

La Güeb de Joaquín Programación VB .NET

Estándar de Programación Documentación de Programas

Introducción

Este documento muestra como documentar programas escritos en Visual Basic con etiquetas XML siguiendo la recomendación de Microsoft para documentar programas escritos en C#

Este documento explica brevemente el estándar empleado en la documentación de programas realizados por Joaquín Medina Serrano

Situacion en la red:

[http://jms32.eresmas.net/tacticos/programacion/VBNet/VBN01/
DocumenCodigo/Estandar_DocumentacionXML.pdf](http://jms32.eresmas.net/tacticos/programacion/VBNet/VBN01/DocumenCodigo/Estandar_DocumentacionXML.pdf)

Estándares empleados

Para la definición de este modelo se ha seguido fundamentalmente las recomendaciones de Microsoft y muchas de ellas están extractadas y/o copiadas de la biblioteca de ayuda MSDN.

Índice del documento

INTRODUCCIÓN	1
ESTÁNDARES EMPLEADOS	1
GENERALES	3
DOCUMENTANDO CON C# .NET	3
DOCUMENTANDO CON VB .NET	3
<i>Requisitos</i>	4
<i>Como funciona</i>	4
CONVENCIONES BÁSICAS PARA LA DOCUMENTACIÓN	9
ETIQUETAS RECOMENDADAS POR MICROSOFT	10
DELIMITADORES PARA ETIQUETAS DE DOCUMENTACIÓN	11
<i>Etiqueta <c></i>	12
<i>Etiqueta <code></i>	13
<i>Etiqueta <example></i>	14
<i>Etiqueta <exception></i>	15
<i>Etiqueta <include></i>	16
<i>Etiqueta <list></i>	18
<i>Etiqueta <para></i>	21
<i>Etiqueta <param></i>	22
<i>Etiqueta <paramref></i>	23
<i>Etiqueta <permission></i>	24
<i>Etiqueta <remarks></i>	25
<i>Etiqueta <returns></i>	26
<i>Etiqueta <see></i>	27
<i>Etiqueta <seealso></i>	28
<i>Etiqueta <summary></i>	29
<i>Etiqueta <value></i>	30
EJEMPLOS	31
<i>Ejemplo de documentación de un Namespace</i>	31
<i>Ejemplo de documentación de una clase</i>	32
<i>Ejemplo de documentación de una constante</i>	35
<i>Ejemplo de documentación de un constructor</i>	36
<i>Ejemplo de documentación de un método</i>	38
<i>Ejemplo de documentación de una propiedad</i>	42
<i>Resultados del ejemplo</i>	43
REFERENCIA BIBLIOGRÁFICA	43

Generales

Existen dos tipos de documentación de software: externa e interna. La documentación externa, como por ejemplo las especificaciones, los archivos de ayuda y los documentos de diseño, se mantiene fuera del código fuente. La documentación interna está formada por los comentarios que los programadores escriben dentro del código fuente durante la fase de desarrollo.

Uno de los problemas de la documentación de software interna es garantizar que se mantienen y actualizan los comentarios al mismo tiempo que el código fuente. Aunque unos buenos comentarios en el código fuente no tienen ningún valor en el tiempo de ejecución, resultan valiosísimos para un programador que tenga que mantener una parte de software particularmente intrincada o compleja.

Documentando con C# .Net

.Net Framework introdujo de manera nativa únicamente al compilador de C# la opción de generar un documento XML a partir de etiquetas específicas escritas en formato XML C#

```
///<summary>
/// Esta clase es de muestra para la documentación
///</summary>
public class documentacion
{
    public documentacion()
    {
        //
        // TODO: Add constructor logic here
        //
    }
}
```

Documentando con VB .Net

El que únicamente este incluido de manera nativa en C# no excluye a Vb.Net o algún otro lenguaje, ya que existen varios complementos como VbCommenter que permiten tener una funcionalidad similar a la que ofrece el compilador de C#, el de generar un archivo XML a partir las etiquetas escritas. Por ejemplo en VB un código documentado de la forma anterior quedaría así:

```
''' <summary>
''' Esta clase es de muestra para la documentación
''' </summary>
Public Class documentacion
    Public Sub New()
        ' hacer lo que sea
    End Sub
End Class
```

Requisitos

Para poder documentar un código escrito en VB . NET hace falta los siguientes requisitos previos:

- Imprimir este documento. No es un requisito imprescindible, pero el documento es lo suficientemente largo y complejo como para que merezca la pena hacerlo.
- Descargar los programas necesarios
 - Programa [**VBCommenter PowerToy**] que es un programa que una vez instalado se sitúa como un complemento de Visual Estudio, y se puede acceder a él, a través del [menú Herramientas]. Prueba en esta dirección <http://www.15seconds.com/issue/040302.htm>, si no corresponde al programa [VBCommenter] recurre a [Google](#) para localizar la ultima versión de este programa y descargalo en tu ordenador. Este programa activa en Visual Estudio la posibilidad de utilizar etiquetas XML de marcado, y también genera un documento XML con la información de las etiquetas utilizadas
 - Programa [**Ndoc**] que lee el documento XML generado por [VBCommenter] y lo transforma en una(s) paginas Web con un formato de presentación parecido al MSDN de Microsoft. También tiene la opción de compilar todas esas paginas en un fichero de ayuda Microsoft [*.chm]. Prueba a bajártelo en la página siguiente <http://sourceforge.net/projects/ndoc/> y si no recurre a [Google](#) para localizar la ultima versión y descargarla en tu ordenador.
- Instalación de los programas descargados
 - Observación importante. Todo esto solo funciona con VB NET 2003
 - (1)Cierra todas las instancias de Visual Studio
 - (2)Instala el programa [**Ndoc**] ejecutando el instalador NDocSetup.Exe
 - (3)Instala el programa [**VBCommenter**] ejecutando VBCommenter.msi
 - (4)El comentador se cargará la próxima vez que se ejecute Visual Studio.
 - (5)La configuración de [VBCommenter] se realiza desde Visual Estudio a través del [Menú -->Herramientas --> VBCommenter Options]

Como funciona.

Evidentemente el **primer paso** es documentar el código, y a esa parte se dedica la segunda parte de este documento, donde se describen las etiquetas recomendadas por Microsoft y su forma de usarse.

NOTA.: XML no reconoce como caracteres estándar las vocales acentuadas, además, éste programa tiene un origen anglosajón y ni siquiera se contempla la posibilidad de su utilización, así que si quieres que en el documento final no aparezcan palabras a las que les faltan vocales (p.e. camin en lugar de camión, pgina en lugar de página) evita utilizarlas

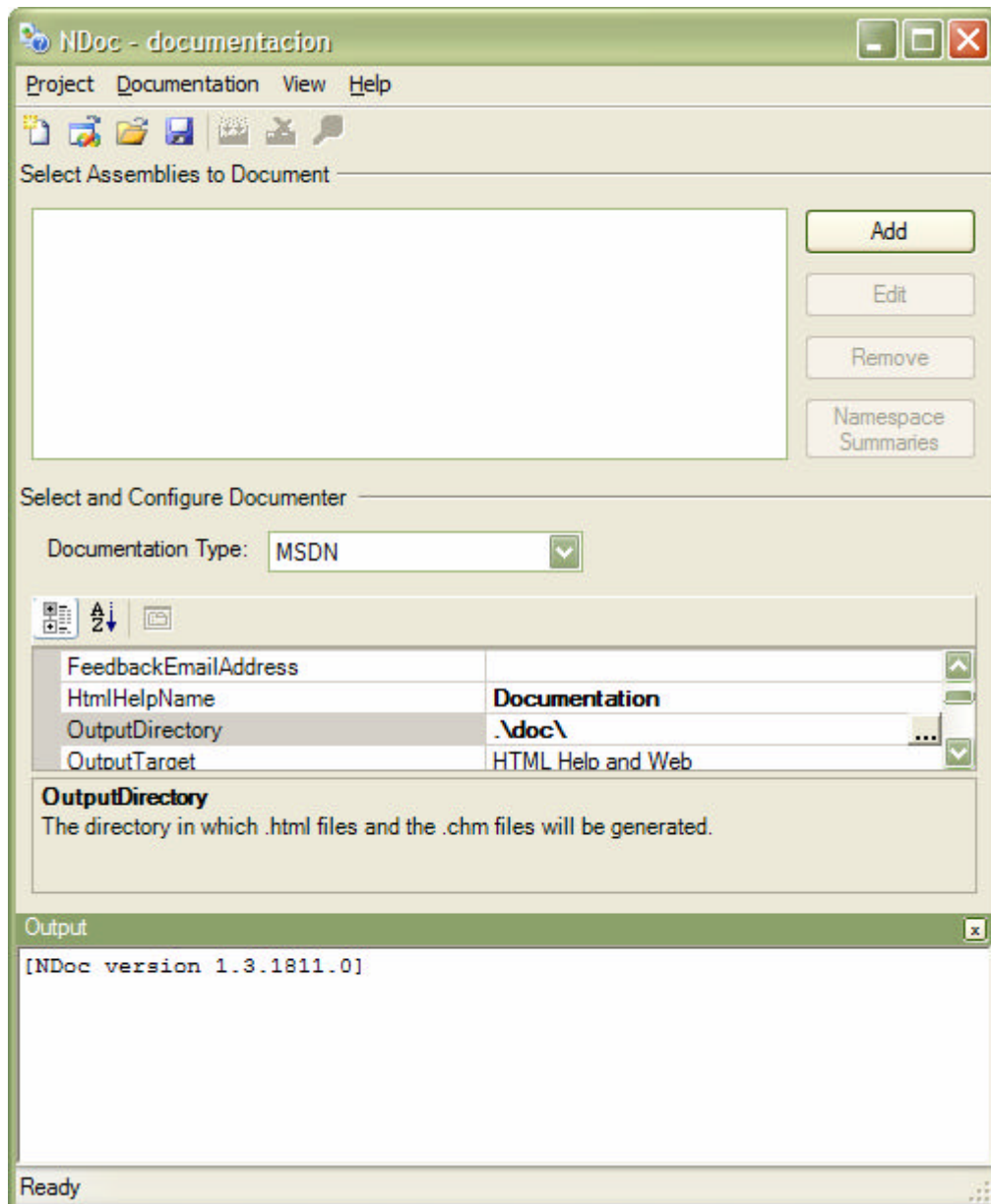
Partiendo de la base de que tenemos, por ejemplo, un proyecto con todas sus clases debidamente documentadas y comentadas, cada vez que compilemos el proyecto (generado Toda la solución) se generará un fichero XML que recogerá todas las etiquetas empleadas.

El documento XML se sitúa en TODOS los directorios del ensamblado, es decir, en el directorio donde están las clases, en el directorio [Bin] y en el directorio [obj\Debug]

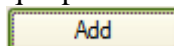
Muy Importante.: En ese directorio [obj\Debug] también se sitúa un documento de texto [VBCommenterLog.txt] que contiene los errores que han aparecido en el código. Por ejemplo, no hemos cerrado una etiqueta, o nos hemos dejado por algún sitio un angulito, etc. **Antes de generar la documentación de ayuda**, hay que comprobar que no existe ningún error sintáctico en las etiquetas XML de comentarios, y si existe corregirlo.

Bien, en este momento hemos escrito nuestro código VB NET, lo hemos comentado en formato XML (con las etiquetas adecuadas) lo hemos compilado (generado Toda la solución) y no hemos tenido ningún error de sintaxis (en el fichero [obj\Debug\VBCommenterLog.txt]) por lo que podemos pasar a generar la documentación de ayuda, para ello utilizaremos el programa **Ndoc**

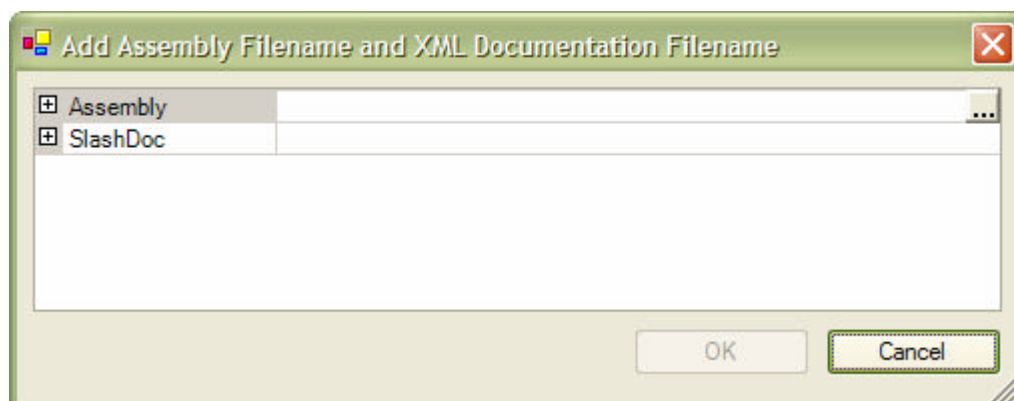
Para ejecutar el programa Ndoc pulsaremos el botón {Inicio} de la {barra de tareas} y localizaremos allí el programa. Cuando lo pongamos en marcha y aparecerá una pantalla parecida a esta.:



Para generar la documentación tenemos que pulsar el botón

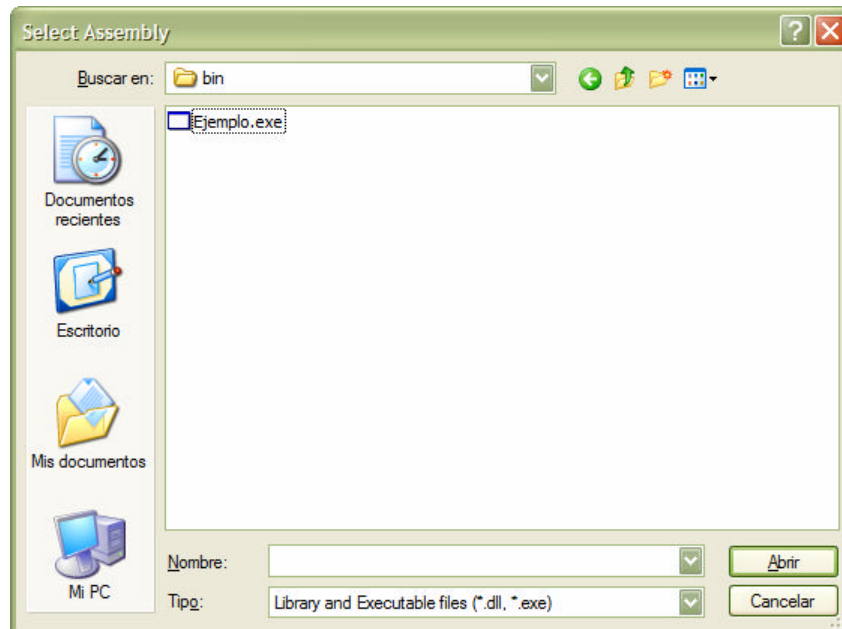


Se abre la ventana siguiente

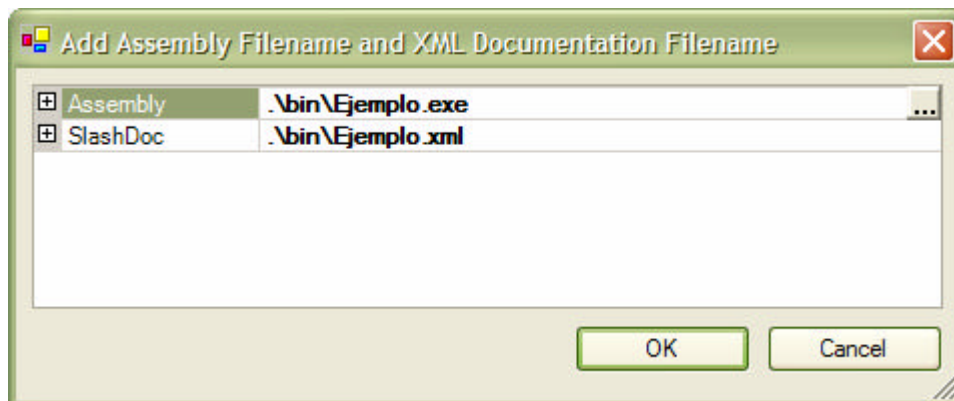


Pulsaremos el (pequeño) botón que esta a la derecha de Assembly y que contiene tres puntos suspensivos

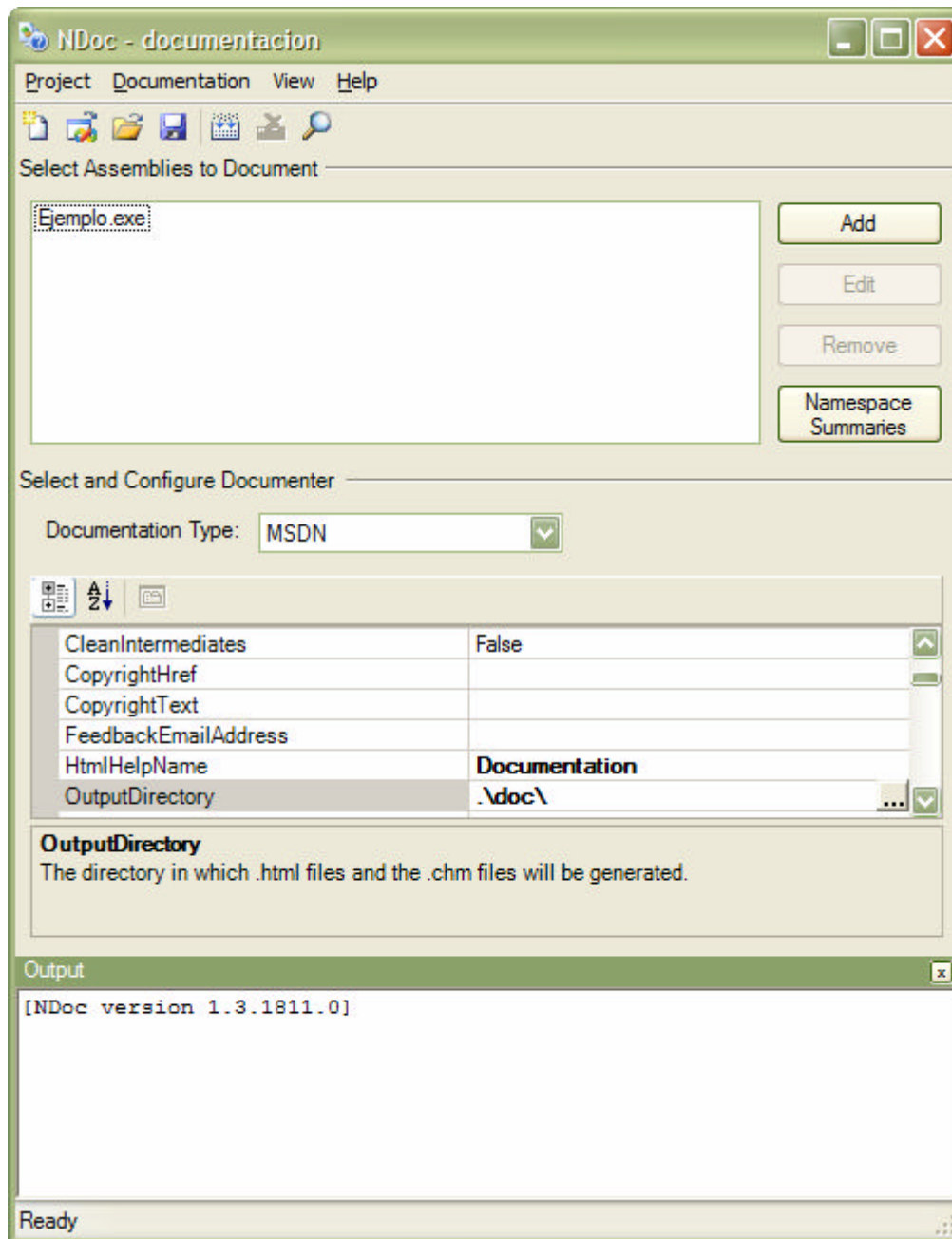
Se abrirá la ventana estándar de dialogo común de abrir fichero



Seleccionaremos el directorio [BIN] de nuestra aplicación, donde está el ejecutable, y seleccionaremos el fichero ejecutable [exe, o dll] pulsando después el botón {abrir}. En la imagen de este documento el ejecutable se llama {Ejemplo.exe}



Pulsaremos a continuación el botón OK, para volver a la pantalla principal, y en ella, estará ya nuestro fichero ejecutable.



Solo queda por hacer dos cosas:

La primera es guardar en un fichero [Menú -> Project-> Save] toda la información necesaria para que el programa trabaje, de forma que sea mucho más fácil volverlo a utilizar cada vez que queramos volver a generar la documentación de este proyecto. Por ejemplo, por haber realizado algún cambio en alguna clase.

El fichero que se genera tiene la extensión {ndoc} y es conveniente guardarlo (dejarlo) en el mismo directorio donde está el código del programa.

La segunda es generar la documentación de ayuda.

Para iniciar el proceso pulsaremos el botón correspondiente de la {barra de tareas} o bien el [menú -> documentation -> Build]. Cuando termine el proceso, se habrá generado un nuevo

directorio llamado [doc] que contendrá todas las páginas HTML de la ayuda y un fichero de ayuda compilado en formato [chm]

Es evidente que el directorio donde se genera y deposita la documentación se puede cambiar. Si te fijas en la imagen anterior veras entre las opciones de la ventana central una con el nombre [OutputDirectory] y a la derecha la cadena [\\doc\\] que indica que la documentación se situará en un directorio llamado [doc] que estará dentro del directorio raíz del proceso (dicho de otra forma, será un directorio hermano de los directorios [bin] y [obj]).

También podemos cambiar el nombre del fichero de ayuda compilado [chm]. En la imagen anterior se aprecia la entrada [HtmlHelpName] que contiene la cadena [documentation].-precisamente {documentation.chm}, es el nombre del fichero de ayuda generado.

Y eso es todo. Después de haber seguido estos pasos tenemos una bonita documentación generada automáticamente.

Convenciones básicas para la documentación

Los siguientes puntos son técnicas de comentarios recomendadas.

- Cuando modifique el código, mantenga siempre actualizados los comentarios circundantes.
- Al principio de cada rutina, resulta útil hacer comentarios estándar, que indiquen el propósito de la rutina, las suposiciones y las limitaciones.
- Las funciones tendrán como mínimo la siguiente documentación, en formato Texto o XML :

```
' /** *****  
' Función ...: <nombreFuncion()>  
' -----  
' Propósito .....:  
'     Que hace la función(no como lo hace)  
' Campos miembros  
' * Lecturas ..: Campos miembro de la clase cuyo valor  
'                se lee y se emplea en la función  
' * Escrituras : Campos miembros de la clase cuyo valor  
'                se cambia por esta función  
' Valor de Retorno .....:  
'     El valor de retorno de la función  
' Observación .....:  
'     Otros comentarios útiles  
' -----  
' */
```

- Procurar no añadir comentarios al final de una línea de código, porque lo hacen más difícil de leer. Sin embargo, los comentarios de final de línea sí son apropiados al anotar declaraciones de variables. En este caso, alinee todos los comentarios de final de línea en la misma posición de tabulación.
- Evite los comentarios recargados, como las líneas enteras de asteriscos. En su lugar, utilice espacios para separar los comentarios y el código.

- Evite rodear un bloque de comentarios con un marco tipográfico. Puede resultar agradable, pero es difícil de mantener.
- Antes de la implementación, quite todos los comentarios temporales o innecesarios, para evitar cualquier confusión en la futura fase de mantenimiento.
- Si necesita realizar comentarios para explicar una sección de código compleja, examine el código para decidir si debería volver a escribirlo. Siempre que sea posible, no documente un código malo, vuelva a escribirlo. Aunque, por regla general, no debe sacrificarse el rendimiento para hacer un código más simple para el usuario, es indispensable un equilibrio entre rendimiento y mantenibilidad.
- Use frases completas cuando escriba comentarios. Los comentarios deben aclarar el código, no añadirle ambigüedad.
- Vaya comentando al mismo tiempo que programa, porque probablemente no tenga tiempo de hacerlo más tarde. Por otro lado, aunque tuviera oportunidad de revisar el código que ha escrito, lo que parece obvio hoy es posible que seis semanas después no lo sea.
- Evite comentarios superfluos o inapropiados, como comentarios divertidos al margen.
- Use los comentarios para explicar el propósito del código. No los use como si fueran traducciones interlineales.
- Comente cualquier cosa que no sea legible de forma obvia en el código.
- Para evitar problemas recurrentes, haga siempre comentarios al depurar errores y solucionar problemas de codificación, especialmente cuando trabaje en equipo.
- Haga comentarios en el código que esté formado por bucles o bifurcaciones lógicas. Se trata en estos casos de áreas clave que ayudarán a los lectores del código fuente.
- Realice los comentarios en un estilo uniforme, respetando una puntuación y estructura coherentes a lo largo de toda la aplicación.
- Separe los comentarios de sus delimitadores mediante espacios. Si respeta estas normas, los comentarios serán más claros y fáciles de localizar si trabaja sin indicaciones de color.

Etiquetas Recomendadas por Microsoft

<c>	<para>	<see>1
<code>	<param>1	<seealso>1
<example>	<paramref>1	<summary>
<exception>1	<permission>1	<value>
<include>1	<remarks>	
<list>	<returns>	

(1). El compilador comprueba la **Sintaxis**..

Se puede incluir cualquier elemento del tipo XML, la única restricción es que los comentarios tienen que estar correctamente estructurados basándose en los estándares del W3 referentes a la estructuración de un archivo XML <http://www.w3.org/TR/REC-xml>

Aún cuando puedes incluir cualquier etiqueta que consideres pertinente, (p.e. <pre></pre>), es recomendable utilizar las recomendadas por Microsoft, esto es para lograr una mayor

adaptación y fácil entendimiento entre los diferentes desarrolladores, ya que al final de cuentas representan un estándar de documentación.

Aquí se presenta una tabla con las etiquetas recomendadas que la mayoría de los generadores de documentación como NDoc reconoce como validas, de igual manera se indica en donde pueden ser utilizadas.

Etiqueta	Clase	Estructura	Interfase	Enumeración	Delegado	Constante	Propiedad	Método
<summary>	X	X	X	X	X	X	X	X
<remarks>	X	X	X	X	X	X	X	X
<para>	X	X	X	X	X	X	X	X
<list>	X	X	X	X	X	X	X	X
<example>	X	X	X	X	X	X	X	X
<code>	X	X	X	X	X	X	X	X
<param>							X	X
<paramref>							X	X
<returns>								X
<see>	X	X	X	X	X	X	X	X
<seealso>	X	X	X	X	X	X	X	X
<exception>							X	X
<permission>						X	X	X
<value>							X	

Delimitadores para etiquetas de documentación

La utilización de etiquetas de documentación requiere delimitadores, que indican al compilador el inicio y el final de un comentario de documentación. Puede utilizar los siguientes tipos de delimitadores con las etiquetas de documentación XML:

- Para trabajar con C# [///] (tres barras inclinadas)
- Para trabajar con VB .NE ["] (tres comillas simples)

Nota El IDE de Visual Studio tiene una función llamada Edición de comentarios automática, que inserta automáticamente las etiquetas <summary> y </summary> y sitúa el cursor entre ambas después de que haya escrito el delimitador /// en el Editor de código. Obtenga acceso a esta función desde el cuadro de diálogo Formato, C#, Editor de texto, Opciones de las páginas de propiedades del proyecto.

El complemento [VbCommenter] permite insertar automáticamente las etiquetas <summary> y </summary> y sitúa el cursor entre ambas después de que haya escrito el delimitador ["] en el Editor de código. Para tener modificar su comportamiento se hace a través del [Menú -> Herramientas -> VbCommenter Options]

Etiqueta <c>

Sintaxis:

<c>text</c>

Donde:

text

Texto que se desea marcar como código.

Comentarios

La etiqueta <c> proporciona un modo de indicar que el texto de una descripción se debería marcar como código. Utilice <code> para marcar varias líneas como código.

Ejemplo:

```
''' <summary>
'''   <c> UnMetodo </c> es un método de la clase <c> Ejemplo </c>
''' </summary>
''' -----
Public Class Ejemplo

    Public Sub UnMetodo(ByVal unNumero As Integer)
        ' este método hace alguna cosa
    End Sub

End Class ' / Ejemplo
```

Etiqueta `<code>`

Sintaxis:

`<code>` content `</code>`

Donde:

content

Texto que se desea marcar como código.

Comentarios

La etiqueta `<code>` proporciona un modo de marcar varias líneas como código.

Todo lo que esta contenido dentro de la etiqueta `<code></code>` es considerado como texto en forma de código y normalmente se utiliza dentro de una etiqueta `<example>`

Ejemplo:

Vea el tema `<example>` para obtener un ejemplo sobre la utilización de la etiqueta `<code>`.

Etiqueta <example>

Sintaxis:

<example> description </example>

Donde:

description

Descripción del ejemplo de código.

Comentarios

La etiqueta <example> permite especificar un ejemplo de cómo utilizar un método u otro miembro de una biblioteca. Normalmente, esto implicaría el uso de la etiqueta <code>.

Ejemplo:

```
''' -----
''' <summary>
'''     Este metodo devuelve el valor cero,
'''     un valor dificil de calcular y obtener
''' </summary>
''' <returns><see cref="Int32" /> (System.Int32)</returns>
'''
'''     <example>
'''     Este ejemplo muestra como llamar
'''     a la función <c>ObtenerCero</c>
'''     <code>
'''
'''         Public Shared Sub main()
'''
'''             ' instanciar una variable
'''             Dim unNumero As Integer
'''
'''             ' obtener el valor cero
'''             unNumero = ObtenerCero()
'''
'''             ' hacer lo que sea con ese valor
'''             End Sub
'''
'''     </code>
'''     </example>
''' -----
Public Shared Function ObtenerCero() As Integer
    Return 0
End Function
```

Etiqueta <exception>

Sintaxis:

```
<exception cref = "member" > description </exception>
```

Donde:

cref = "member" :

Referencia a una excepción disponible desde el entorno de compilación actual. El compilador comprueba si la excepción dada existe y traduce member al nombre del elemento canónico en el resultado XML. member debe aparecer entre comillas dobles (" ").

description

Descripción.

Comentarios

La etiqueta <exception> permite especificar las excepciones que se pueden iniciar. Esta etiqueta se aplica a una definición de método.

Ejemplo:

```
''' -----
''' <summary>
'''     Este metodo devuelve el valor cero,
'''     un valor dificil de calcular y obtener
''' </summary>
'''
''' <exception cref="System.ApplicationException">
'''     Este es un error generico que ocurre cuando algo va mal.
''' </exception>
''' <exception cref="System.Exception">
'''     Este es un error generico que ocurre cuando algo va mal.
'''     la funcion no discrimina los errores, si ocurre alguno
'''     lo lanza hacia
''' </exception>

''' <returns><see cref="Int32" /> (System.Int32)</returns>
''' -----
Public Shared Function ObtenerCero() As Integer
    Return 0
End Function
```

Etiqueta <include>

Sintaxis:

```
<include file='filename' path='tagpath[@name="id"]' />
```

Donde:

filename

Nombre del archivo que contiene la documentación. El nombre de archivo se puede completar con una ruta de acceso.

Ponga filename entre comillas simples (' ').

tagpath

Ruta de acceso de las etiquetas de filename que conduce a la etiqueta name. Ponga la ruta de acceso entre comillas simples (' ').

name

Especificador de nombre en la etiqueta que precede a los comentarios; name poseerá un id.

id

Identificador para la etiqueta que precede a los comentarios. Ponga el id. entre comillas dobles (" ").

Comentarios

La etiqueta <include> permite hacer referencia a comentarios colocados en otro archivo que describen los tipos y miembros del código fuente. Ésta es una alternativa al método habitual de colocar los comentarios de la documentación directamente en el archivo de código fuente.

La etiqueta <include> utiliza la sintaxis XPath de XML. Consulte la documentación de XPath para conocer diversos modos de personalizar el uso de <include>.

Ejemplo:

Este ejemplo utiliza varios archivos. El primer archivo, que utiliza <include>, se muestra a continuación:

El siguiente ejemplo esta copiado tal cual de la documentación de Microsoft para C#

```
// xml_include_tag.cs
// compile with: /doc:xml_include_tag.xml
/// <include file='xml_include_tag.doc'
//          path='MyDocs/MyMembers[@name="test"]/*' />
class Test
{
    public static void Main()
    {
    }
}

/// <include file='xml_include_tag.doc'
//          path='MyDocs/MyMembers[@name="test2"]/*' />
class Test2
{
    public void Test()
    {
    }
}
```


El segundo archivo, xml_include_tag.doc, contiene los siguientes comentarios de documentación:

```
<MyDocs>

  <MyMembers name="test">
    <summary>
      The summary for this type.
    </summary>
  </MyMembers>

  <MyMembers name="test2">
    <summary>
      The summary for this other type.
    </summary>
  </MyMembers>

</MyDocs>
```

Resultado del programa

```
<?xml version="1.0"?>
<doc>
  <assembly>
    <name>t2</name>
  </assembly>
  <members>
    <member name="T:Test">
      <summary>
        The summary for this type.
      </summary>
    </member>
    <member name="T:Test2">
      <summary>
        The summary for this other type.
      </summary>
    </member>
  </members>
</doc>
```

Etiqueta <list>

Sintaxis:

```
<list type="bullet" | "number" | "table">
  <listheader>
    <term>¡Error! Referencia de hipervínculo no válida.</term>
    <description>¡Error! Referencia de hipervínculo no
válida.</description>
  </listheader>
  <item>
    <term>¡Error! Referencia de hipervínculo no válida.</term>
    <description>¡Error! Referencia de hipervínculo no
válida.</description>
  </item>
</list>
```

Donde:

term

Término que se define en text.

description

Elemento de una lista numerada o con viñetas, o definición de un término.

Comentarios

El bloque <listheader> se utiliza para definir la fila de encabezado de una tabla o de una lista de definiciones. Cuando se define una tabla, sólo es necesario suministrar una entrada para un término en el encabezado.

Cada elemento de la lista se especifica con un bloque <item>. Cuando se crea una lista de definiciones, se deberán especificar tanto term como text. Sin embargo, para una tabla, lista con viñetas o lista numerada, sólo es necesario suministrar una entrada para text.

Una lista o una tabla pueden tener tantos bloques <item> como sean necesarios.

Ejemplo:

La etiqueta <list> Permite crear listas numeradas en viñetas o tablas

```
<List type="bullet">      Lista en viñetas
<List type="numeric">     Lista numérica
<List type="table">       Crea una tabla
```

Para especificar los elementos que contendrá la lista o la tabla se especifica la etiqueta <item></item> especificando el contenido del elemento en la etiqueta <description></description>

```
'''<summary>
'''  <para>
'''    Proporciona las propiedades y métodos necesarios
'''    Agregar, Actualizar y Eliminar un cliente.
'''  </para>
'''  <para>
'''    Cuando utilice esta clase asegúrese de que
'''    utilizara todas las propiedades expuestas por la misma
'''  </para>
```

```
''' <para> Tipos de Cliente
''' <list type="bullet">
''' <item>
''' <description>Vip</description>
''' </item>
''' <item>
''' <description>Cancelados</description>
''' </item>
''' <item>
''' <description>Inactivos</description>
''' </item>
''' </list>
''' </para>
'''</summary>
```

Resultado

Proporciona las propiedades y métodos necesarios
Agregar, Actualizar y Eliminar un cliente.

Cuando utilice esta clase asegúrese de que
utilizara todas las propiedades expuestas por la misma
Tipos de Clientes

- Vip
- Cancelados
- Inactivos

Cuando se desee crear una tabla se puede especificar el encabezado utilizando la etiqueta `<listheader>` que al igual que `<item>` utiliza la etiqueta `<description>` para especificar el contenido.

```
''' <para> Tipos de Cliente
''' <list type=" table ">
''' <listheader>
''' <description>Tipos de Cliente</description>
''' </listheader>
''' <item>
''' <description>Vip</description>
''' </item>
''' <item>
''' <description>Cancelados</description>
''' </item>
''' <item>
''' <description>Inactivos</description>
''' </item>
''' </list>
''' </para>
```

Resultado

Cuando utilice esta clase asegúrese de que utilizara todas las propiedades expuestas por la misma
Tipos de Clientes

Descripcion
Vip
Cancelados
Inactivos

Las etiquetas <item> y <listheader> cuentan con la etiqueta <term> la cual sirve para poder crear una tabla o lista a manera de poder especificar una columna que contenga la especificación de la información descrita.

```
''' <list type="table">
''' <listheader>
''' <term>Tipo</term>
''' <description>Descripción</description>
''' </listheader>
''' <item>
''' <term>A</term>
''' <description>Vip</description>
''' </item>
''' <item>
''' <term>B</term>
''' <description>Cancelados</description>
''' </item>
''' <item>
''' <term>C</term>
''' <description>Inactivos</description>
''' </item>
''' </list>
```

Resultado

Tipo	Descripcion
A	Vip
B	Cancelados
C	Inactivos

Etiqueta <para>

Sintaxis:

<para> content </para>

Donde:

content

Texto del párrafo.

Comentarios

Ayuda a separar la sección que se esté escribiendo en párrafos, con lo cual se logra mayor claridad en nuestra redacción.

La etiqueta <para> se utiliza dentro de otra etiqueta, tal como <summary>, <remarks> o <returns>, y permite dar una estructura al texto.

Ejemplo:

Vea <summary> para obtener un ejemplo del uso de <para>.

Etiqueta <param>

Sintaxis:

<param name='name'>description</param>

Donde:

name
Nombre de un parámetro de método. Ponga el nombre entre comillas simples (' ').
description
Descripción del parámetro.

Comentarios

Esta etiqueta describe los parámetros que requiere una determinada función, Para utilizar esta etiqueta es necesario especificar sus atributos **name** el cual especifica el nombre del parámetro y **description** mediante el cual proporcionamos la descripción del parámetro.

La etiqueta <param> se usa para describir parámetros. Cuando se utiliza, el compilador comprueba si el parámetro existe y si todos los parámetros están descritos en la documentación. Si la comprobación no tiene éxito, el compilador emite una advertencia.

El texto para la etiqueta <param> se mostrará en IntelliSense, el Examinador de objetos y en el Informe Web de comentario de código.

Ejemplo:

```
''' -----  
''' <summary>  
'''     Este metodo hace alguna cosa  
''' </summary>  
'''  
''' <param name='unNumero'>  
'''     Aqui va descrito el cometido del parametro  
'''     Este número es muy importante para la ejecucion del proceso.  
'''     Value Type: <see cref="Int32" >(System.Int32) </see>  
''' </param>  
''' -----  
Public Sub UnMetodo(ByVal unNumero As Integer)  
    ' este metodo hace alguna cosa  
End Sub
```

Etiqueta <paramref>

Sintaxis:

<paramref name="name"/>

Donde:

name

Nombre del parámetro al que hay que hacer referencia. Ponga el nombre entre comillas dobles (" ").

Comentarios

Cuando necesitemos hacer referencia a los parámetros que recibe alguna función podemos utilizar la etiqueta <paramref>, en su propiedad **name** es necesario especificar el nombre del parámetro al cual estamos haciendo referencia.

La etiqueta <paramref> proporciona un modo de indicar que una palabra es un parámetro. El archivo XML se puede procesar de manera que aplique formato a este parámetro de algún modo diferente.

Ejemplo:

```
''' -----
Public Class Ejemplo
    ''' -----
    ''' <summary>
    '''     Este metodo hace alguna cosa
    ''' </summary>
    '''
    ''' <param name="unNumero">
    '''     Aqui va descrito el cometido del parametro
    '''     Este número es muy importante para la ejecucion del proceso.
    '''     Value Type: <see cref="Int32" >(System.Int32) </see>
    ''' </param>
    ''' <remarks>
    '''     UnMetodo es un procedimiento de la clase Ejemplo
    '''     El parametro <paramref name="unNumero"/>
    '''     es un numero no muy grande.
    ''' </remarks>
    ''' -----
    Public Sub UnMetodo(ByVal unNumero As Integer)
        ' este metodo hace alguna cosa
    End Sub

    . . . . .

    '''<remarks>El valor predeterminado para el parámetro
    '''<paramref name="unNumero"></paramref> es de 20 unidades
    '''</remarks>
    ' -> (aqui una funcion que falta)

End Class ' / Ejemplo
```

Etiqueta <permission>

Sintaxis:

```
<permission cref="member"> description </permission>
```

Donde:

cref = "member"

Referencia a un miembro o campo al cual se puede llamar desde el entorno de compilación actual. El compilador comprueba si el elemento de código dado existe y traduce member al nombre de elemento canónico en el resultado XML. member debe aparecer entre comillas dobles (" ").

description

Descripción del acceso al miembro.

Comentarios

Especifica los permisos necesarios que se deben de cumplir para poder utilizar el método. La etiqueta <permission> permite documentar el acceso de un miembro. El conjunto de permisos System.Security.PermissionSet permite especificar el acceso a un miembro.

Ejemplo:

```
''' -----  
''' <summary>  
'''     Este metodo devuelve el valor cero,  
'''     un valor dificil de calcular y obtener  
''' </summary>  
'''  
''' <permission cref="System.Security.Permissions.FileIOPermission">  
'''     Debe de tener permisos de escritura en la  
'''     ruta especificada  
''' </permission>  
''' <permission cref="System.Security.PermissionSet">  
'''     Todo el mundo puede acceder a este metodo  
''' </permission>  
''' -----  
Public Shared Function ObtenerCero() As Integer  
    Return 0  
End Function
```

Resultado

```
Requeriments  
    NET Framework Security  
        System.Security.Permissions.FileIOPermission debe tener permisos  
        de escritura en la ruta especificada  
        System.Security.PermissionSet Todo el mundo puede acceder a  
        este metodo
```


Etiqueta <remarks>

Sintaxis:

<remarks> description </remarks>

Donde:

description

Descripción del miembro.

Comentarios

Normalmente es utilizado en conjunto con la etiqueta <Summary>, y su objetivo es proporcionar documentación a manera de información complementaria con lo cual ayuda a ser un poco más específico en cuanto a consideraciones que deben de tomarse en cuenta. La etiqueta <remarks> se utiliza para agregar información sobre un tipo, de modo que completa la información especificada con <summary>. Esta información se muestra en el Examinador de objetos y en el Informe Web de comentario de código.

Ejemplo:

```
''' -----
'''<summary>
'''    Proporciona las propiedades y métodos necesarios
'''    Agregar, Actualizar y Eliminar un cliente.
'''</summary>
'''<remarks>
'''    Recuerde utilizar esta clase solo cuando
'''    necesite modificar toda la información
'''    referente al cliente
'''</remarks>
''' -----

Public Class Ejemplo
    . . . . .
End Class ' / Ejemplo
```

Etiqueta <returns>

Sintaxis:

<returns> description </returns>

Donde:

description

Descripción del valor devuelto.

Comentarios

Especifica el valor de retorno de una función

La etiqueta <returns> se debe utilizar en el comentario de una declaración de método para describir el valor devuelto.

Ejemplo:

```
''' <returns>
'''   <para>
'''       Devuelve una <see cref="String">cadena </see>
'''       que contiene un texto SOAP que
'''       representa el objeto serializado
'''   </para>
''' </returns>

''' <returns>
'''   <para>
'''       Devuelve un valor <see cref="Boolean"> Booleano </see>
'''       que indica lo siguiente:
'''   </para>
'''   <para> Valor TRUE   si se pudo cargar el fichero</para>
'''   <para> Valor FALSE no se pudo cargar
'''   </para>
''' </returns>
```

Etiqueta <see>

Sintaxis:

```
<see cref="member"/>
```

Donde:

```
cref = "member"
```

Referencia a un miembro o campo al cual se puede llamar desde el entorno de compilación actual. El compilador comprueba si el elemento de código dado existe y pasa member al nombre de elemento en el resultado XML. member debe aparecer entre comillas dobles (" ").

Comentarios

Esta etiqueta nos permite poner la referencia hacia un elemento de programación en nuestro código (namespace, clase), de manera que cuando el usuario de un clic en el elemento resaltado se podrá obtener más información del miembro solicitado. Comunmente puede ser utilizado para obtener más información del tipo al cual estamos haciendo referencia.

La etiqueta <see> permite especificar un vínculo desde dentro del texto. Utilice <seealso> para indicar el texto que desea que aparezca en una sección. [Vea también].

El atributo cref se puede asociar a cualquier etiqueta para proporcionar una referencia a un elemento de código. El compilador comprobará si existe ese elemento de código. Si la comprobación no tiene éxito, el compilador emite una advertencia. El compilador también respeta cualquier instrucción using cuando busca un tipo descrito en el atributo cref.

Ejemplo:

Vea < returns > para obtener un ejemplo del uso de <see>.

Etiqueta <seealso>

Sintaxis:

<seealso cref="member"/>

Donde:

cref = "member"

Referencia a un miembro o campo al cual se puede llamar desde el entorno de compilación actual. El compilador comprueba si el elemento de código dado existe y pasa member al nombre de elemento en el resultado XML. member debe aparecer entre comillas dobles (" ").

Comentarios

La etiqueta <seealso> permite especificar el texto que se desea que aparezca en una sección [Vea también]. Utilice <see> para especificar un vínculo desde dentro del texto.

Es común, por ejemplo, en el caso de clases heredadas o submiembros el hacer referencia a sus clases base para poder, ya sea ampliar un tema o consultar el detalle de alguna implementación en particular, la etiqueta <seealso> nos ayuda a hacer referencia a una clase o namespace ya existente, el cual es especificado en su parámetro cref

<seealso> se utiliza (normalmente) dentro de la etiqueta <summary>

El atributo cref se puede asociar a cualquier etiqueta para proporcionar una referencia a un elemento de código. El compilador comprobará si existe ese elemento de código. Si la comprobación no tiene éxito, el compilador emite una advertencia. El compilador también respeta cualquier instrucción using cuando busca un tipo descrito en el atributo cref.

Ejemplo:

Vea <summary> para obtener un ejemplo del uso de <seealso>.

```
''' <returns>
'''     <para>
'''         Devuelve una <see cref="String">cadena </see>
'''         que contiene un texto SOAP que
'''         representa el objeto serializado
'''     </para>
'''     <seealso cref="MyClass.Main"/>
''' </returns>
```

Etiqueta <summary>

Sintaxis:

<summary> description </summary>

Donde:

description
Resumen del objeto.

Comentarios

Proporciona una descripción del tipo del miembro que estamos documentando, procure ser lo más específico posible para que se entienda de manera sencilla el propósito que se logra obtener.

La etiqueta <summary> se utiliza para describir un tipo o un miembro de tipo.

Utilice <remarks> para suministrar información adicional a una descripción de tipo.

El texto para la etiqueta <summary> es la única fuente de información sobre el tipo en IntelliSense, y también se muestra en el Explorador de objetos y en el Informe Web de comentario de código.

Ejemplo:

```
'''<summary> clase muy interesante que dibuja clientes</summary>
'''<remarks>
'''   <para>
'''       Proporciona las propiedades y métodos necesarios
'''       Agregar, Actualizar y Eliminar un cliente.
'''   </para>
'''   <para>
'''       Cuando utilice esta clase asegúrese de que utilizara
'''       todas las propiedades expuestas por la misma
'''   </para>
'''<remarks>
''' -----
Public Class Ejemplo
    ''' -----
    ''' <summary>
    '''     Este metodo devuelve el valor cero,
    '''     un valor dificil de calcular y obtener
    ''' </summary>
    ''' -----
    Public Shared Function ObtenerCero() As Integer
        Return 0
    End Function
End Class ' / Ejemplo
```

Etiqueta <value>

Sintaxis:

<value> property-description </value>

Donde:

property-description
Descripción de la propiedad.

Comentarios

En el caso de las propiedades, éste elemento proporciona una descripción del valor que se está estableciendo o recuperando.

La etiqueta <value> permite describir una propiedad. Tenga en cuenta que al agregar una propiedad a través de un asistente para código en el entorno de desarrollo de Visual Studio .NET, se agregará una etiqueta <summary> para la nueva propiedad. A continuación, se debe agregar manualmente una etiqueta <value> que describa el valor que representa la propiedad.

Ejemplo:

```
''' -----  
''' Class.Method: RemitenteVO.Email.Get  
''' <summary> Correo electronico del remitente </summary>  
''' <value>  
'''     <para>  
'''         Permite acceder al campo miembro que contiene el  
'''         correo electronico del que envia el mensaje  
'''     </para>  
''' </value>  
'''  
''' <exception cref="System.ApplicationException">  
'''     Es la excepcion general del sistema  
'''     que ocurre cuando algo va mal  
''' </exception>  
'''  
''' <seealso cref="System.String"> (System.String) </seealso>  
''' -----  
Public Property Email() As String  
    Get  
        Return m_Email  
    End Get  
    Set(ByVal Value As String)  
        m_Email = Value  
    End Set  
End Property
```

Ejemplos

Ejemplo de documentación de un Namespace

Ejemplo

```
''' -----  
''' <summary>  
'''     Espacio de nombres jms32  
''' </summary>  
''' <remarks>  
'''     Corresponde al codigo generado por el  
'''     famoso programador Joaquin Medina Serrano  
''' </remarks>  
''' -----  
Namespace jms32
```

Imagen de la documentación obtenida

Aunque el código este en la clase no se refleja en la documentación.

Ejemplo de documentación de una clase

Ejemplo

```
''' -----
''' Project:  Ejemplo
''' Class:    jms32.RemitenteVO
'''
'''<summary>
'''    Contiene informacion sobre el remitente de un mensaje
'''</summary>
'''
'''<remarks>
'''    <para>
'''        Esta clase es un ejemplo ficticio que contiene la
'''        informacion y los metodos necesarios para manejar la
'''        informacion asociada al remitente de un mensaje.
'''    </para>
'''    <para>
'''        Se emplea formando parte de del mensaje que se envia
'''        a traves de un socket
'''    </para>
'''    <para>
'''        Es una clase serializable, porque su informacion se
'''        serializará en formato XML SOAP para enviarse a traves
'''        de la red
'''    </para>
'''
'''    <para>
'''        Caracteristicas de la clase:
'''    <list type="bullet">
'''        <item> Lenguaje...: Visual Basic .NET (2003) </item>
'''        <item> Tipo.....: (CLASE) </item>
'''        <item> Herencia ..: (Ninguna) </item>
'''        <item> Autor.....: Joaquin Medina Serrano </item>
'''        <item> Correo.....:
'''            <a href="mailto:joaquin@medina.name">
'''                joaquin@medina.name
'''            </a>
'''        </item>
'''        <item>
'''            Copyright...: Todo el mundo puede acceder a
'''            esta clase es de uso libre
'''        </item>
'''    </list>
'''    </para>
'''
'''    <para>
'''    <b> Historia de Revisiones: </b>
'''    <list type="bullet">
'''        <item>17/01/2005 - Creacion</item>
'''        <item>17/01/2005 - Modificacion de la documentacion </item>
'''    </list>
'''    </para>
'''    </remarks>
'''    -----
'''
'''<Serializable()> Public Class RemitenteVO
```


Imagen de la documentación obtenida

An NDoc Documented Class Library

RemitenteVO Class

Contiene informacion sobre el remitente de un mensaje
For a list of all members of this type, see [RemitenteVO Members](#).

System.Object
jms32.RemitenteVO

```
public class RemitenteVO
```

Thread Safety

Public static (**Shared** in Visual Basic) members of this type are safe for multithreaded operations. Instance members are **not** guaranteed to be thread-safe.

Remarks

Esta clase es un ejemplo ficticio que contiene la informacion y los metodos necesarios para manejar la informacion asociada al remitente de un mensaje.

Se emplea formando parte de del mensaje que se envia a traves de un socket

Es una clase serializable, porque su informacion se serializar en formato XML SOAP para enviarse a traves de la red

Caracteristicas de la clase:

- ◆ Lenguaje...: Visual Basic .NET (2003)
- ◆ Tipo.....: (CLASE)
- ◆ Herencia ..: (Ninguna)
- ◆ Autor.....: Joaquin Medina Serrano
- ◆ Correo.....: joaquin@medina.name
- ◆ Copyright...: Todo el mundo puede acceder a esta clase es de uso libre

Historia de Revisiones:

- ◆ 17/01/2005 - Creacion
- ◆ 17/01/2005 - Modificacion de la documentacion

Requirements















Namespace: [jms32](#)

Assembly: Ejemplo (in Ejemplo.exe)

See Also

[RemitenteVO Members](#) | [jms32 Namespace](#)

En el apartado Miembros de la clase se genera la siguiente documentación

An NDoc Documented Class Library	
RemitenteVO Members	
RemitenteVO overview	
Public Static Methods	
 S main	Punto de entrada de la aplicacion
Public Instance Constructors	
 RemitenteVO Constructor	Constructor que carga los dos campos de la clase
Public Instance Fields	
 m_Nombre	Contiene el nombre del remitente
Public Instance Properties	
 Email	Correo electronico del remitente
 Nombre	Nombre del remitente
 Separador	Caracter [/] separador de contenidosd
Public Instance Methods	
 Copia	Esta funcion es una funcion copia
 Equals (inherited from Object)	
 GetHashCode (inherited from Object)	
 GetType (inherited from Object)	
 ToString	Funcion ToString [Overrides]
Public Instance Events	
 DatosCopiados	
Protected Instance Methods	
 Finalize (inherited from Object)	
 MemberwiseClone (inherited from Object)	
See Also	
RemitenteVO Class ims32 Namespace	

Ejemplo de documentación de una constante

¡Ojo! Solo Tiene sentido documentar una constante o un campo de la clase que sea público

Ejemplo

```
''' -----  
''' <summary> Contiene el nombre del remitente </summary>  
''' <remarks>  
'''     <para>  
'''         Es una variable de  
'''         <see cref="String"> (System.String) </see>  
'''         cadena que guarda el nombre de la persona  
'''         que envia el mensaje  
'''     </para>  
''' </remarks>  
''' -----  
Public m_Nombre As String = String.Empty
```

Imagen de la documentación obtenida



Ejemplo de documentación de un constructor

Ejemplo

```
''' -----  
''' Class.Method:      jms32.RemitenteVO.New  
''' -----  
''' <summary>  
'''     Constructor que carga los dos campos de la clase  
''' </summary>  
'''  
''' <param name='nombre'>  
'''     Es una cadena que contiene el nombre del  
'''     remitente del mensaje  
''' </param>  
''' <param name='email'>  
'''     Es una cadena que contiene el correo del  
'''     remitente del mensaje  
''' </param>  
'''  
''' <exception cref="System.ApplicationException">  
'''     Es la excepcion general del sistema que ocurre  
'''     cuando algo va mal  
''' </exception>  
''' <seealso cref="System.String"> (System.String) </seealso>  
''' -----  
Public Sub New(ByVal nombre As String, ByVal email As String)  
    m_Nombre = nombre  
    m_Email = email  
End Sub
```

Imagen de la documentación obtenida

An NDoc Documented Class Library

RemitenteVO Constructor

Constructor que carga los dos campos de la clase

```
public RemitenteVO(  
    string nombre,  
    string email  
);
```

Parameters

nombre

Es una cadena que contiene el nombre del remitente del mensaje

email

Es una cadena que contiene el correo del remitente del mensaje

Exceptions

Exception Type	Condition
ApplicationException	Es la excepcion general del sistema que ocurre cuando algo va mal

See Also

[RemitenteVO Class](#) | [jms32 Namespace](#) | **String**

Ejemplo de documentación de un método

Ejemplo

```
'''
''' Class.Method: RemitenteVO.Copia
''' <summary>
'''     Esta funcion es una funcion copia
''' </summary>
'''
''' <remarks>
'''     <para>
'''         se emplea para copiar otro objeto en este. se recibe a
'''         traves del parametro un objeto y esta funcion copia en los
'''         campos miembros de la clase los valores que estan en los
'''         campos miembro del objeto recibido a traves del parametro,
'''         de forma que al final tengo en este objeto una copia
'''         identica del objeto recibido
'''     </para>
'''     <para> Campos Miembro Leidos .....: Ninguno </para>
'''     <para> Campos Miembro Modificados ...:
'''     <list type="table">
'''         <listheader>
'''             <term> Campo </term>
'''             <description> Descripcion </description>
'''         </listheader>
'''         <item>
'''             <term> <c>m_Nombre</c> </term>
'''             <description> El nombre del que envia el mensaje
'''             </description>
'''         </item>
'''         <item>
'''             <term> <c>m_Email</c> </term>
'''             <description>
'''                 El correo electronico del que envia el mensaje
'''             </description>
'''         </item>
'''         <item>
'''             <term> <c>m_Separador</c> </term>
'''             <description> el carcacter separador </description>
'''         </item>
'''     </list>
''' </para>
'''
'''     <para>
'''     Historia de Revisiones:
'''     <list type="bullet">
'''         <item> Autor.: Joaquin Medina Serrano </item>
'''         <item> Correo.: joaquin@medina.name </item>
'''         <item>17/01/2005 - Creacion</item>
'''         <item>
'''             17/01/2005 - Modificacion de la documentacion
'''         </item>
'''     </list>
''' </para>
'''
''' </remarks>
```

```

'''
''' <param name="p_oRemitenteVO">
'''     Es el objeto origen, el objeto que pretendo copiar.
''' </param>
'''
''' <exception cref="System.ApplicationException">
'''     Esta excepcion se lanza cuando ocurre algun error en
'''     la copia de los campos miembro, se supone que el objeto
'''     recibido es del tipo null (nothing) y no puedo copiar nada
''' </exception>
''' <exception cref="System.ApplicationException">
'''     Es la excepcion general del sistema que ocurre cuando
'''     algo va mal
''' </exception>
'''
''' <returns>
'''     <para> Devuelve el objeto que se recibio a traves
'''           del parametro
'''     </para>
'''     Tipo del valor devuelto:
'''     <see cref="RemitenteVO"> (jms32.RemitenteVO) </see>
''' </returns>
'''
''' <example>
'''     Este es un ejemplo que muestra como se usa la funcion
'''     <code>
'''         Dim o1 As jms32.RemitenteVO
'''         o1 = New jms32.RemitenteVO( _
'''             "Joaquin", "Emilio@telefonica.net")
'''
'''         Dim o2 As jms32.RemitenteVO
'''         o2.Copia(o1)
'''         Console.WriteLine(o2.ToString)
'''     </code>
''' </example>
'''
''' <seealso cref="DatosCopiados">
'''     el evento Datos Copiados
''' </seealso>
'''
''' <permission cref="System.Security.PermissionSet">
'''     Todo el mundo puede acceder a este metodo, es de uso libre
'''     Este parametro esta copiado literalmente de la
'''     doumentacion MSDN y no se como se utiliza exactamente
''' </permission>
''' -----
Public Function Copia(ByRef p_oRemitenteVO As jms32.RemitenteVO) _
    As RemitenteVO
    Try
        ' procedo a copiar en mis campos los valores
        ' de los campos del objeto pasado por parametro
        Me.m_Nombre = p_oRemitenteVO.m_Nombre
        Me.m_Email = p_oRemitenteVO.m_Email
        Me.m_Separador = p_oRemitenteVO.m_Separador
    Catch ex As ArgumentNullException
        ' esta excepcion esta solo como ejemplo de documentacion
        Throw ex
    End Try

```

```
' disparar el evento
RaiseEvent DatosCopiados(Me, New System.EventArgs)

' devolver el objeto recibido
Return p_oRemitenteVO
End Function
```

Imagen de la documentación obtenida

An NDoc Documented Class Library

RemitenteVO.Copia Method

Esta funcion es una funcion copia

```
public RemitenteVO Copia(
    ref RemitenteVO p_oRemitenteVO
);
```

Parameters

p_oRemitenteVO
Es el objeto origen, el objeto que pretendo copiar.

Return Value

Devuelve el objeto que se recibio a traves del parametro
Tipo del valor devuelto: **RemitenteVO**

Remarks

se emplea para copiar otro objeto en este. se recibe a traves del parametro un objeto y esta funion copia en los campos miembros de la clase los valores que estan en los campos miembro del objeto recibido a traves del parametro, de forma que al final tengo en este objeto una copia identica del objeto recibido

Campos Miembro Leidos: Ninguno

Campos Miembro Modificados ...:

Campo	Descripcion
<i>m_Nombre</i>	El nombre del que envia el mensaje
<i>m_Email</i>	el correo electronico del que envia el mensjae
<i>m_Separador</i>	el carcacter separador

Historia de Revisiones:

- ◆ Autor.: Joaquin Medina Serrano
- ◆ Correo.: joaquin@medina.name
- ◆ 17/01/2005 - Creacion
- ◆ 17/01/2005 - Modificacion de la documentacion

Exceptions

Exception Type	Condition
ApplicationException	Esta excepcion se lanza cuando ocurre algun error en la copia de los campos miembro, se supone que el objeto recibido es del tipo null (nothing) y no puedo copiar nada
ApplicationException	Es la excepcion general del sistema que ocurre cuando algo va mal

Example

Este es un ejemplo que muestra como se usa la funcion

```
Dim o1 As jms32.RemitenteVO
o1 = New jms32.RemitenteVO("Joaquin", "Emilio@telefonica.net")

Dim o2 As jms32.RemitenteVO
o2.Copia(o1)
Console.WriteLine(o2.ToString)
```

Requirements

.NET Framework Security:

- ♦ Todo el mundo puede acceder a este metodo, es de uso libre Este parametro esta copiado literalmente de la doumentacion MSDN y no se como se utiliza exactamente

See Also

[RemitenteVO Class](#) | [jms32 Namespace](#) | **DatosCopiados**

Ejemplo de documentación de una propiedad

Ejemplo

```
''' -----  
''' Class.Method: RemitenteVO.Email.Get  
''' <summary> Correo electronico del remitente </summary>  
''' <value>  
'''     <para>  
'''         Permite acceder al campo miembro que contiene el  
'''         correo electronico del que envia el mensaje  
'''     </para>  
''' </value>  
'''  
''' <exception cref="System.ApplicationException">  
'''     Es la excepcion general del sistema  
'''     que ocurre cuando algo va mal  
''' </exception>  
'''  
''' <seealso cref="System.String"> (System.String) </seealso>  
''' -----  
Public Property Email() As String  
    Get  
        Return m_Email  
    End Get  
    Set(ByVal Value As String)  
        m_Email = Value  
    End Set  
End Property
```

Imagen de la documentación obtenida

An NDoc Documented Class Library
RemitenteVO.Email Property

Correo electronico del remitente

```
public string Email {get; set;}
```

Property Value

Permite acceder al campo miembro que contiene el correo electronico del que envia el mensaje

Exceptions

Exception Type	Condition
ApplicationException	Es la excepcion general del sistema que ocurre cuando algo va mal

See Also

[RemitenteVO Class](#) | [ims32 Namespace](#) | **String**

Resultados del ejemplo

La clase y el archivo XML generado puedes descargártelos en este enlace :-))

Referencia bibliográfica

- Librería MSDN
 - Referencia del programador de C#
 - Documentación XML
 - Etiquetas recomendadas para comentarios de documentación
 - Procesar comentarios de documentación
- Pagina del El Guille Titulo.: Documentación de código, Autor: Misael Monterroca
mmonterroca@neo-mx.com

Historial del documento

- | | | | |
|---|-----------|---------------------|------------------------------------|
| • | Martes | 18 de enero de 2005 | Creación y publicación |
| • | Domingo | 23 de enero de 2005 | Modifico la introducción |
| • | Miércoles | 20 de abril de 2005 | Corrijo algunos errores de formato |

Fecha de impresión

- 20/04/2005 12:05:02