**.Net Coding Standards**

# **Document Overview**

## **Overview**

This document provides a set of guidelines and leading practices to be used during the development of .Net Applications.

# **Layout Conventions**

Below are the layout conventions, following which would make the application code more readable and maintainable.

|  |
| --- |
| * Write only one statement per line. |
| * Write only one declaration per line. |
| * If continuation lines are not indented automatically, indent them one tab stop (four spaces). |
| * Add at least one blank line between method definitions and property definitions. |
| * Use parentheses to make clauses in an expression apparent. |
| * Write method summary above the method definition, whenever a new method is added. |

# **Commenting Conventions**

This section lays down the ground rules for adding comments to the code.

|  |
| --- |
| * Place the comment on a separate line, not at the end of a line of code. |
| * Begin comment text with an uppercase letter. |
| * End comment text with a period. |
| * Insert one space between the comment delimiter (//) and the comment text. |
| * Do not create formatted blocks of asterisks around comments. |

# **String Data Type Guidelines**

String data type in C# has certain best practices to be followed which optimizes the code.

|  |
| --- |
| * Use string interpolation to concatenate short strings, as shown in the following code.   string displayName = $"{nameList[n].LastName}, {nameList[n].FirstName}";   * To append strings in loops, especially when you are working with large amounts of text,   use a StringBuilder object.  var phrase = "lalalalalalalalalalalalalalalalalalalalalalalalalalalalalala";  var manyPhrases = new StringBuilder();  for (var i = 0; i < 10000; i++)  {  manyPhrases.Append(phrase);  } |
| **Implicitly Typed Local Variables** In C#, we can either explicitly declare the type of the variable during its declaration or use the var keyword  and let the compiler determine its type implicitly. Below are the guidelines for  explicitly declaring the type of the variables   |  | | --- | | * Use implicit typing for local variables when the type of the variable is obvious from the right   side of the assignment, or when the precise type is not important. As shown in below example:   * var var1 = "This is clearly a string.";  var var2 = 27; | | * Do not use var when the type is not apparent from the right side of the assignment.   int var3 = Convert.ToInt32(Console.ReadLine());  int var4 = ExampleClass.ResultSoFar(); | | * Do not rely on the variable name to specify the type of the variable. It might not be correct.   var inputInt = Console.ReadLine();  Console.WriteLine(inputInt); | | * Avoid the use of var in place of dynamic. | | * Use implicit typing to determine the type of the loop variable in for loops. The following example   uses implicit typing in a for statement.  var phrase = "lalalalalalalalalalalalalalalalalalalalalalalalalalalalalala";  var manyPhrases = new StringBuilder();  for (var i = 0; i < 10000; i++)  {  manyPhrases.Append(phrase);  } | | * Do not use implicit typing to determine the type of the loop variable in foreach loops. The following example uses explicit typing in a foreach statement.   foreach (char ch in laugh)  {  if (ch == 'h')  Console.Write("H");  else  Console.Write(ch);  }  Console.WriteLine(); | |