Anexo Documentación de Código

Descargar estos apuntes en pdf o html

Índice

Introducción

La creación de documentación, aunque a menudo es una tarea aburrida y pesada, es muy importante sobre todo en un enfoque de programación orientada a componentes en tanto que los componentes desarrollados muchas veces van a ser reutilizados por otros, e incluso para el propio creador del componente puede resultar de inestimable ayuda si en el futuro tiene que modificarlo o usarlo y no recuerda exactamente cómo lo implementó.

Para facilitar la pesada tarea de escribir la documentación, el compilador de C# es capaz de generarla automáticamente a partir de los comentarios que el programador escriba en los ficheros de código fuente. Así se evita trabajar con dos tipos de documentos por separado (fuentes y documentación) que deban actualizarse simultáneamente para evitar inconsistencias. El compilador genera la documentación en formato XML con la idea de que sea fácilmente legible para cualquier aplicación. Más información en la web oficial

Sintaxis general

Los comentarios de documentación XML se escriben como comentarios normales de una línea pero con las peculiaridad de que su primer carácter ha de ser siempre / y de que su contenido ha de estar escrito en XML ya que será insertado por el compilador en el fichero XML de documentación que genere.
Por tanto, son comentarios de la forma:

```
        /// <textoXML>
```

Estos comentarios han de preceder las definiciones de los elementos a documentar. Estos elementos sólo pueden ser definiciones de miembros, ya sean tipos de datos o los propios miembros de estos.

Etiquetas recomendadas para documentación XML

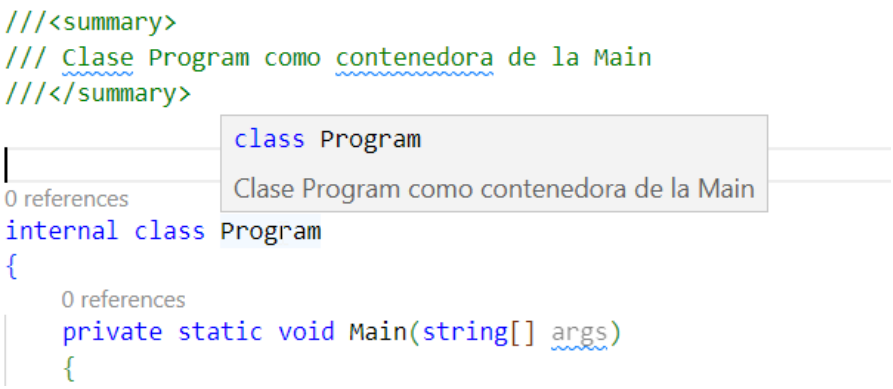
Aunque el programador puede utilizar las etiquetas que estime oportunas en sus comentarios de documentación y darles el significado que quiera, Microsoft recomienda usar un juego de etiquetas concreto con significados concretos para escribir ciertos tipos de información común. Con ello se obtendría un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

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 Visual Studio 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, normalmente siempre va junto con la etiqueta `<code>` que permite mostrar las líneas con formato de código

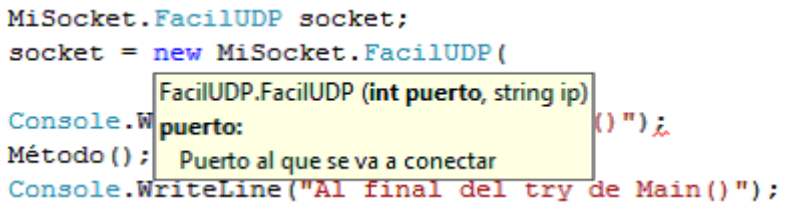
```
///<summary>
///<example>
///Esto muestra como incrementar un entero
///<code>
///    var index = 5;
///    index++;
///</code>
///</example>
///</summary>
```

Etiquetas relativas a métodos

Además de las etiquetas de 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:

Se mostraría como se ve en la siguiente imagen:



- `<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){...}
```

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

- `<returns>:` Permite documentar el significado del valor de retorno de un método, su contenido deberá ser descripción sobre el uso del valor de retorno. 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
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 es posible describir el significado de ese valor:

```
private int edad
/// <summary>
///     Almacena la edad de una persona. Si se 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 control de entrada de números no permitidos
/// </exception
class NumerosNoPermitidosExcepcion : Exception
{ }
```