Código Vainilla

Prácticas Ágiles para Desarrollo de Software

C# BUENAS PRACTICAS

La siguiente entrada será para hablar de algunas cosas que tenemos que tener en cuenta a la hora de programar en C# y las cuales son consideradas buenas practicas en este lenguage.

los temas que veremos a continuación son los siguiente:

- Definición de variables por default.
- Declaración de variables.
- Comparaciones en C#.
- Métodos y propiedades.
- Abrir y cerrar conexiones.
- Declaraciones de clases.
- Indentación, espaciado y Agrupación de código.
- Comentarios y notas en código (documentación).
- Manejo de excepciones.
- Logueo de excepciones.

Definición de variables por default.

> *string var1= String.empty;*
> *int var2 = 0;*

> *boolean = true;*

La declaración de las variables locales de un método, deberá realizarse en el bloque superior de éste. Todas las variables deben estar declaradas en el mismo lugar y deben ser inicializadas para limpiar su valor predeterminado.

Se debe elegir antes de empezar a codificar si usar Pascal Casing, camel Casing, Snake_Case

Declaración de variables

Declaración de variables C# tipos definidos, ya que muchas veces usamos los tipos definidos en las clases de Framework de .NET

> *int var1;*
> *string var2;*
> *object var3;*

No usar

Int, String, Object, Boolean.

Utilizar prefijos como “is” o similares al nombrar variables de tipo bool.

> *private bool isFinished;*

Comparaciones en C#

Es preferible utilizar las clases para manejo de string como se define a continuación:

> *string.Equals(var)*
>
> *string.Contrains(Var)*

* Siempre que se comparan se recomienda hacerlos en mayúsculas o minúsculas.

Métodos y propiedades

Evitar métodos y propiedades públicas, a menos que sea estrictamente necesario accederlas desde afuera de la clase. Utilizar el keyword internal si estos miembros deben ser accedidos desde dentro del mismo assembly.

Abrir y cerrar conexiones

Al abrir conexiones a bases de datos, Sockets, Streams, etc. siempre cerrarlos en los bloques finally. Esto asegurará que aún ante la eventualidad de una excepción, estos accesos serán cerrados. Se puede usar el Using en caso de no tener que capturar una excepción en particular dentro del catch.

Declaración de clases

Utilizar Pascal Casing para declarar nombres de clases.
Utilizar el prefijo “I” con Pascal Casing para nomenclar Interfaces (Ejemplo: IUser).

Indentación, espaciado y agrupación de código

- Utilice TAB de 4 posiciones para indentar código. No utilice espacios.
- Los comentarios deben estar en el mismo nivel que el código, es decir que han de utilizar el mismo nivel de indentación.
- Las llaves de apertura y cierre deben estar en el mismo nivel que el código fuera de éstas.
- Las llaves deben estar en una línea independiente y no en la misma línea que if, for, etc.
- Utilizar un único espacio antes y después de cada operador, así como también entre términos utilizados durante la invocación de métodos.
- Mantener las variables de miembro, propiedades y métodos en la parte superior del archivo, y miembros públicos en la parte inferior del mismo.
- Utilizar #region para agrupar piezas de código relacionadas. Si se utiliza la agrupación adecuada mediante #region, una clase o página deberá lucir solamente con las regiones cuando todas las definiciones se contraigan. Mantener las variables privadas, las propiedades y métodos privados en la parte superior del archivo y los miembros públicos en la parte inferior.

Comentarios y notas de código.

- Corroborar ortografía, gramática y semántica de los comentarios, asegurándose de que la puntuación es utilizada correctamente.

- Agregar la documentación del encabezado de cada método (utilizando ///), especificando correctamente:
  - El propósito del método
  - Nombre, Tipo de datos y Contenido de cada parámetro recibido
  - El propósito de la información retornada
  - Esto es particularmente útil para generar documentación de forma automática, a partir de estos comentarios.

Manejo de excepciones

Nunca “capturar una excepción y no hacer nada”. Si se oculta una excepción, nunca se sabrá si la excepción sucedido. Muchos desarrolladores utilizan este método para ignorar errores no significativos. Trate de evitar las excepciones comprobando todas las condiciones de error mediante programación. En el peor de los casos, usted debe registrar la excepción en un log y continuar.

Logueo de errores

Siempre se debe loguear cada uno de los errores para poder identificar cualquier comportamiento irregular en la App.
No escribir bloques TRY-CATCH muy grandes.

Post By Felix Malerva (2 Posts) CONNECT

11 junio, 2016 Felix Malerva NET Buenas Practicas, C#