**C#语言的编码的规范**

1 文档介绍  
1.1 文档目的  
规范C#语言的编码，尽量统一项目组的编码风格，提高代码可读性。

1.2 文档范围  
仅针对C#语言描述其编码规范。

1.3 读者对象  
项目组所有人员。

1.4 参考文档  
《MSDN》

1.5 术语与缩写解释  
缩写、术语 解释   
    
    
  

2 概述

2.1 规范制定原则  
方便代码的交流和维护。  
不影响编码的效率，不与大众习惯冲突。  
使代码更美观、阅读更方便。  
使代码的逻辑更清晰、更易于理解。

2.2 术语定义

2.2.1 Pascal 大小写  
将标识符的首字母和后面连接的每个单词的首字母都大写。可以对三字符或更多字符的标识符使用Pascal 大小写。例如：BackColor

2.2.2 Camel 大小写  
标识符的首字母小写，而每个后面连接的单词的首字母都大写。例如： backColor

2.3 文件命名组织  
2.3.1 文件命名  
文件名遵从Pascal命名法，无特殊情况，扩展名小写。form类前不用加form或frm等。  
使用统一的文件扩展名：类名.cs

3 代码外观

3.1 列宽  
代码列宽控制在150字符左右，即避免不出横行滚动条。

3.2 换行  
当表达式超出或即将超出规定的列宽，遵循以下规则进行换行  
  在逗号后换行；  
  在操作符前换行；  
  规则1优先于规则2。  
当以上规则会导致代码混乱的时候自己采取更灵活的换行规则。

3.3 缩进   
缩进应该是4个空格，不使用Tab。

3.4 空行  
空行是为了将逻辑上相关联的代码分块，以便提高代码的可阅读性。  
在以下情况下使用两个空行  
  接口和类的定义之间。  
  枚举和类的定义之间。  
  类与类的定义之间。  
在以下情况下使用一个空行  
  方法与方法、属性与属性之间。  
  方法中变量声明与语句之间。  
  方法与方法之间。  
  方法中不同的逻辑块之间。  
  方法中的返回语句与其他的语句之间。  
  属性与方法、属性与字段、方法与字段之间。  
  注释与它注释的语句间不空行，但与其他的语句间空一行。

3.5 空格  
在以下情况中要使用到空格  
  关键字和左括符 “(” 应该用空格隔开。如  
     while (true)  
注意：在方法名和左括符 “(” 之间不要使用空格，这样有助于辨认代码中的方法与关键字。  
  多个参数用逗号隔开，每个逗号后都应加一个空格。  
  除了 . 之外，所有的二元操作符都应用空格与它们的操作数隔开。一元操作符、++ 及 -- 与操作数间不需要空格。如  
     a += c + d;  
     a = (a  +  b) / (c \* d);  
     while (d++ = s++)  
         {  
              n++;  
         }  
     PrintSize(“size is “ + size + “\n”);  
  4、 语句中的表达式之间用空格隔开。如  
     for (expr1; expr2; expr3)

3.6 括号()  
  左括号“(”不要紧靠关键字，中间用一个空格隔开。  
  左括号“(”与方法名之间不要添加任何空格。  
  没有必要的话不要在返回语句中使用()。如  
     if (condition)  
     Array.Remove(1)  
     return 1

3.7 花括号{}  
  左花括号“{”放于关键字或方法名的下一行并与之对齐。如  
     if (condition)  
     {  
     }  
     public int Add(int x, int y)  
     {  
     }

  左花括号“{”要与相应的右花括号“}”对齐。  
  通常情况下左花括号“{”单独成行，不与任何语句并列一行。  
  if、while、do语句后一定要使用{}，即使{}号中为空或只有一条语句。如  
     if (somevalue == 1)  
     {  
         somevalue = 2;  
     }

  5、 右花括号 “}” 后建议加一个注释以便于方便的找到与之相应的右花括号 “{”。如  
     while (1)  
     {  
          if (valid)  
          {  
          } // if valid  
          else  
          {  
          } // not valid  
     } // end forever

4 程序注释  
4.1 注释概述  
  修改代码时，总是使代码周围的注释保持最新。  
  在每个例程的开始，提供标准的注释样本以指示例程的用途、假设和限制很有帮助。注释样本应该是解释它为什么存在和可以做什么的简短介绍。  
  避免在代码行的末尾添加注释；行尾注释使代码更难阅读。不过在批注变量声明时，行尾注释是合适的；在这种情况下，将所有行尾注释在公共制表位处对齐。   
  避免杂乱的注释，如一整行星号。而是应该使用空白将注释同代码分开。   
  避免在块注释的周围加上印刷框。这样看起来可能很漂亮，但是难于维护。  
  在部署发布之前，移除所有临时或无关的注释，以避免在日后的维护工作中产生混乱。  
  如果需要用注释来解释复杂的代码节，请检查此代码以确定是否应该重写它。尽一切可能不注释难以理解的代码，而应该重写它。尽管一般不应该为了使代码更简单以便于人们使用而牺牲性能，但必须保持性能和可维护性之间的平衡。  
  在编写注释时使用完整的句子。注释应该阐明代码，而不应该增加多义性。   
  在编写代码时就注释，因为以后很可能没有时间这样做。另外，如果有机会复查已编写的代码，在今天看来很明显的东西六周以后或许就不明显了。  
  避免多余的或不适当的注释，如幽默的不主要的备注。  
  使用注释来解释代码的意图。它们不应作为代码的联机翻译。   
  注释代码中不十分明显的任何内容。  
  为了防止问题反复出现，对错误修复和解决方法代码总是使用注释，尤其是在团队环境中。  
  对由循环和逻辑分支组成的代码使用注释。这些是帮助源代码读者的主要方面。   
  在整个应用程序中，使用具有一致的标点和结构的统一样式来构造注释。   
  用空白将注释同注释分隔符分开。在没有颜色提示的情况下查看注释时，这样做会使注释很明显且容易被找到。  
  在所有的代码修改处加上修改标识的注释。  
  为了是层次清晰，建议在闭合的右花括号后注释该闭合所对应的起点。  
namespace Langchao.Procument.Web   
{  
} // namespace Langchao.Procument.Web  
4.2 文档型注释  
该类注释采用.Net已定义好的XML标签来标记，在声明接口、类、方法、属性、字段都应该使用该类注释，以便代码完成后直接生成代码文档，让别人更好的了解代码的实现和接口。如  
///<summary>MyMethod is a method in the MyClass class.  
///<para>Here's how you could make a second paragraph in a description.  
///<see cref="System.Console.WriteLine"/>   
///for information about output statements.  
///</para>  
///<seealso cref="MyClass.Main"/>  
///</summary>  
public void MyMethod(int Int1)  
{  
}  
标签的用法与作用在4.5节有详细描述。  
4.3 类c注释  
该类注释用于  
  不再使用的代码。  
  临时测试屏蔽某些代码。  
用法  
/\*  
[修改标识]  
[修改原因]  
...(the source code )  
\*/

4.4 单行注释  
该类注释用于  
  方法内的代码注释。如变量的声明、代码或代码段的解释。注释示例：  
// 注释语句  
private int number;  
  方法内变量的声明或花括号后的注释, 注释示例：  
if ( 1 == 1)    // always true  
{   
    statement;   
} // always true

4.5 注释标签  
这是生成生成代码文档的基础，所以在编写公共类时必需熟习本节内容。  
标签 用法 作用   
<c> <c>text <></c>  
   text 希望将其指示为代码的文本。 为您提供了一种将说明中的文本标记为代码的方法。使用 <code> <vclrfcode.htm> 将多行指示为代码   
<para> <para>content <></para> content段落文本。 用于诸如 <remarks> 或 <returns> <vclrfreturns.htm> 等标记内，使您得以将结构添加到文本中。   
<param> <param name='name <>'>description <></param> name 为方法参数名。将此名称用单引号括起来 (' ')。 应当用于方法声明的注释中，以描述方法的一个参数。   
<paramref>   <paramref name="name <>"/>  name  要引用的参数名。将此名称用双引号括起来 (" ")。 <paramref> 标记为您提供了一种指示词为参数的方法。可以处理 XML 文件，从而用某种独特的方法格式化该参数。   
<see> <see cref <><>"member"/>  cref = "member" 对可以通过当前编译环境进行调用的成员或字段的引用。编译器检查到给定代码元素存在后，将 member 传递给输出 XML 中的元素名。必须将 member 括在双引号 (" ") 中。 使您得以从文本内指定链接。使用 <seealso> <vclrfseealso.htm> 指示希望在“请参阅”一节中出现的文本。   
<seealso> <seealso cref <><>"member"/>  cref = "member" 对可以通过当前编译环境进行调用的成员或字段的引用。编译器检查到给定代码元素存在后，将 member 传递给输出 XML 中的元素名。必须将 member 括在双引号 (" ") 中 使您得以指定希望在“请参阅”一节中出现的文本。使用 <see> <vclrfsee.htm> 从文本   
<example> <example>description <></example>  description  代码示例的说明。 使用 <example> 标记可以指定使用方法或其他库成员的示例。一般情况下，这将涉及到 <code> <vclrfcode.htm> 标记的使用。   
<code> <code>content <></code>  content 为希望将其标记为代码的文本。   记为您提供了一种将多行指示为代码的方法。使用 <c> <vclrfc.htm> 指示应将说明中的文本标记为代码   
<summary> <summary>description <></summary>  此处description 为对象的摘要。 应当用于描述类型成员。使用 <remarks> <vclrfremarks.htm> 以提供有关类型本身的信息。   
<exception> <exception cref <><>"member">description <></exception> cref = "member" 对可从当前编译环境中获取的异常的引用。编译器检查到给定异常存在后，将 member 转换为输出 XML 中的规范化元素名。必须将 member 括在双引号 (" ") 中。 description 说明。 <exception> 标记使您可以指定类能够引发的异常。