable. The members of a flags enum should be initialized with powers of 2 (1, 2, 4, 8...).

```
Module Module1

    <Flags>
```

```
    Enum Material
        Wood = 1
        Plastic = 2
        Metal = 4
        Stone = 8
    End Enum

    Sub Main()
        Dim houseMaterials As Material = Material.Wood Or Material.Stone
        Dim carMaterials as Material = Material.Plastic Or Material.Metal
        Dim knifeMaterials as Material = Material.Metal

        Console.WriteLine(houseMaterials.ToString()) 'Prints "Wood, Stone"
        Console.WriteLine(CType(carMaterials, Integer)) 'Prints 6
    End Sub

End Module
```

Section 10.6: GetValues()

' This method is useful for iterating Enum values '

```
Enum Animal
    Dog = 1
    Cat = 2
    Frog = 4
End Enum

Dim Animals = [Enum].GetValues(GetType(Animal))

For Each animal in Animals
    Console.WriteLine(animal)
Next
```

Prints:

> 1
>
> 2
>
> 4

Section 10.7: String parsing

An Enum instance can be created by parsing a string representation of the Enum.

```
Module Module1

    Enum Size
        Small
        Medium
        Large
    End Enum

    Sub Main()
        Dim shirtSize As Size =  DirectCast([Enum].Parse(GetType(Size), "Medium"), Size)
```

```
        ' Prints 'Medium'
        Console.WriteLine(shirtSize.ToString())

        ' Waits until user presses any key
        Console.ReadKey()
    End Sub

End Module
```

See also: Parse a string to an Enum value in VB.NET

Section 10.8: ToString()

The `ToString` method on an enum returns the string name of the enumeration. For instance:

```
Module Module1
    Enum Size
        Small
        Medium
        Large
    End Enum

    Sub Main()
        Dim shirtSize As Size = Size.Medium
        Dim output As String = shirtSize.ToString()
        Console.WriteLine(output)  ' Writes "Medium"
    End Sub
End Module
```

If, however, the string representation of the actual integer value of the enum is desired, you can cast the enum to an `Integer` and then call `ToString`:

```
Dim shirtSize As Size = Size.Medium
Dim output As String = CInt(shirtSize).ToString()
Console.WriteLine(output)  ' Writes "1"
```

Section 10.9: Determine whether a Enum has FlagsAttribute specified or not

The next example can be used to determine whether a enumeration has the **FlagsAttribute** specified. The methodology used is based on **Reflection**

This example will give a **`True`** result:

```
Dim enu As [Enum] = New FileAttributes()
Dim hasFlags As Boolean = enu.GetType().GetCustomAttributes(GetType(FlagsAttribute),
inherit:=False).Any()
Console.WriteLine("{0} Enum has FlagsAttribute?: {1}", enu.GetType().Name, hasFlags)
```

This example will give a **`False`** result:

```
Dim enu As [Enum] = New ConsoleColor()
Dim hasFlags As Boolean = enu.GetType().GetCustomAttributes(GetType(FlagsAttribute),
inherit:=False).Any()
Console.WriteLine("{0} Enum has FlagsAttribute?: {1}", enu.GetType().Name, hasFlags)
```

We can design a generic usage extension method like this one:

```
<DebuggerStepThrough>
<Extension>
<EditorBrowsable(EditorBrowsableState.Always)>
Public Function HasFlagsAttribute(ByVal sender As [Enum]) As Boolean
    Return sender.GetType().GetCustomAttributes(GetType(FlagsAttribute), inherit:=False).Any()
End Function
```

Usage Example:

```
Dim result As Boolean = (New FileAttributes).HasFlagsAttribute()
```

Section 10.10: For-each flag (flag iteration)

In some very specific scenarios we would feel the need to perform a specific action for each flag of the source enumeration.

We can write a simple *Generic* extension method to realize this task.

```
<DebuggerStepThrough>
<Extension>
<EditorBrowsable(EditorBrowsableState.Always)>
Public Sub ForEachFlag(Of T)(ByVal sender As [Enum],
                             ByVal action As Action(Of T))

    For Each flag As T In sender.Flags(Of T)
        action.Invoke(flag)
    Next flag

End Sub
```

Usage Example:

```
Dim flags As FileAttributes = (FileAttributes.ReadOnly Or FileAttributes.Hidden)

flags.ForEachFlag(Of FileAttributes)(
      Sub(ByVal x As FileAttributes)
          Console.WriteLine(x.ToString())
      End Sub)
```

Section 10.11: Determine the amount of flags in a flag combination

The next example is intended to count the amount of flags in the specified flag combination.

The example is provided as a extension method:

```
<DebuggerStepThrough>
<Extension>
<EditorBrowsable(EditorBrowsableState.Always)>
Public Function CountFlags(ByVal sender As [Enum]) As Integer
    Return sender.ToString().Split(","c).Count()
End Function
```

Usage Example:

```
Dim flags As FileAttributes = (FileAttributes.Archive Or FileAttributes.Compressed)
Dim count As Integer = flags.CountFlags()
```

```
Console.WriteLine(count)
```

Section 10.12: Find the nearest value in a Enum

The next code illustrates how to find the nearest value of a **Enum**.

First we define this **Enum** that will serve to specify search criteria (search direction)

```
Public Enum EnumFindDirection As Integer
    Nearest = 0
    Less = 1
    LessOrEqual = 2
    Greater = 3
    GreaterOrEqual = 4
End Enum
```

And now we implement the search algorithm:

```
<DebuggerStepThrough>
Public Shared Function FindNearestEnumValue(Of T)(ByVal value As Long,
                                                  ByVal direction As EnumFindDirection) As T

    Select Case direction

        Case EnumFindDirection.Nearest
            Return (From enumValue As T In [Enum].GetValues(GetType(T)).Cast(Of T)()
                    Order By Math.Abs(value - Convert.ToInt64(enumValue))
                    ).FirstOrDefault

        Case EnumFindDirection.Less
            If value < Convert.ToInt64([Enum].GetValues(GetType(T)).Cast(Of T).First) Then
                Return [Enum].GetValues(GetType(T)).Cast(Of T).FirstOrDefault

            Else
                Return (From enumValue As T In [Enum].GetValues(GetType(T)).Cast(Of T)()
                        Where Convert.ToInt64(enumValue) < value
                        ).LastOrDefault
            End If

        Case EnumFindDirection.LessOrEqual
            If value < Convert.ToInt64([Enum].GetValues(GetType(T)).Cast(Of T).First) Then
                Return [Enum].GetValues(GetType(T)).Cast(Of T).FirstOrDefault

            Else
                Return (From enumValue As T In [Enum].GetValues(GetType(T)).Cast(Of T)()
                        Where Convert.ToInt64(enumValue) <= value
                        ).LastOrDefault
            End If

        Case EnumFindDirection.Greater
            If value > Convert.ToInt64([Enum].GetValues(GetType(T)).Cast(Of T).Last) Then
                Return [Enum].GetValues(GetType(T)).Cast(Of T).LastOrDefault

            Else
                Return (From enumValue As T In [Enum].GetValues(GetType(T)).Cast(Of T)()
                        Where Convert.ToInt64(enumValue) > value
                        ).FirstOrDefault
            End If

        Case EnumFindDirection.GreaterOrEqual
```

```
            If value > Convert.ToInt64([Enum].GetValues(GetType(T)).Cast(Of T).Last) Then
                Return [Enum].GetValues(GetType(T)).Cast(Of T).LastOrDefault

            Else
                Return (From enumValue As T In [Enum].GetValues(GetType(T)).Cast(Of T)()
                        Where Convert.ToInt64(enumValue) >= value
                        ).FirstOrDefault
            End If

    End Select

End Function
```

Usage Example:

```
Public Enum Bitrate As Integer
    Kbps128 = 128
    Kbps192 = 192
    Kbps256 = 256
    Kbps320 = 320
End Enum

Dim nearestValue As Bitrate = FindNearestEnumValue(Of Bitrate)(224,
EnumFindDirection.GreaterOrEqual)
```