Helper class to split a long word into a single one.

```
/*
* Author: Kishore Reddy
* Url: http://commonlibrarynet.codeplex.com/
* Title: CommonLibrary.NET
* Copyright: ? 2009 Kishore Reddy
* License: LGPL License
* LicenseUrl: http://commonlibrarynet.codeplex.com/license
* Description: A C# based .NET 3.5 Open-Source collection of reusable components.
* Usage: Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an “AS IS” BASIS,
* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
* See the License for the specific language governing permissions and
* limitations under the License.
*/
using System;
using System.Collections.Generic;
using System.Text;

namespace GenericCode
{
///

/// Helper class to split a long word into a single one.
/// Alternative to possibly using Regular expression.
///
public class TextSplitter
{
```

```
///

/// Determine how many times the word has to be split.
///
///  ///  ///
internal static int GetNumberOfTimesToSplit(int wordLength,
int maxCharsInWord)
{
// Validate.
if (wordLength <= maxCharsInWord) return 0; // Now calc. int
splitCount = wordLength / maxCharsInWord; int leftOver =
wordLength % maxCharsInWord; if (leftOver > 0) splitCount++;

return splitCount;
}
///

/// Split the word, N number of times.
///
/// The text to split. /// 40 chars in each word. /// ” “ ///
internal static string SplitWord(string text, int
charsPerWord, string spacer)
{
// Validate.
if (string.IsNullOrEmpty(text)) { return text; }

// Determine how many times we have to split.
int splitCount = GetNumberOfTimesToSplit(text.Length,
charsPerWord);

// Validate.
if (splitCount == 0) return text;

// Use buffer instead of string concatenation.
StringBuilder buffer = new StringBuilder();
int currentPosition = 0;

// Split N number of times.
for (int count = 1; count <= splitCount; count++) { string
```

```
word = (count < splitCount) ? text.Substring(currentPosition, charsPerWord) : text.Substring(currentPosition); buffer.Append(word); // Condition to prevent adding spacer at the end. // This is to leave the supplied text the same except for splitting ofcourse. if (count < splitCount) buffer.Append(spacer); // Move to next split start position. currentPosition += charsPerWord; } return buffer.ToString(); } ///
/// Check the single line of text for long word that exceeds the
/// maximum allowed.
/// If found, splits the word.
///

///  ///  ///
public static string CheckAndSplitText(string text, int maxCharsInWord)
{
// Validate.
if (string.IsNullOrEmpty(text)) return text;

bool isSpacerNewLine = false;
int currentPosition = 0;
int ndxSpace = GetIndexOfSpacer(text, currentPosition, ref isSpacerNewLine);

// Case 1: Single long word.
if (ndxSpace < 0 && text.Length > maxCharsInWord) return SplitWord(text, maxCharsInWord, ” “);

StringBuilder buffer = new StringBuilder();

// Now go through all the text and check word and split.
while ((currentPosition < text.Length && ndxSpace > 0))
{
//Lenght of word
int wordLength = ndxSpace – (currentPosition);
string currentWord = text.Substring(currentPosition,
```

```
wordLength);
string spacer = isSpacerNewLine ? Environment.NewLine : ” “;

if (wordLength > maxCharsInWord)
{
string splitWord = SplitWord(currentWord, maxCharsInWord, ”
“);
buffer.Append(splitWord + spacer);
}
else
{
buffer.Append(currentWord + spacer);
}

currentPosition = (isSpacerNewLine) ? ndxSpace + 2 : ndxSpace
+ 1;
ndxSpace = GetIndexOfSpacer(text, (currentPosition), ref
isSpacerNewLine);
}

// Final check.. no space found but check complete length now.
if (currentPosition < text.Length && ndxSpace < 0) { //Lenght
of word int wordLength = (text.Length) - currentPosition;
string currentWord = text.Substring(currentPosition,
wordLength); string spacer = isSpacerNewLine ?
Environment.NewLine : " "; if (wordLength > maxCharsInWord)
{
string splitWord = SplitWord(currentWord, maxCharsInWord, ”
“);
buffer.Append(splitWord);
}
else
{
buffer.Append(currentWord);
}
}
return buffer.ToString();
}
```

```
///

/// Get the index of a spacer ( space” ” or newline )
///
///  ///  ///
public static int GetIndexOfSpacer(string txt, int currentPosition, ref bool isNewLine)
{
// Take the first spacer that you find. it could be eithr
// space or newline, if space is before the newline take space
// otherwise newline.
int ndxSpace = txt.IndexOf(” “, currentPosition);
int ndxNewLine = txt.IndexOf(Environment.NewLine, currentPosition);
bool hasSpace = ndxSpace > -1;
bool hasNewLine = ndxNewLine > -1;
isNewLine = false;

// Found both space and newline.
if (hasSpace && hasNewLine)
{
if (ndxSpace < ndxNewLine) { return ndxSpace; } isNewLine = true; return ndxNewLine; } // Found space only. if (hasSpace && !hasNewLine) { return ndxSpace; } // Found newline only. if (!hasSpace && hasNewLine) { isNewLine = true; return ndxNewLine; } // no space or newline. return -1; } } }
[/csharp]
```