

ETIQUETAS DE DOCUMENTACIÓN XML



12/12/2008

Etiquetas recomendadas para documentación XML

Aunque el programador puede utilizar las etiquetas que estime oportunas en sus comentarios de documentación y darles el significado que quiera, Microsoft recomienda usar un juego de etiquetas concreto con significados concretos para escribir ciertos tipos de información común. Con ello se obtendría un conjunto básico de etiquetas que cualquier herramienta que trabaje con documentación XML pueda estar preparada para procesar (como se verá más adelante, el propio Visual Studio.NET da ciertos usos específicos a la información así documentada)

Etiquetas de documentación XML

En los siguientes epígrafes se explican estas etiquetas recomendadas agrupándolas según su utilidad. Todas son opcionales, y no incluirlas sólo tiene el efecto de que en la documentación resultante no se generarían las secciones correspondientes a ellas.

ETIQUETAS DE USO GENÉRICO

Hay una serie de etiquetas predefinidas que pueden colocarse, en cualquier orden, precediendo las definiciones de miembros en los ficheros fuente. Estas etiquetas, junto al significado recomendado para su contenido, son las explicadas a continuación:

- **<summary>**: Su contenido se utiliza para indicar un resumen sobre el significado del elemento al que precede. Cada vez que en VS.NET se use el operador “.” para acceder a algún miembro de un objeto o tipo se usará esta información para mostrar sobre la pantalla del editor de texto un resumen acerca de su utilidad.
- **<remarks>**: Su contenido indica una explicación detallada sobre el elemento al que precede. Se recomienda usar **<remarks>** para dar una explicación detallada de los tipos de datos y **<summary>** para dar una resumida de cada uno de sus miembros.
- **<example>**: Su contenido es un ejemplo sobre cómo usar el elemento al que precede.
- **<seealso>**: Se usa para indicar un elemento cuya documentación guarda alguna relación con la del elemento al que precede. No tiene contenido y el nombre del elemento al que se remite se indica en su atributo **cref**, por lo que el compilador comprobará si existe. Para indicar múltiples documentaciones relativas a un cierto elemento basta usar una etiqueta **<seealso>** por cada una.
- **<permission>**: Se utiliza para indicar qué permiso necesita un elemento para poder funcionar. En su contenido se indica una descripción del mismo, y su atributo **cref** suele usarse para indicar el tipo que representa a ese permiso. Por ejemplo:

```
/// <permission cref="System.Security.Permissions.FileIOPermission">  
/// Necesita permiso de lectura/escritura en el directorio C:\Datos  
/// </permission>
```

Como con **<seealso>**, si un miembro ha de disponer varios tipos de permisos puede documentarse su definición con tantas etiquetas **<permission>** como sea necesario.

ETIQUETAS RELATIVAS A MÉTODOS

Además de las etiquetas uso general ya vistas, en las definiciones de métodos se pueden usar las siguientes etiquetas recomendadas adicionales para describir sus parámetros y valor de retorno:

- **<param>**: Permite documentar el significado de un parámetro de un método. En su propiedad `name` se indica el nombre del parámetro a documentar y en su contenido se describe su utilidad. Por ejemplo:

```
/// <summary> Método que muestra un texto por pantalla </summary>
/// <param name="texto"> Texto a mostrar </param>
bool MuestraTexto(string texto)
{...}
```

Al generarse la documentación se comprueba si el método documentado dispone de algún parámetro con el nombre indicado en `name` y, como ocurre con **cref**, si no fuese así se generaría un mensaje de aviso informando de ello.

- **<paramref>**: Se usa para referenciar a parámetros de métodos. No tiene contenido y el nombre del parámetro referenciado se indica en su atributo `name`. Por ejemplo:

```
/// <summary>
///     Método que muestra por pantalla un texto con un determinado color
/// </summary>
/// <param name="texto"> Texto a mostrar </param>
/// <param name="color">
///     Color con el que mostrar el <paramref name="texto"/> indicado
/// </param>
bool MuestraTexto(string texto, Color color)
{...}
```

Nuevamente, al generarse la documentación se comprobará si realmente el parámetro referenciado existe en la definición del método documentado y si no es así se genera un mensaje de aviso informando de ello.

- **<returns>**: Permite documentar el significado del valor de retorno de un método, indicando como contenido suyo una descripción sobre el mismo. Por ejemplo:

```
/// <summary>
///     Método que muestra por pantalla un texto con un determinado color
/// </summary>
/// <param name="texto"> Texto a mostrar </param>
/// <param name="color">
///     Color con el que mostrar el <paramref name="texto"/> indicado
/// </param>
/// <returns> Indica si el método se ha ejecutado con éxito o no </returns>
bool MuestraTexto(string texto, Color color)
{...}
```

ETIQUETAS RELATIVAS A PROPIEDADES

El uso más habitual de una propiedad consiste en controlar la forma en que se accede a un campo privado, por lo que esta se comporta como si almacenase un valor. Mediante el contenido de la etiqueta `<value>` es posible describir el significado de ese valor:

```
private int edad;
/// <summary>
/// Almacena la edad de una persona. Si se le asigna una edad menor que 0 la
/// sustituye por 0.
/// </summary>
/// <value> Edad de la persona representada </value>
public int Edad
{
    set { edad = (value < 0) ? 0 : value; }
    get { return edad; }
}
```

ETIQUETAS RELATIVAS A EXCEPCIONES

Para documentar el significado de un tipo definido como excepción puede incluirse un resumen sobre el mismo como contenido de una etiqueta de documentación `<exception>` que preceda a su definición. El atributo `cref` de ésta suele usarse para indicar la clase de la que deriva la excepción definida. Por ejemplo:

```
/// <exception cref="System.Exception">
///     Excepción de ejemplo creada por Josan
/// </exception>

class JosanExcepción : Exception
{...}
```

ETIQUETAS RELATIVAS A FORMATO

Para mejorarla forma de expresar el contenido de las etiquetas de documentación que se utilicen es posible incluir en ellas las siguientes etiquetas de formato:

- **<see>**: Se utiliza para indicar hipervínculos a otros elementos de la documentación generada. Es una etiqueta sin contenido en la que el destino del enlace es la documentación del miembro cuyo nombre completo se indica en su atributo `cref`. Ese nombre es también el texto que las hojas de estilo suelen mostrar para representar por pantalla el enlace, por lo que los usos de esta etiqueta suelen ser de la forma:

```
/// <summary>
/// Muestra por la salida estándar el mensaje ¡Hola!
/// Si no sabe como se escribe en pantalla puede consultar
/// la documentación del método <see cref="System.Console.WriteLine"/>
/// </summary>
public static void Saluda()
{
    Console.WriteLine("¡Hola!");
}
```

Nótese que la diferencia de `<see>` y `<seealso>` es que la primera se usa para indicar enlaces en medio de textos mientras que la otra se usa para indicar enlaces que se deseen incluir en una sección aparte tipo “Véase también”.

- **`<code>` y `<c>`**: Ambas etiquetas se usan para delimitar textos han de ser considerarse fragmentos de código fuente. La diferencia entre ellas es que `<code>` se recomienda usar para fragmentos multilínea y `<c>` para los de una única línea; y que las hojas de estilo mostrarán el contenido de las etiquetas `<code>` respetando su espaciado y el de las etiquetas `<c>` sin respetarlo y tratando cualquier aparición consecutiva de varios caracteres de espaciado como si fuesen un único espacio en blanco.

En general, `<code>` suele usarse dentro de etiquetas `<example>` para mostrar fragmentos de códigos de ejemplo, mientras que `<c>` suele usarse para hacer referencia a elementos puntales de los códigos fuente. Por ejemplo:

```
/// <example>
/// Este ejemplo muestra cómo llamar al método <c>Cumple()</c> de esta clase:
/// <code>
///     Persona p = new Persona(...);
///     p.Cumple();
/// </code>
/// </example>
```

- **`<para>`**: Se usa para delimitar párrafos dentro del texto contenido en otras etiquetas, considerándose que el contenido de cada etiqueta `<para>` forma parte de un párrafo distinto. Generalmente se usa dentro de etiquetas `<remarks>`, ya que son las que suelen necesitar párrafos al tener un contenido más largo. Por ejemplo:

```
/// <remarks>
///     <para>
///         Primer párrafo de la descripción del miembro...
///     </para>
///     <para>
///         Segundo párrafo de la descripción del miembro...
///     </para>
/// </remarks>
```

- **`<list>`**: Se utiliza para incluir listas y tablas como contenido de otras etiquetas. Todo uso de esta etiqueta debería incluir un atributo **type** que indique el tipo de estructura se desea definir según tome uno de los siguientes valores:
 - ✓ **bullet**: Indica que se trata de una lista no numerada
 - ✓ **number**: Indica que se trata de una lista numerada
 - ✓ **table**: Indica que se trata de una tabla

El contenido de `<list>` dependerá del tipo de estructura representado en cada caso:

- Si se trata de una lista normal –ya sea numerada o no numerada- su contenido será una etiqueta **<item>** por cada elemento de la lista, y cada etiqueta de este tipo contendrá una etiqueta **<description>** con el texto correspondiente a ese elemento. Por ejemplo:

```
/// <list type="bullet">
///   <item>
///     <description>
///       Elemento 1
///     </description>
///   </item>
///   <item>
///     <description>
///       Elemento 2
///     </description>
///   </item>
/// </list>
```

Si se tratase de una tabla, su contenido sería similar al de las listas normales sólo que por cada fila se incluiría una etiqueta **<item>** y dentro de ésta se incluiría una etiqueta **<description>** por cada columna de esa fila. Opcionalmente se podría incluir también una etiqueta **~<listheader>** antes de las **~<item>** donde se indicaría el texto de la cabecera de la tabla. Esta etiqueta se usa igual que las etiquetas **~<item>**: incluirá una etiqueta **~<description>** por cada columna.

- Por último, si fuese una lista de definiciones cada **<item>** contendría una primera etiqueta **<term>** con el nombre del elemento a definir y otra segunda etiqueta **<description>** con su definición. Opcionalmente también podría incluirse una etiqueta **<listheader>** con la cabecera de la lista. Por ejemplo:

```
/// <list type="bullet">
///   <item>
///     <term>
///       Término 1
///     </term>
///     <description>
///       Descripción de término 1
///     </description>
///   </item>
///   <item>
///     <term>
///       Término 2
///     </term>
///     <description>
///       Descripción de término 2
///     </description>
///   </item>
/// </list>
```

REFERENCIA

La principal y más exhaustiva fuente de información sobre C# es la "C# Language Specification", originalmente escrita por Anders Hejlsberg, Scott Wiltamuth y Peter Golde. Incluye la especificación completa del lenguaje, y su última versión siempre puede descargarse gratuitamente de <http://www.msdn.microsoft.com/vcsharp/language>.

Este extracto de información ha sido obtenida a partir del libro "El lenguaje de programación C#". Lo pueden encontrar en:

<http://www.josanguapo.com/>

"El lenguaje de programación C#" es un libro escrito en castellano que explica detalladamente cómo se programa en C#, mientras que "C#: El nuevo lenguaje de Internet" es documentación que forma parte de un seminario sobre C# en la que se explica en forma de tutorial cómo programar aplicaciones de ventanas y servicios web usando C# y VS.NET. Puede descargarlos **gratuitamente** pulsando sobre los siguientes enlaces:

- El lenguaje de programación C# [\[Formato Word \(819 KB, v2.0\)\]](#)
- El lenguaje de programación C# [\[Formato Word \(658 KB, v1.0\)\]](#) [\[Formato HTML v1.0\]](#)
- [C#: El nuevo lenguaje de Internet \(285 KB, v1.0\)](#)

La versión original de "El lenguaje de programación C#" realizada por mi es la que se encuentra en formato Word. La traducción a HTML esta realizada por Daniel Rodríguez, WebMaster de [Programación en castellano](#)