La Güeb de Joaquín
Mi sitio de Internet

Interfaz IFormatable

[Descripción general]

Una de las funciones más usadas en nuestras clases, es la función ToString, que se emplea para proporcionar una salida en formato de cadena para los valores de nuestra clase
.NET sugiere implementar la interfaz IFormattable [Ver IFormattable (Interfaz)] para implementar las funciones ToString() que proporcionan funcionalidad para dar formato al valor de un objeto en una representación de cadena.
.NET Framework define y utiliza docenas de interfaces y los programadores expertos en Visual Basic .NET deben aprender a sacar partido de todas ellas

Contenido

Introducción

Un formato, describe cual debe ser la apariencia de una cadena, como debe escribirse la cadena.

- La primera forma de describir el formato de una cadena es utilizar una cadena de Formato. La forma más fácil de entender lo que es una 'cadena de formato' es describir una. A continuación se describe con palabras (en lugar del formato de cadena) un formato para escribir como cadena un número cualquiera. Por ejemplo: el número tendrá separadores de millar -que será el punto (.)-, separador de decimales, -que será la coma (,)- y dos decimales. El número se ajustara a la derecha en un espacio de 20 caracteres. Todo este 'formato' tiene su traducción en una cadena de formato que se describe más adelante.

- Otra forma de describir un formato es indicar la cultura en la que vamos a escribir esa información. Proporcionando un valor de 'CultureInfo' [Ver CultureInfo (Clase)]. Por ejemplo, el numero quiero que se escriba con el formato de la cultura Española, o bien con el formato de la cultura Inglesa. Una forma alternativa a esta es proporciona una clase que implemente la Interfaz IFormatProvider [Ver IFormatProvider (Interfaz)] y que contenga esa información

La interfaz IFormatable [Ver IFormattable (Interfaz)] utiliza las dos informaciones (una cadena de formato, y una Interfaz IFormatProvider) para dar formato, tanto a números como a cadenas de caracteres

La clase para ver el ejemplo

Para ver el concepto vamos a trabajar con una clase Persona que se muestra a continuación

```
Friend Class Persona

    Private _nombre As String = String.Empty
    Private _apellidos As String = String.Empty
    Private _dni As String = String.Empty

    Public Property Nombre() As String
        Get
            Return _nombre
        End Get
        Set(ByVal value As String)
            _nombre = value
        End Set
    End Property


    Public Property Apellidos() As String
        Get
            Return _apellidos
        End Get
        Set(ByVal value As String)
            _apellidos = value
        End Set
    End Property
```

```
    Public Property DNI() As String
        Get
            Return _dni
        End Get
        Set(ByVal value As String)
            _dni = value
        End Set
    End Property


    Public Sub New(ByVal nombre As String, ByVal apellidos As String, ByVal dni As String)
        Me._nombre = nombre
        Me._apellidos = apellidos
        Me._dni = dni
    End Sub

End Class
```

La Interfaz

Firma de la interfaz

La firma de la interface es:

```
    Public Interface IFormattable
```

Firma de las funciones de la interfaz

La función que hay que implementar tiene la firma:

```
Public Function ToString(format As String,IFormatProvider) As String _
               Implements System.IFormattable.ToString
```

Implementando la Interfaz

Paso Uno Definir la interfaz

```
Friend Class Persona
    Implements IConvertible
...
...
End Class
```

Paso Dos - Implementar la función de la interfaz

La siguiente función, es la que implementa la interfaz IFormattable.

```
Public Function ToString(format As String,IFormatProvider) As String _
               Implements System.IFormattable.ToString
```

En ella, el parametro [Format] es una cadena que especifica un formato de impresión. Y [`formatProvider`] admite cualquier objeto qué implemente la interfaz IformatProvider

En este ejemplo, el formato de cadena es el siguiente " | {0,10}, {1,15} | {2,12} | ": y esa cadena significa lo siguiente

- Primero aparece un carácter “|” a modo de separador
- El nombre {0,10}se escribe en un espacio de 10 caracteres y alineado a la Izquierda
- Luego una coma de separación y dos espacio s en blanco“, ”

- Los apellidos {1,15} se escribe en un espacio de 15 caracteres y alineados a la Izquierda
- Un carácter " | " a modo de separador con un espacio delante y otro detrás
- El DNI {2,12} escribe en un espacio de 12 caracteres y alineados a la Izquierda
- Un carácter " | " a modo de separador con un espacio delante y otro detrás

```
Public Function ToString(cadenaFormato As String,IFormatProvider) As String _
               Implements System.IFormattable.ToString

        If String.IsNullOrEmpty(cadenaFormato) = True Then
            cadenaFormato = " | {0,10},  {1,15} | {2,12} | "
        End If

        Return String.Format(System.Globalization.CultureInfo.CurrentCulture, _
                    cadenaFormato, _
                    Me.Nombre, _
                    Me.Apellidos, _
                    Me.DNI)
    End Function
```

Implementación Avanzada

.NET sugiere implementar la interfaz IFormattable para implementar las funciones ToString() que proporcionan funcionalidad para dar formato al valor de un objeto en una representación de cadena.

Pero ya que estamos metidos en harina también podemos implementar las funciones siguientes que hacen un trabajo parecido y así queda todo más completo

En resumen, lo que hago a continuación es definir las cuatro funciones ToString generales que, en principio, deberían implementar todas las clases, y después, cuando defino la interfaz, las funciones de la interfaz apuntan a una de estas funciones que es las que hace el trabajo real que necesita la interfaz. Es una forma de no duplicar código

1) Sombrear la implementación de System.Object

En general, todos los objetos que definimos derivan de la clase System.Object [Ver Object (Clase)], por eso todas heredan la función ToString, que tiene para todas las clases el mismo comportamiento, es decir, devuelve una cadena con el nombre de la clase. Pero si lo que queremos es que muestre información formateada de todos o algunos de los campos de la clase tendremos que "sombrearla" para lo que basta con declararla 'overrides'. Además, como habrá otras implementaciones de la función ToString, tendremos que declararla 'overloads'.

```
    ' ------------------------------------------------------------------------
    ' 1) Sombrear System.Object.ToString
    ' ------------------------------------------------------------------------
    ''' <summary>
    '''  Salida en formato de cadena para los valores de nuestra  clase
    ''' </summary>
    ''' <returns>Una cadena con los datos de la clase</returns>
    ''' <remarks>
    ''' <para>
    '''     Esta función se declara 'overrides' porque sustituye un método que tiene el
    '''     mismo nombre de la clase base, en este caso la clase base es 'Object'
    ''' </para>
    ''' <para>
    '''     Esta función se declara 'Overloads' porque hay varias implementaciones
    '''     de esta función con el mismo nombre pero con firmas diferentes
    ''' </para>
    '''</remarks>
```

```
Public Overloads Overrides Function ToString() As String
    ''-------------------------------------------------------
    '' Forma elemental de obtener la informacion
    ''Return "(" & Me._numerador & ", " & Me._denominador & ")"
    ''-------------------------------------------------------
    Dim cadenaFormato As String = " | {0,10},  {1,15} | {2,12} | "
    Dim formatProvider As System.Globalization.CultureInfo
    formatProvider = System.Globalization.CultureInfo.CurrentCulture
    Return String.Format(formatProvider, cadenaFormato, Me.Nombre, Me.Apellidos, Me.DNI)
End Function
```

2) Function ToString(cadenaFormato)As String

En la siguiente función el parámetro [cadenaFormato] es una cadena que especifica un formato de impresión

```
' ------------------------------------------------------------------------
' 2) Function ToString(cadenaFormato)As String
' ------------------------------------------------------------------------
Public Overloads Function ToString(ByVal cadenaFormato As String) As String
    ' --------------------------------------------------------------------
    'Error, Certainty 95, for SpecifyIFormatProvider
    '    Resolution   : "Because the behavior of 'String.Format(String,
    '                   Object, Object)' could vary based on the current user's
    '                   locale settings, replace this call in 'NumeroRacional.ToString(
    '                   String)' with a call to 'String.Format(IFormatProvider,
    '                   String, ParamArray Object())'. If the result of 'String.Format(
    '                   IFormatProvider, String, ParamArray Object())' will
    '                   be displayed to the user, specify 'CultureInfo.CurrentCulture'
    '                   as the 'IFormatProvider' parameter. Otherwise, if the
    '                   result will be stored and accessed by software, such
    '                   as when it is persisted to disk or to a database, specify
    '                   'CultureInfo.InvariantCulture'."
    '    Help         : http://msdn2.microsoft.com/ms182190(VS.90).aspx  (String)
    ' --------------------------------------------------------------------
    If String.IsNullOrEmpty(cadenaFormato) = True Then
        cadenaFormato = " | {0,10},  {1,15} | {2,12} | "
    End If
    Dim formatProvider As System.Globalization.CultureInfo
    formatProvider = System.Globalization.CultureInfo.CurrentCulture
    ' --------------------------------------------------------------------
    Return String.Format(formatProvider, cadenaFormato, Me.Nombre, Me.Apellidos, Me.DNI)
End Function
```

3) Función que será llamada por la Interfaz IFormattable (Si se Implementa)

```
' ------------------------------------------------------------------------
' 3) Función que será llamada por la Interfaz IFormattable (Si se Implementa)
' ------------------------------------------------------------------------
Public Overloads Function ToString( _
                ByVal cadenaFormato As String, _
                ByVal formatProvider As System.IFormatProvider) As String

    If String.IsNullOrEmpty(cadenaFormato) = True Then
        cadenaFormato = " | {0,10},  {1,15} | {2,12} | "
    End If
    Return String.Format(formatProvider, cadenaFormato, Me.Nombre, Me.Apellidos, Me.DNI)
End Function
```

4) Función que será llamada por la Interfaz Inconvertible (Si se Implementa)

```
' ------------------------------------------------------------------------
' 4) Función que será llamada por la Interfaz IConvertible (Si se Implementa)
'    Esta función tiene que intentar usar la clase [Convert] siempre que sea posible
' ------------------------------------------------------------------------
Public Overloads Function ToString( _
                ByVal formatProvider As System.IFormatProvider) _
                As String
    Dim cadenaFormato As String = " | {0,10},  {1,15} | {2,12} | "
    Return String.Format(formatProvider, cadenaFormato, Me.Nombre, Me.Apellidos, Me.DNI)
End Function
```

5) En la Interfaz IConvertible hay otra función 'ToString'

En la interfaz IConvertible [Ver IConvertible (Interfaz)], Existe otra función ToString(). Evidentemente pertenece a otra interfaz, pero la escribo a continuación para mantener una cierta lógica de presentación y para que se vea también la forma de implementarla.

```
Public Overloads Function ToString( _
                ByVal provider As System.IFormatProvider) _
                As String _
                Implements System.IConvertible.ToString

    Dim cadenaFormato As String = " | {0,10},  {1,15} | {2,12} | "
    Return String.Format(provider, _
                         cadenaFormato, _
                         Me.Nombre, _
                         Me.Apellidos, _
                         Me.DNI)
    ''---------------------------------------------------------
    '' Forma alternativa que realiza el mismo trabajo
    ' Return Me.Nombre & ", " & Me.Apellidos & " - " & Me.DNI
    ''---------------------------------------------------------
End Function
```

Una forma alternativa de implementarla es llamar a la funcion ToString para que haga el trabajo

```
' ------------------------------------------------------------------------
' Interfaz System.IConvertible.ToString
' ------------------------------------------------------------------------
' Esta función lo que hace es llamar a la función que realmente hace el trabajo
' Observa que se utiliza un truco que consiste en declararla como privada y
' apuntar a la función ToString que hace el trabajo.
' De esta forma esta función sólo se ve desde la interfaz (que es lo que queremos)
' El nombre no puede ser ToString, porque ya hay una función con la misma firma y
' con el mismo nombre, por eso le cambio el nombre y le pongo [ToStringIConvertible]
' que considero que es un nombre representativo
' ------------------------------------------------------------------------
Private Overloads Function ToStringIConvertible( _
                ByVal provider As IFormatProvider) As String _
                Implements IConvertible.ToString

    Return ToString(provider)
End Function
```

A modo de resumen

```
#Region "   Las Funciones ToString "


    ' ------------------------------------------------------------------------
    ' Funciones que proporcionan un formato de salida
    ' de Cadena para nuestra Clase
    ' Son cuatro funciones las que debes implementar
    ' ------------------------------------------------------------------------

    ' ------------------------------------------------------------------------
    ' 1) Sombrear System.Object.ToString
    ' ------------------------------------------------------------------------
    ''' <summary>
    '''  Salida en formato de cadena para los valores de nuestra  clase
    ''' </summary>
    ''' <returns>Una cadena con los datos de la clase</returns>
    ''' <remarks>
    ''' <para>
    '''     Esta función se declara 'overrides' porque sustituye un método que tiene
    '''     el mismo nombre de la clase base, en este caso la clase base es 'Object'
    ''' </para>
    ''' <para>
    '''     Esta función se declara 'Overloads' porque hay varias implementaciones
    '''     de esta función con el mismo nombre pero con firmas diferentes
    ''' </para>
    '''</remarks>
    Public Overloads Overrides Function ToString() As String
        ''-------------------------------------------------------
        '' Forma elemental de obtener la informacion
        ''Return "(" & Me._numerador & ", " & Me._denominador & ")"
        ''-------------------------------------------------------
        Dim formatProvider As System.Globalization.CultureInfo
        formatProvider = System.Globalization.CultureInfo.CurrentCulture
        '-------------------------------------------------------
        ' llamar a (3) La funcion que realmente hace el trabajo
        Return Me.ToString(String.Empty, formatProvider)
    End Function



    ' ------------------------------------------------------------------------
    ' 2) Function ToString(cadenaFormato)As String
    ' ------------------------------------------------------------------------
    Public Overloads Function ToString(ByVal cadenaFormato As String) As String
        ' ------------------------------------------------------------------------
        Dim formatProvider As System.Globalization.CultureInfo
        formatProvider = System.Globalization.CultureInfo.CurrentCulture
        ' ------------------------------------------------------------------------
        ' llamar a (3) La funcion que realmente hace el trabajo
        Return Me.ToString(String.Empty, formatProvider)
    End Function



    ' ------------------------------------------------------------------------
    ' Esta es la funcion que realmente hace el trabajo
    ' las demas finciones ToString la llaman para hecer el trabajo
    ' ------------------------------------------------------------------------
    ' 3) Función que será llamada por la Interfaz IFormattable (Si se Implementa)
    ' ------------------------------------------------------------------------
    Public Overloads Function ToString( _
                    ByVal cadenaFormato As String, _
                    ByVal formatProvider As System.IFormatProvider) As String

        If String.IsNullOrEmpty(cadenaFormato) = True Then
            cadenaFormato = " | {0,10},  {1,15} | {2,12} | "
        End If
        Return String.Format(formatProvider, cadenaFormato, Me.Nombre, Me.Apellidos, Me.DNI)
    End Function
```

```
    ' ------------------------------------------------------------------------
    ' 4) Función que será llamada por la Interfaz IConvertible (Si se Implementa)
    '    Esta función tiene que intentar usar la clase [Convert] siempre que sea posible
    ' ------------------------------------------------------------------------
    Public Overloads Function ToString( _
                    ByVal formatProvider As System.IFormatProvider) _
                    As String
        '-------------------------------------------------------------
        ' llamar a (3) La funcion que realmente hace el trabajo
        Return Me.ToString(String.Empty, formatProvider)
    End Function

#End Region


#Region "    Implements IFormattable "

    '**************************************************************************

    ' ------------------------------------------------------------------------
    ' Interfaz System.IFormattable.ToString
    ' ------------------------------------------------------------------------
    ' Esta función lo que hace es llamar a la función que realmente hace el trabajo
    ' Observa que se utiliza un truco que consiste en declararla como privada y
    ' apuntar a la función ToString que hace el trabajo.
    ' De esta forma esta función sólo se ve desde la interfaz (que es lo que queremos)
    ' El nombre no puede ser ToString, porque ya hay una función con la misma firma y
    ' con el mismo nombre, por eso le cambio el nombre y le pongo [ToStringIFormattable]
    ' que considero que es un nombre representativo
    ' ------------------------------------------------------------------------
    Private Overloads Function ToStringIFormattable( _
                               ByVal format As String, _
                               ByVal formatProvider As System.IFormatProvider) As String _
                               Implements System.IFormattable.ToString
        ' ------------------------------------------------------------------------
        ' llamar a (3) La funcion que realmente hace el trabajo de esta interfaz
        Return ToString(format, formatProvider)
    End Function


    '' ------------------------------------------------------------------------
    '' Interfaz System.IConvertible.ToString
    '' ------------------------------------------------------------------------
    '' Esta función lo que hace es llamar a la función que realmente hace el trabajo
    '' Observa que se utiliza un truco que consiste en declararla como privada y
    '' apuntar a la función ToString que hace el trabajo.
    '' De esta forma esta función sólo se ve desde la interfaz (que es lo que queremos)
    '' El nombre no puede ser ToString, porque ya hay una función con la misma firma y
    '' con el mismo nombre, por eso le cambio el nombre y le pongo [ToStringIConvertible]
    '' que considero que es un nombre representativo
    '' ------------------------------------------------------------------------
    'Private Overloads Function ToStringIConvertible( _
    '                ByVal provider As IFormatProvider) As String _
    '                Implements IConvertible.ToString
    ''  ------------------------------------------------------------------------
    ''  llamar a (4) La funcion que realmente hace el trabajo de esta interfaz
    '    Return ToString(provider)
    'End Function


#End Region
```

Más información:

Información adicional sobre formatos

La interfaz se llama IFormatable porque aplica a los datos un formato para convertirlos en una cadena.
Un formato describe la apariencia de un objeto cuando se convierte en una cadena.

.NET Framework proporciona un mecanismo de aplicación de formato para uso general, que se puede personalizar para convertir un valor en una cadena apropiada para su presentación.

- Por ejemplo, a un valor numérico se le puede asignar un formato hexadecimal, con notación científica, o una serie de dígitos separados en grupos con un signo de puntuación especificado por el usuario. A las fechas y horas se les puede asignar un formato apropiado de un país, región o cultura concretos. A una constante enumerada se le puede asignar un formato conforme a su valor numérico o su nombre.

Se puede controlar la aplicación de formato, mediante la especificación de una cadena de formato y un proveedor de formato, o al utilizar los valores predeterminados.

- Una cadena de formato contiene uno o más caracteres especificadores de formato que indican cómo se tiene que convertir un valor.
- Un proveedor de formato suministra información de control adicional, de sustitución y cultural necesaria para convertir un tipo específico.

Sintaxis de elemento de formato
Cada elemento de formato presenta la siguiente sintaxis, [Ver formato compuesto] formada por los siguientes componentes:

{ index[,alignment][:formatString]}

Las llaves ("{" y "}") son necesarias.

Index (Componente)
El componente index obligatorio, denominado también especificador de parámetros, es un número que empieza por 0, que identifica un elemento correspondiente de la lista de objetos. O sea, el elemento de formato cuyo especificador de parámetro es 0 da formato al primer objeto de la lista, el elemento de formato cuyo especificador de parámetro es 1 da formato al segundo objeto de la lista, etc.

Los elementos de formato múltiple se pueden referir al mismo elemento de la lista de objetos mediante la especificación del mismo especificador de parámetro. Por ejemplo, se puede dar formato al mismo valor numérico en formato hexadecimal, científico y de número mediante la especificación de una cadena de formato compuesto como ésta: "{0:X} {0:E} {0:N}".

Cada elemento de formato puede hacer referencia a cualquier objeto de la lista. Por ejemplo, si existen tres objetos, se puede dar formato al segundo, primero y tercer objeto mediante la especificación de una cadena de formato compuesto como ésta: "{1} {0} {2}". Un objeto al que no hace referencia ningún elemento de formato se omite. Se produce una excepción en tiempo de ejecución si un especificador de parámetro designa un elemento fuera de los límites de la lista de objetos.

Alignment (Componente)
El componente opcional alignment es un entero con signo que indica el ancho de campo con formato preferido. Si el valor de alignment es menor que la longitud de la cadena con formato, se omite alignment y se utiliza la longitud de la cadena con formato como el ancho de campo. Los datos con formato del campo están alineados a la derecha si alignment es positivo y, a la izquierda, si alignment es negativo. Si hace falta relleno, se utiliza un espacio en blanco. Si se especifica alignment, es necesaria la coma.

Format String (Componente)
El componente formatString opcional es una cadena de formato adecuada para el tipo de objeto al que se da formato. Especifique una cadena de formato numérico si el objeto correspondiente es un valor numérico, una cadena de formato de fecha y hora si el objeto correspondiente es un objeto DateTime o una cadena de formato de enumeración si el objeto correspondiente es un valor de enumeración. Si no se especifica formatString, se utiliza el especificador de formato general ("G") para un tipo numérico, de fecha y hora o de enumeración. Si se especifica formatString, es necesario el punto.

Cadenas con formato numérico estándar
http://msdn.microsoft.com/es-es/library/dwhawy9k.aspx

- C o c - Moneda
- D o d - Decimal
- E o e - Científico (exponencial)
- F o f - Punto fijo
- G o g - General
- N o n - Número
- P o p - Porcentaje
- R o r - Acción de ida y vuelta
- X o x - Hexadecimal

Cadenas con formato numérico personalizado
http://msdn.microsoft.com/es-es/library/0c899ak8.aspx

- 0 - Marcador de posición cero
- # - Marcador de posición de dígito.
- . - (punto) Separador decimal
- , - (coma) Separador de miles y escala numérica
- % - Marcador de posición de porcentaje.
- E0, E+0, E-0, e0, e+0, e-0 - Notación científica
- \ - Carácter de escape
- 'ABC', "ABC" - Cadena literal
- ; (punto y coma) - Separador de secciones

Llaves de escape
Las llaves de apertura y de cierre se interpretan como el inicio y el final de un elemento de formato. Por lo tanto, debe utilizar una secuencia de escape para que se muestre una llave de apertura o de cierre literal. Especifique dos llaves de apertura ("{{") en el texto fijo para que se muestre una llave de apertura ("{"), o dos llaves de cierre ("}}") para que se muestre una llave de cierre ("}"). Las llaves de un elemento de formato se interpretan secuencialmente, en el orden en que se encuentran. No se admite la interpretación de llaves anidadas.

A modo de comentario

- El modo de interpretar las llaves de escape puede dar lugar a resultados inesperados. Tomemos como ejemplo el elemento de formato "{{{0:D}}}", cuyo propósito es mostrar una llave de apertura, un valor numérico con formato de número decimal y una llave de cierre; pero que, en la práctica, se interpreta de la siguiente forma:
- Las dos primeras llaves de apertura ("{{") son llaves de escape y dan lugar a en una llave de apertura.
- Los tres caracteres siguientes ("{0:") se interpretan como el inicio de un elemento de formato.
- El siguiente carácter ("D") se interpretaría como el especificador de formato numérico estándar decimal, pero las dos llaves de escape siguientes ("}}") dan lugar a una única llave. Como la cadena resultante ("D}") no es un especificador de formato numérico estándar, se interpreta como una cadena de formato personalizado que significa que debe mostrarse la cadena literal "D}".
- La última llave ("}") se interpreta como el final del elemento de formato.
- Como resultado final, se muestra la cadena literal "{D}". No se muestra el valor numérico al que se debía dar formato.
- Una forma de escribir código e impedir que las llaves de escape y los elementos de formato se malinterpreten consiste en dar formato a las llaves y elementos de formato por separado. Es decir, en la primera operación de formato mostrar una llave de apertura literal, en la siguiente operación mostrar el resultado del elemento de formato y, por último, en la operación final mostrar una llave de cierre literal.

Orden de procesamiento

- Si el valor al que se va a dar formato es null (Nothing en Visual Basic), se devuelve una cadena vacía ("").
- Si el tipo al que se va a dar formato implementa la interfaz ICustomFormatter, se llama al método ICustomFormatter..::.Format.
- Si en el paso anterior no se da formato al tipo y el tipo implementa la interfaz IFormattable, se llama al método IFormattable..::.ToString.
- Si en el paso anterior no se ha dado formato al tipo, se llama al método ToString del tipo, que se hereda de la clase Object.
- La alineación se aplica después de que se hayan realizado los pasos anteriores.

Excepciones

- ArgumentNullException - El valor de format es null (Nothing en Visual Basic).

- FormatException - format no es válido, O bien, el número correspondiente a un argumento al que se va a dar formato es menor que cero, o mayor o igual que el número de objetos proporcionados a los que se va a dar formato.

Ejemplo

```
Dim nombre As String = "Perico"
Dim apellidos As String = "El de los Palotes"
Dim edad As Integer = 100

Dim FormatNombre As String = String.Format("Nombre    = |{0,20}|", nombre)
Dim FormatApellidos As String = String.Format("Apellidos = |{0,20}|", apellidos)
Dim FormatPrice As String = String.Format("Edad      = |{0,20:G}|", edad)

Dim FormatNombreYFecha As String = _
String.Format("Nombre = {0}, Horas = {1:hh}, Minutos = {1:mm}", nombre, DateTime.Now)


Console.WriteLine("")
Console.WriteLine("Varios campos de salida")
Console.WriteLine(FormatNombreYFecha)


Console.WriteLine("")
Console.WriteLine("Ajuste a la Derecha en un espacio determinado")
Console.WriteLine(FormatNombre)
Console.WriteLine(FormatApellidos)
Console.WriteLine(FormatPrice)

Console.WriteLine("")
Console.WriteLine("Ajuste a la Izquierda en un espacio determinado")
FormatNombre = String.Format("Nombre    = |{0,-20}|", nombre)
FormatApellidos = String.Format("Apellidos = |{0,-20}|", apellidos)
FormatPrice = String.Format("Edad      = |{0,-20:G}|", edad)

Console.WriteLine(FormatNombre)
Console.WriteLine(FormatApellidos)
Console.WriteLine(FormatPrice)


Console.Read()


'-----------------------------------------------------
' Salida de este codigo
'-----------------------------------------------------
'Varios campos de salida
'Nombre = Perico, Horas = 01, Minutos = 56

'Ajuste a la Derecha en un espacio determinado
'Nombre    = |              Perico|
'Apellidos = |   El de los Palotes|
'Edad      = |                 100|

'Ajuste a la Izquierda en un espacio determinado
'Nombre    = |Perico              |
'Apellidos = |El de los Palotes   |
'Edad      = |100                 |
```

Código de ejemplo

En el siguiente enlace hay un fichero ZIP que contiene ejemplo

- Fichero con el código de ejemplo: [2008_10_08_SolucionInterfazIComparer.zip]
- MD5 checksum: [139F3E0FA74D626AE82DC4212DEF6671]
- MD5 checksum: Información
    - [http://www.elguille.info/colabora/MD5_checksum.aspx]

Información de este Documento

[Historial de este documento]

- *Autor*:
  - *Nombre*: Joaquín Medina Serrano
  - *Email* : joaquin@medina.name
  - *Web*: http://joaquin.medina.name

- *Publicador*:
  - *Nombre*: Joaquín Medina Serrano

- *Fechas* (Formato de fecha ISO 8610:2004. [Año-Mes-DiaTHora:Minutos])
  - *Fecha de Creación*: 2008-09-05T23:07
  - *Fecha de la última modificación*: 2008-10-08T21:42
  - *Fecha de Impresión*: 2008-10-08T21:42

- *Copyright*
  - © Joaquín 'jms32' Medina Serrano 2008 - Reservados todos los derechos."

[Direcciones Web Interesantes]

Biblioteca de clases de .NET Framework
[http://msdn.microsoft.com/es-es/library/ms229335.aspx]

- IFormattable (Interfaz)
  - http://msdn.microsoft.com/library/system.iformattable.aspx
- Object (Clase)
  - http://msdn.microsoft.com/es-es/library/system.object.aspx
- CultureInfo (Clase)
  - http://msdn.microsoft.com/es-es/library/system.globalization.cultureinfo.aspx
- IFormatProvider (Interfaz)
  - http://msdn.microsoft.com/es-es/library/system.iformatprovider.aspx
- Decimal..::.ToString (Método) (String)
  - http://msdn.microsoft.com/es-es/library/fzeeb5cd.aspx
- IConvertible (Interfaz)
  - http://msdn.microsoft.com/es-es/library/system.iconvertible.aspx
- String.Format (Método) (String, Object, Object, Object)
  - http://msdn.microsoft.com/es-es/library/d9t40k6d.aspx

Más información sobre formatos numéricos
Manual del programador de .NET Framework
[http://msdn.microsoft.com/es-es/library/k1s94fta.aspx]

- Información general sobre formatos.
  - [http://msdn.microsoft.com/es-es/library/26etazsy.aspx]
- Cadenas de formato numérico
  - http://msdn.microsoft.com/es-es/library/427bttx3.aspx
- Cadenas con formato numérico estándar
  - http://msdn.microsoft.com/es-es/library/dwhawy9k.aspx

- Cadenas con formato numérico personalizado
  - http://msdn.microsoft.com/es-es/library/0c899ak8.aspx
- Formatos compuestos
  - http://msdn.microsoft.com/es-es/library/txafckwd.aspx

[Palabras Clave]

- IFormatable, 'interfaz IFormatable', cadenas, ‘formato numérico’, Cultura, CultureInfo, IFormatProvider, ‘interfaz IFormatProvider’,

[Grupo de documentos]

- [Documento Index]
- [Documento Start]