How to: Programmatically Exclude Paragraph Marks When Creating Ranges

Whenever you create a Range object based on a paragraph, all non-printing characters, such as paragraph marks, are included in the range. You may want to insert the text from a source paragraph into a destination paragraph. If you do not want to split the destination paragraph into separate paragraphs, then you must first remove the paragraph mark from the source paragraph. Additionally, since paragraph formatting information is stored within the paragraph mark, you might not want to include this when you insert the range into an existing paragraph.

Applies to: The information in this topic applies to document-level projects and VSTO add-in projects for Word. For more information, see Features Available by Office Application and Project Type.

The following example procedure declares two string variables, retrieves the contents of the first and second paragraphs in the active document, and then exchanges their contents. The example then demonstrates removing the paragraph marker from the range by using the M:Microsoft.Office.Interop.Word.Range.MoveEnd(System.Object@,System.Object@) method and inserting text inside the paragraph.

To control paragraph structure when inserting text

1. Create two range variables for the first and second paragraphs, and retrieve their contents using the P:Microsoft.Office.Interop.Word.Range.Text property.

   The following code example can be used in a document-level customization.

   C#

```
            Word.Range firstRange = this.Paragraphs[1].Range;
            Word.Range secondRange = this.Paragraphs[2].Range;

            string firstString = firstRange.Text;
            string secondString = secondRange.Text;
```

   The following code example can be used in an application-level VSTO Add-in. This code uses the active document.

   C#

```
            Word.Document document = this.Application.ActiveDocument;
            Word.Range firstRange = document.Paragraphs[1].Range;
            Word.Range secondRange = document.Paragraphs[2].Range;

            string firstString = firstRange.Text;
            string secondString = secondRange.Text;
```

2. Assign the P:Microsoft.Office.Interop.Word.Range.Text property, swapping the text between the two paragraphs.

C#

```
            firstRange.Text = secondString;
            secondRange.Text = firstString;
```

3. Select each range in turn and pause to display the results in a message box.

C#

```
            firstRange.Select();
            MessageBox.Show(firstRange.Text);
            secondRange.Select();
            MessageBox.Show(secondRange.Text);
```

4. Adjust `firstRange` using the M:Microsoft.Office.Interop.Word.Range.MoveEnd(System.Object@,System.Object@) method so that the paragraph marker is no longer a part of `firstRange`.

C#

```
            object charUnit = Word.WdUnits.wdCharacter;
            object move = -1;  // move left 1

            firstRange.MoveEnd(ref charUnit, ref move);
```

5. Replace the rest of the text in the first paragraph, assigning a new string to the P:Microsoft.Office.Interop.Word.Range.Text property of the range.

C#

```
            firstRange.Text = "New content for paragraph 1.";
```

6. Replace the text in `secondRange`, including the paragraph mark.

C#

```
            secondRange.Text = "New content for paragraph 2.";
```

7. Select `firstRange` and pause to display the results in a message box, and then do the same with `secondRange`.

   Since `firstRange` was redefined to exclude the paragraph mark, the original formatting of the paragraph is preserved. However, a sentence was inserted over the paragraph mark in `secondRange`, removing the separate paragraph.

C#

```
            firstRange.Select();
            MessageBox.Show(firstRange.Text);
            secondRange.Select();
            MessageBox.Show(secondRange.Text);
```

The original contents of both ranges were saved as strings, so you can restore the document to its original condition.

8. Readjust `firstRange` to include the paragraph mark by using the M:Microsoft.Office.Interop.Word.Range.MoveEnd(System.Object@,System.Object@) method for one character position.

C#

```
            move = 1;  // move right 1
            firstRange.MoveEnd(ref charUnit, ref move);
```

9. Delete `secondRange`. This restores paragraph three to its original position.

C#

```
            secondRange.Delete(ref missing, ref missing);
```

10. Restore the original paragraph text in `firstRange`.

C#

```
            firstRange.Text = firstString;
```

11. Use the M:Microsoft.Office.Interop.Word.Range.InsertAfter(System.String) method of the Range object to insert the original paragraph-two content after `firstRange`, and then select `firstRange`.

C#

```
            firstRange.InsertAfter(secondString);
            firstRange.Select();
```

Document-Level Customization Example

To control paragraph structure when inserting text in document-level customizations

1. The following example shows the complete method for a document-level customization. To use this code, run it from the `ThisDocument` class in your project.

C#

```
        private void ReplaceParagraphText()
        {
            Word.Range firstRange = this.Paragraphs[1].Range;
            Word.Range secondRange = this.Paragraphs[2].Range;

            string firstString = firstRange.Text;
            string secondString = secondRange.Text;

            firstRange.Text = secondString;
            secondRange.Text = firstString;

            firstRange.Select();
            MessageBox.Show(firstRange.Text);
            secondRange.Select();
            MessageBox.Show(secondRange.Text);

            object charUnit = Word.WdUnits.wdCharacter;
            object move = -1;  // move left 1

            firstRange.MoveEnd(ref charUnit, ref move);

            firstRange.Text = "New content for paragraph 1.";
            secondRange.Text = "New content for paragraph 2.";

            firstRange.Select();
            MessageBox.Show(firstRange.Text);
            secondRange.Select();
            MessageBox.Show(secondRange.Text);

            move = 1;  // move right 1
            firstRange.MoveEnd(ref charUnit, ref move);

            secondRange.Delete(ref missing, ref missing);

            firstRange.Text = firstString;

            firstRange.InsertAfter(secondString);
            firstRange.Select();
        }
```

VSTO Add-in Example

To control paragraph structure when inserting text in an VSTO Add-in

1. The following example shows the complete method for an VSTO Add-in. To use this code, run it from the `ThisAddIn` class in your project.

C#

```
        private void ReplaceParagraphText()
        {
            Word.Document document = this.Application.ActiveDocument;
            Word.Range firstRange = document.Paragraphs[1].Range;
            Word.Range secondRange = document.Paragraphs[2].Range;

            string firstString = firstRange.Text;
            string secondString = secondRange.Text;

            firstRange.Text = secondString;
            secondRange.Text = firstString;

            firstRange.Select();
            MessageBox.Show(firstRange.Text);
            secondRange.Select();
            MessageBox.Show(secondRange.Text);

            object charUnit = Word.WdUnits.wdCharacter;
            object move = -1;  // move left 1

            firstRange.MoveEnd(ref charUnit, ref move);

            firstRange.Text = "New content for paragraph 1.";
            secondRange.Text = "New content for paragraph 2.";

            firstRange.Select();
            MessageBox.Show(firstRange.Text);
            secondRange.Select();
            MessageBox.Show(secondRange.Text);

            move = 1;  // move right 1
            firstRange.MoveEnd(ref charUnit, ref move);

            secondRange.Delete(ref missing, ref missing);

            firstRange.Text = firstString;

            firstRange.InsertAfter(secondString);
            firstRange.Select();
        }
```

See Also

How to: Programmatically Extend Ranges in Documents
How to: Programmatically Collapse Ranges or Selections in Documents
How to: Programmatically Insert Text into Word Documents
How to: Programmatically Reset Ranges in Word Documents
How to: Programmatically Define and Select Ranges in Documents
Optional Parameters in Office Solutions