Constants and Enumerations in Visual Basic

Visual Studio 2015

Constants are a way to use meaningful names in place of a value that does not change. Constants store values that, as the name implies, remain constant throughout the execution of an application. You can use constants to provide meaningful names, instead of numbers, making your code more readable.

Enumerations provide a convenient way to work with sets of related constants, and to associate constant values with names. For example, you can declare an enumeration for a set of integer constants associated with the days of the week, and then use the names of the days rather than their integer values in your code.

In This Section

Term	Definition
Constants Overview (Visual Basic)	Topics in this section describe constants and their uses.
Enumerations Overview (Visual Basic)	Topics in this section describe enumerations and their uses.

Related Sections

Term	Definition
Const Statement (Visual Basic)	Describes the **Const** statement, which is used to declare constants.
Enum Statement (Visual Basic)	Describes the **Enum** statement, which is used to create enumerations.
Option Explicit Statement (Visual Basic)	Describes the **Option Explicit** statement, which is used at module level to force explicit declaration of all variables in that module.
Option Infer Statement	Describes the **Option Infer** statement, which enables the use of local type inference in declaring variables.
Option Strict Statement	Describes the **Option Strict** statement, which restricts implicit data type conversions to only widening conversions, disallows late binding, and disallows implicit typing that results in an **Object** type.

User-Defined Constants (Visual Basic)

Visual Studio 2015

A constant is a meaningful name that takes the place of a number or string that does not change. Constants store values that, as the name implies, remain constant throughout the execution of an application. You can use constants that are defined by the controls or components you work with, or you can create your own. Constants you create yourself are described as *user-defined*.

You declare a constant with the **Const** statement, using the same guidelines you would for creating a variable name. If **Option Strict** is **On**, you must explicitly declare the constant type.

Const Statement Usage

A **Const** statement can represent a mathematical or date/time quantity:

VB

```
Const conPi = 3.14159265358979
Public Const conMaxPlanets As Integer = 9
Const conReleaseDate = #1/1/1995#
```

It also can define **String** constants:

VB

```
Public Const conVersion = "07.10.A"
Const conCodeName = "Enigma"
```

The expression on the right side of the equal sign (**=**) is often a number or literal string, but it also can be an expression that results in a number or string (although that expression cannot contain calls to functions). You can even define constants in terms of previously defined constants:

VB

```
Const conPi2 = conPi * 2
```

Scope of User-Defined Constants

A **Const** statement's scope is the same as that of a variable declared in the same location. You can specify scope in any of the following ways:

- To create a constant that exists only within a procedure, declare it within that procedure.

- To create a constant available to all procedures within a class, but not to any code outside that module, declare it in the declarations section of the class.
- To create a constant that is available to all members of an assembly, but not to outside clients of the assembly, declare it using the **Friend** keyword in the declarations section of the class.
- To create a constant available throughout the application, declare it using the **Public** keyword in the declarations section the class.

For more information, see How to: Declare A Constant (Visual Basic).

Avoiding Circular References

Because constants can be defined in terms of other constants, it is possible to inadvertently create a *cycle*, or circular reference, between two or more constants. A cycle occurs when you have two or more public constants, each of which is defined in terms of the other, as in the following example:

VB

```
Public Const conA = conB * 2
```

VB

```
Public Const conB = conA / 2
```

If a cycle occurs, Visual Basic generates a compiler error.

See Also

Const Statement (Visual Basic)
Constant and Literal Data Types (Visual Basic)
Constants and Enumerations in Visual Basic
Constants and Enumerations (Visual Basic)
Enumerations Overview (Visual Basic)
Constants Overview (Visual Basic)
How to: Declare Enumerations (Visual Basic)
Enumerations and Name Qualification (Visual Basic)
Option Strict Statement

Enumerations Overview (Visual Basic)

Visual Studio 2015

Enumerations provide a convenient way to work with sets of related constants and to associate constant values with names. For example, you can declare an enumeration for a set of integer constants associated with the days of the week, and then use the names of the days rather than their integer values in your code.

Tasks involving Enumerations

The following table lists common tasks involving enumerations.

To do this	See
Find a pre-defined enumeration	Constants and Enumerations (Visual Basic)
Declare an enumeration	How to: Declare Enumerations (Visual Basic)
Fully qualify an enumeration's name	Enumerations and Name Qualification (Visual Basic)
Refer to an enumeration member	How to: Refer to an Enumeration Member (Visual Basic)
Iterate through an enumeration	How to: Iterate Through An Enumeration in Visual Basic
Determine the string associated with an enumeration	How to: Determine the String Associated with an Enumeration Value (Visual Basic)
Decide when to use an enumeration	When to Use an Enumeration (Visual Basic)

See Also

Constants Overview (Visual Basic)
User-Defined Constants (Visual Basic)
How to: Declare A Constant (Visual Basic)
Constant and Literal Data Types (Visual Basic)
Enum Statement (Visual Basic)

How to: Iterate Through An Enumeration in Visual Basic

Visual Studio 2015

Enumerations provide a convenient way to work with sets of related constants, and to associate constant values with names. To iterate through an enumeration, you can move it into an array using the GetValues method. You could also iterate through an enumeration using a **For...Each** statement, using the GetNames or GetValues method to extract the string or numeric value.

To iterate through an enumeration

- Declare an array and convert the enumeration to it with the GetValues method before passing the array as you would any other variable. The following example displays each member of the enumeration FirstDayOfWeek as it iterates through the enumeration.

VB

```
Dim items As Array
items = System.Enum.GetValues(GetType(FirstDayOfWeek))
Dim item As String
For Each item In items
    MsgBox(item)
Next
```

See Also

Enumerations Overview (Visual Basic)
How to: Declare Enumerations (Visual Basic)
When to Use an Enumeration (Visual Basic)
How to: Determine the String Associated with an Enumeration Value (Visual Basic)
How to: Refer to an Enumeration Member (Visual Basic)
Enumerations and Name Qualification (Visual Basic)
Arrays in Visual Basic

How to: Determine the String Associated with an Enumeration Value (Visual Basic)

Visual Studio 2015

The GetValues and GetNames methods allow you to determine the strings and values associated with enumeration members.

To determine the string associated with an enumeration

- Use the GetNames method to retrieve the strings associated with the enumeration members. This example declares an enumeration, `flavorEnum`, then uses the GetNames method to display the strings associated with each member.

VB

```
Public Enum flavorEnum
  salty
  sweet
  sour
  bitter
End Enum

Private Sub TestMethod()
  MsgBox("The strings in the flavorEnum are:")
  Dim i As String
  For Each i In [Enum].GetNames(GetType(flavorEnum))
      MsgBox(i)
  Next
End Sub
```

See Also

GetValues
GetNames
Enum
How to: Declare Enumerations (Visual Basic)
How to: Refer to an Enumeration Member (Visual Basic)
Enumerations and Name Qualification (Visual Basic)
How to: Iterate Through An Enumeration in Visual Basic
When to Use an Enumeration (Visual Basic)
Enum Statement (Visual Basic)