VB.NET - DATE & TIME

Most of the softwares you write need implementing some form of date functions returning current date and time. Dates are so much part of everyday life that it becomes easy to work with them without thinking. VB.Net also provides powerful tools for date arithmetic that makes manipulating dates easy.

The **Date** data type contains date values, time values, or date and time values. The default value of Date is 0:00:00 *midnight* on January 1, 0001. The equivalent .NET data type is **System.DateTime**.

The **DateTime** structure represents an instant in time, typically expressed as a date and time of day

```
'Declaration
<SerializableAttribute> _
Public Structure DateTime _
 Implements IComparable, IFormattable, IConvertible, ISerializable,
 IComparable(Of DateTime), IEquatable(Of DateTime)
```

You can also get the current date and time from the DateAndTime class.

The **DateAndTime** module contains the procedures and properties used in date and time operations.

```
'Declaration
<StandardModuleAttribute> _
Public NotInheritable Class DateAndTime
```

Note:

Both the DateTime structure and the DateAndTime module contain properties like **Now** and **Today**, so often beginners find it confusing. The DateAndTime class belongs to the Microsoft.VisualBasic namespace and the DateTime structure belongs to the System namespace.
Therefore, using the later would help you in porting your code to another .Net language like C#. However, the DateAndTime class/module contains all the legacy date functions available in Visual Basic.

Properties and Methods of the DateTime Structure

The following table lists some of the commonly used **properties** of the **DateTime** Structure:

S.N	Property	Description
1	**Date**	Gets the date component of this instance.
2	**Day**	Gets the day of the month represented by this instance.
3	**DayOfWeek**	Gets the day of the week represented by this instance.
4	**DayOfYear**	Gets the day of the year represented by this instance.
5	**Hour**	Gets the hour component of the date represented by this instance.
6	**Kind**	Gets a value that indicates whether the time represented by this instance is based on local time, Coordinated Universal Time *UTC*, or neither.
7	**Millisecond**	Gets the milliseconds component of the date represented by this instance.
8	**Minute**	Gets the minute component of the date represented by this instance.
9	**Month**	Gets the month component of the date represented by this instance.
10	**Now**	Gets a **DateTime** object that is set to the current date and time

		on this computer, expressed as the local time.
11	**Second**	Gets the seconds component of the date represented by this instance.
12	**Ticks**	Gets the number of ticks that represent the date and time of this instance.
13	**TimeOfDay**	Gets the time of day for this instance.
14	**Today**	Gets the current date.
15	**UtcNow**	Gets a **DateTime** object that is set to the current date and time on this computer, expressed as the Coordinated Universal Time *UTC*.
16	**Year**	Gets the year component of the date represented by this instance.

The following table lists some of the commonly used **methods** of the **DateTime** structure:

S.N	Method Name & Description
1	**Public Function Add** ***valueAsTimeSpan*** **As DateTime** Returns a new DateTime that adds the value of the specified TimeSpan to the value of this instance.
2	**Public Function AddDays** ***valueAsDouble*** **As DateTime** Returns a new DateTime that adds the specified number of days to the value of this instance.
3	**Public Function AddHours** ***valueAsDouble*** **As DateTime** Returns a new DateTime that adds the specified number of hours to the value of this instance.
4	**Public Function AddMinutes** ***valueAsDouble*** **As DateTime** Returns a new DateTime that adds the specified number of minutes to the value of this instance.
5	**Public Function AddMonths** ***monthsAsInteger*** **As DateTime** Returns a new DateTime that adds the specified number of months to the value of this instance.
6	**Public Function AddSeconds** ***valueAsDouble*** **As DateTime** Returns a new DateTime that adds the specified number of seconds to the value of this instance.
7	**Public Function AddYears** ***valueAsInteger*** **As DateTime** Returns a new DateTime that adds the specified number of years to the value of this instance.
8	**Public Shared Function Compare** ***t1AsDateTime, t2AsDateTime*** **As Integer** Compares two instances of DateTime and returns an integer that indicates whether the first instance is earlier than, the same as, or later than the second instance.

9

Public Function CompareTo *valueAsDateTime* **As Integer**

Compares the value of this instance to a specified DateTime value and returns an integer that indicates whether this instance is earlier than, the same as, or later than the specified DateTime value.

10

Public Function Equals *valueAsDateTime* **As Boolean**

Returns a value indicating whether the value of this instance is equal to the value of the specified DateTime instance.

11

Public Shared Function Equals *t1AsDateTime, t2AsDateTime* **As Boolean**

Returns a value indicating whether two DateTime instances have the same date and time value.

12

Public Overrides Function ToString As String

Converts the value of the current DateTime object to its equivalent string representation.

The above list of methods is not exhaustive, please visit Microsoft documentation for the complete list of methods and properties of the DateTime structure.

Creating a DateTime Object

You can create a DateTime object in one of the following ways:

- By calling a DateTime constructor from any of the overloaded DateTime constructors.
- By assigning the DateTime object a date and time value returned by a property or method.
- By parsing the string representation of a date and time value.
- By calling the DateTime structure's implicit default constructor.

The following example demonstrates this:

```
Module Module1
   Sub Main()
      'DateTime constructor: parameters year, month, day, hour, min, sec
      Dim date1 As New Date(2012, 12, 16, 12, 0, 0)
      'initializes a new DateTime value
      Dim date2 As Date = #12/16/2012 12:00:52 AM#
      'using properties
      Dim date3 As Date = Date.Now
      Dim date4 As Date = Date.UtcNow
      Dim date5 As Date = Date.Today
      Console.WriteLine(date1)
      Console.WriteLine(date2)
      Console.WriteLine(date3)
      Console.WriteLine(date4)
      Console.WriteLine(date5)
      Console.ReadKey()
   End Sub
End Module
```

When the above code was compiled and executed, it produces the following result:

```
12/16/2012 12:00:00 PM
12/16/2012 12:00:52 PM
12/12/2012 10:22:50 PM
12/12/2012 12:00:00 PM
```

Getting the Current Date and Time:

The following programs demonstrate how to get the current date and time in VB.Net:

Current Time:

```
Module dateNtime
```

```
   Sub Main()
      Console.Write("Current Time: ")
      Console.WriteLine(Now.ToLongTimeString)
      Console.ReadKey()
   End Sub
End Module
```

When the above code is compiled and executed, it produces the following result:

```
Current Time: 11 :05 :32 AM
```

Current Date:

```
Module dateNtime
   Sub Main()
      Console.WriteLine("Current Date: ")
      Dim dt As Date = Today
      Console.WriteLine("Today is: {0}", dt)
      Console.ReadKey()
   End Sub
End Module
```

When the above code is compiled and executed, it produces the following result:

```
Today is: 12/11/2012 12:00:00 AM
```

Formatting Date

A Date literal should be enclosed within hash signs `##`, and specified in the format M/d/yyyy, for example #12/16/2012#. Otherwise, your code may change depending on the locale in which your application is running.

For example, you specified Date literal of #2/6/2012# for the date February 6, 2012. It is alright for the locale that uses mm/dd/yyyy format. However, in a locale that uses dd/mm/yyyy format, your literal would compile to June 2, 2012. If a locale uses another format say, yyyy/mm/dd, the literal would be invalid and cause a compiler error.

To convert a Date literal to the format of your locale or to a custom format, use the **Format** function of String class, specifying either a predefined or user-defined date format.

The following example demonstrates this.

```
Module dateNtime
   Sub Main()
      Console.WriteLine("India Wins Freedom: ")
      Dim independenceDay As New Date(1947, 8, 15, 0, 0, 0)
      ' Use format specifiers to control the date display.
      Console.WriteLine(" Format 'd:' " & independenceDay.ToString("d"))
      Console.WriteLine(" Format 'D:' " & independenceDay.ToString("D"))
      Console.WriteLine(" Format 't:' " & independenceDay.ToString("t"))
      Console.WriteLine(" Format 'T:' " & independenceDay.ToString("T"))
      Console.WriteLine(" Format 'f:' " & independenceDay.ToString("f"))
      Console.WriteLine(" Format 'F:' " & independenceDay.ToString("F"))
      Console.WriteLine(" Format 'g:' " & independenceDay.ToString("g"))
      Console.WriteLine(" Format 'G:' " & independenceDay.ToString("G"))
      Console.WriteLine(" Format 'M:' " & independenceDay.ToString("M"))
      Console.WriteLine(" Format 'R:' " & independenceDay.ToString("R"))
      Console.WriteLine(" Format 'y:' " & independenceDay.ToString("y"))
      Console.ReadKey()
   End Sub
End Module
```

When the above code is compiled and executed, it produces the following result:

```
India Wins Freedom:
Format 'd:' 8/15/1947
Format 'D:' Friday, August 15, 1947
Format 't:' 12:00 AM
Format 'T:' 12:00:00 AM
Format 'f:' Friday, August 15, 1947 12:00 AM
Format 'F:' Friday, August 15, 1947 12:00:00 AM
Format 'g:' 8/15/1947 12:00 AM
Format 'G:' 8/15/1947 12:00:00 AM
Format 'M:' 8/15/1947 August 15
Format 'R:' Fri, 15 August 1947 00:00:00 GMT
Format 'y:' August, 1947
```

Predefined Date/Time Formats

The following table identifies the predefined date and time format names. These may be used by name as the style argument for the **Format** function:

Format	Description
General Date, or G	Displays a date and/or time. For example, 1/12/2012 07:07:30 AM.
Long Date,Medium Date, or D	Displays a date according to your current culture's long date format. For example, Sunday, December 16, 2012.
Short Date, or d	Displays a date using your current culture's short date format. For example, 12/12/2012.
Long Time,Medium Time, orT	Displays a time using your current culture's long time format; typically includes hours, minutes, seconds. For example, 01:07:30 AM.
Short Time or t	Displays a time using your current culture's short time format. For example, 11:07 AM.
f	Displays the long date and short time according to your current culture's format. For example, Sunday, December 16, 2012 12:15 AM.
F	Displays the long date and long time according to your current culture's format. For example, Sunday, December 16, 2012 12:15:31 AM.
g	Displays the short date and short time according to your current culture's format. For example, 12/16/2012 12:15 AM.
M, m	Displays the month and the day of a date. For example, December 16.
R, r	Formats the date according to the RFC1123Pattern property.
s	Formats the date and time as a sortable index. For example, 2012-12-16T12:07:31.
u	Formats the date and time as a GMT sortable index. For example, 2012-12-16 12:15:31Z.
U	Formats the date and time with the long date and long time as GMT. For example, Sunday, December 16, 2012 6:07:31 PM.
Y, y	Formats the date as the year and month. For example, December, 2012.

For other formats like user-defined formats, please consult Microsoft Documentation.

Properties and Methods of the DateAndTime Class

The following table lists some of the commonly used **properties** of the **DateAndTime** Class:

S.N	Property	Description
1	**Date**	Returns or sets a String value representing the current date according to your system.
2	**Now**	Returns a Date value containing the current date and time according to your system.
3	**TimeOfDay**	Returns or sets a Date value containing the current time of day according to your system.
4	**Timer**	Returns a Double value representing the number of seconds elapsed since midnight.
5	**TimeString**	Returns or sets a String value representing the current time of day according to your system.
6	**Today**	Gets the current date.

The following table lists some of the commonly used **methods** of the **DateAndTime** class:

S.N	Method Name & Description
1	**Public Shared Function DateAdd** *IntervalAsDateInterval, NumberAsDouble, DateValueAsDateTime* **As DateTime** Returns a Date value containing a date and time value to which a specified time interval has been added.
2	**Public Shared Function DateAdd** *IntervalAsString, NumberAsDouble, DateValueAsObject* **As DateTime** Returns a Date value containing a date and time value to which a specified time interval has been added.
3	**Public Shared Function DateDiff** *IntervalAsDateInterval, Date1AsDateTime, Date2AsDateTime, DayOfWeekAsFirstDayOfWeek, WeekOfYearAsFirstWeekOfYear* **As Long** Returns a Long value specifying the number of time intervals between two Date values.
4	**Public Shared Function DatePart** *IntervalAsDateInterval, DateValueAsDateTime, FirstDayOfWeekValueAsFirstDayOfWeek, FirstWeekOfYearValueAsFirstWeekOfYear* **As Integer** Returns an Integer value containing the specified component of a given Date value.
5	**Public Shared Function Day** *DateValueAsDateTime* **As Integer** Returns an Integer value from 1 through 31 representing the day of the month.
6	**Public Shared Function Hour** *TimeValueAsDateTime* **As Integer** Returns an Integer value from 0 through 23 representing the hour of the day.
7	**Public Shared Function Minute** *TimeValueAsDateTime* **As Integer** Returns an Integer value from 0 through 59 representing the minute of the hour.
8	**Public Shared Function Month** *DateValueAsDateTime* **As Integer** Returns an Integer value from 1 through 12 representing the month of the year.
9	**Public Shared Function MonthName** *MonthAsInteger, AbbreviateAsBoolean* **As String** Returns a String value containing the name of the specified month.
10	**Public Shared Function Second** *TimeValueAsDateTime* **As Integer** Returns an Integer value from 0 through 59 representing the second of the minute.
11	**Public Overridable Function ToString As String** Returns a string that represents the current object.
12	**Public Shared Function Weekday** *DateValueAsDateTime, DayOfWeekAsFirstDayOfWeek* **As Integer** Returns an Integer value containing a number representing the day of the week.
13	**Public Shared Function WeekdayName** *WeekdayAsInteger, AbbreviateAsBoolean, FirstDayOfWeekValueAsFirstDayOfWeek* **As String** Returns a String value containing the name of the specified weekday.

14

Public Shared Function Year ***DateValueAsDateTime*** **As Integer**

Returns an Integer value from 1 through 9999 representing the year.

The above list is not exhaustive. For complete list of properties and methods of the DateAndTime class, please consult Microsoft Documentation.

The following program demonstrates some of these and methods:

```
Module Module1
   Sub Main()
      Dim birthday As Date
      Dim bday As Integer
      Dim month As Integer
      Dim monthname As String
      ' Assign a date using standard short format.
      birthday = #7/27/1998#
      bday = Microsoft.VisualBasic.DateAndTime.Day(birthday)
      month = Microsoft.VisualBasic.DateAndTime.Month(birthday)
      monthname = Microsoft.VisualBasic.DateAndTime.MonthName(month)
      Console.WriteLine(birthday)
      Console.WriteLine(bday)
      Console.WriteLine(month)
      Console.WriteLine(monthname)
      Console.ReadKey()
   End Sub
End Module
```

When the above code is compiled and executed, it produces the following result:

```
7/27/1998 12:00:00 AM
27
7
July
```