

Formatting Text

Table of Contents

- [Applying Fonts](#)
- [Named Styles](#)
- [Paragraph Formatting](#)
- [Table Formatting](#)

This article covers applying fonts, named styles, paragraph formatting and table formatting.

Applying Fonts

Using the Font class, you can apply text formatting to a CharacterRun or HTMLToWord object. To return a Font object use:

```
Document.CreateFont
```

Returns a Font object that represents the font used by the document's Normal style.

Or

```
NamedStyle.Font
```

Returns a copy of the specified named style's font.

To define a font and apply it to a paragraph, first create a Font object:

```
WordApplication wwapp = new WordApplication();
Document doc = wwapp.Create();
Font paragraphFont = doc.CreateFont();
paragraphFont.FontName = "Times New Roman";
paragraphFont.FontSize = 10;
```

Add an empty paragraph to your document:

```
//--- InsertParagraphAfter's style argument may be null.  If it
//--- is null, the Normal style will be applied to the paragraph. The
//--- Normal style is the default style used by Microsoft Word.
Paragraph para = doc.InsertParagraphAfter(null);
```

To add text to the paragraph call one of the Element class's InsertTextAfter or InsertTextBefore methods. Pass the the text to add and your Font object to the method:

```
//--- The following method takes a string and inserts it
//--- at the end of the paragraph. The method's second
//--- parameter specifies a Font object to apply to the
//--- text.
para.InsertTextAfter("This document was created with
    OfficeWriter for Word.",
    paragraphFont);
```

To get a copy of a built-in style's font, use the `NamedStyle.Font` property.

```
NamedStyle heading1Style = doc.Styles[NamedStyle.BuiltIn.Heading1];
Font heading1Font = heading1Style.Font;
```

Named Styles

The `Styles` collection contains all the named styles in the Word document. All Word documents contain a set of built-in styles. These are included in the `Styles` collection. If you use `WordWriter` to open an existing Word file, it may also contain custom named styles which will be added to the `Styles` collection.

You cannot create or modify a named style with `WordWriter`.

To return a document's `Styles` collection, use the property `Document.Styles`:

```
WordApplication wwapp = new WordApplication();
Document doc = wwapp.Create();
NamedStyle heading1Style = doc.Styles[NamedStyle.BuiltIn.Heading1];
```

To apply a style to a paragraph, pass it to `InsertParagraphAfter` or `InsertParagraphBefore` when you add the paragraph to your document:

```
Paragraph para =
    doc.InsertParagraphAfter(doc.Styles[NamedStyle.BuiltIn.BodyText]);
```

Paragraph Formatting

When you add a paragraph to a document using `Element.InsertParagraphAfter` or `Element.InsertParagraphBefore`, you can pass a `ParagraphFormatting` object to the method. A `ParagraphFormatting` object contains a set of formatting properties that can be applied to a paragraph.

To return a `ParagraphFormatting` object, use:

- `Document.CreateParagraphFormatting`

Returns a `ParagraphFormatting` object that represents the font used by the document's `BodyText` style.

Or

- `NamedStyle.ParagraphFormatting`

Returns a copy of the specified named style's paragraph formatting properties.//--- Return the paragraph formatting of the `BodyText` style.

```
WordApplication wwapp = new WordApplication();
Document doc = wwapp.Create();
ParagraphFormatting bodyTextFormatting = doc.CreateParagraphFormatting();

//--- Return a copy of the paragraph formatting of the BodyText2 Style.
WordApplication wwapp = new WordApplication();
Document doc = wwapp.Create();
ParagraphFormatting bodyText2Format =
    doc.Styles[NamedStyle.BuiltIn.BodyText2].ParagraphFormatting;
```

To assign the ParagraphFormatting object that you created to a paragraph, pass it to Element.InsertParagraphAfter or Element.InsertParagraphBefore when you create a new paragraph.

```
Paragraph para =
doc.InsertParagraphAfter(doc.Styles[NamedStyle.BuiltIn.BodyText],
    bodyTextFormatting);
```

Table Formatting

To apply a font to text in a table cell, create a Font object and pass it to TableCell.InsertTextAfter or TableCell.InsertTextBefore, as demonstrated above.

Other paragraph formatting can be applied through the Table object or the TableFormatting object. To create a TableFormatting object, call Document.CreateTableFormatting:

```
WordApplication wwapp = new WordApplication();
Document doc = wwapp.Create();
TableFormatting tableFormat = doc.CreateTableFormatting();
```

Next, set TableFormatting properties.

```
tableFormat.DefaultShading.BackgroundColor = Color.Gray;
tableFormat.SetDefaultPadding(TableCell.Location.Top, 720);
```

To assign the TableFormatting object that you created to a table, pass it to Element.InsertTableAfter or Element.InsertTableBefore when you create a new table.

```
Table tbl = doc.InsertTableAfter(3, 5, tableFormat);
```