Word for WPF

2018.04.11 更新

グレープシティ株式会社

Word for WPF

目次

Word for WPF の概要

ComponentOne に導入された **Word (Beta) for WPF** には、高度な機能を備えた Word ドキュメントを作成するために機能豊富な API が提供されています。**Word for WPF** は、Microsoft Word Open XML 形式の拡張子（*.docx）を持つ Word ドキュメントおよびリッチテキスト形式の拡張子（*.rtf）を持つ RTF ドキュメントの作成、読み取り、書き込みを行うことができます。

Word for WPF は **C1.WPF.Word.C1WordDocument** を使用します。このクラスは、Microsoft Word ドキュメントと RTF ドキュメントの生成に必要な高度なプロパティおよびメソッドをすべて提供します。**Word for WPF** を使用して生成されたドキュメントは、ファイルシステムに保存することも、標準の Microsoft Word ドキュメント形式にエクスポートすることも簡単です。

主要な機能

Word for WPF の主要な機能は次のとおりです。

- **充実したオブジェクトモデル**
  Word for WPF は、機能豊富で強力だが簡単にプログラム可能なオブジェクトモデルを提供します。高度なプロパティとメソッドをすべて提供する **C1WordDocument** クラスだけを使用して、Microsoft Word および RTF ドキュメントの両方を作成できます。

- **高度なライブラリ**
  Word for WPF は、高度なメソッドとして、Word ドキュメント内に画像、テキスト、グラフィック、引用文、および表を追加します。

- **さまざまな用紙サイズ**
  Word for WPF を使用して、さまざまな用紙サイズと向きでドキュメントを作成します。

- **表の追加**
  Word for WPF の RTFTable オブジェクトを使用して、表セルにデータを追加できます。

- **目次の追加**
  Word for WPF では、テキストと見出しを含むドキュメントに目次 (TOC) を追加できます。目次を使用して、ドキュメント内の別のページに移動することもできます。

- **ハイパーリンクとブックマーク**
  Word for WPF は、ドキュメント内を移動するためのブックマークと、別の URL に移動するためのハイパーリンクを提供します。

- **テキストの描画**
  Word for WPF を使用すると、Word ドキュメント内でさまざまなフォントでテキストを描画したり、フォントプロパティを使用することができます。

- **テキストフロー**
  Word for WPF を使用すると、Word ドキュメント内の段やページにテキストをフロー挿入できます。

オブジェクトモデルの概要

Word for WPF は、機能豊富で強力だが簡単にプログラム可能なオブジェクトモデルを提供します。**C1.WPF.Word.C1WordDocument** クラスは、Microsoft Word および RTF ドキュメントを作成するための高度なプロパティとメソッドをすべて提供します。

C1WordDocument オブジェクト
Add、AddBookmark、AddBookmarkStart、AddBookmarkEnd、AddLink、AddParagraph、AddPicture、ColumnBreak、DrawArc、DrawRectangle、DrawString、FillPie、Load、Count、Current、Hyperlink、ShapesWord2007Compatible
Font
Bold、FontFamily、Italic、Name、Size、Strikeout、Style、Underline
Pen
Color、DashPattern、DashStyle
RTFPageSize オブジェクト
A0、A1、A4、A10、B1、B5、HalfLetter、Legal、Letter
StringFormat
Alignment、Angle、LineAlignment、LineSpacing

クイックスタート

クイックスタートガイドでは、**Word for WPF**について詳しく説明します。このセクションでは、Visual Studio で新しい WPF プロジェクトを作成する方法を学びます。ドキュメントを作成して保存するには、C1Word の参照 (dll) とボタンをアプリケーションに追加する必要があります。

手順 1：アプリケーションの設定

最初に Visual Studio で WPF アプリケーションを作成し、次にそのアプリケーションに次の手順で **Word** 参照とボタンコントロールを追加します。

1. Visual Studio で新しい WPF プロジェクトを作成します。
2. アプリケーションに **C1Word** 参照（dll）を追加します。
3. コードビューで、次の名前空間を追加します。

Visual Basic

```
using C1.WPF.Word;
using Microsoft.Win32;
```

C#

```
using C1.WPF.Word;
using Microsoft.Win32;
```

4. デザインビューに切り替えて、アプリケーション内のフォームに Button コントロールを追加し、**C1Word** の使用を開始します。**Content** プロパティに **Text** などの適切なテキストを設定し、**Name** プロパティを **btnText** に設定し、**Click** イベントを **btnText_Click** に設定します。
5. プロパティウィンドウで **btnText_Click** イベントをダブルクリックします。
   コードビューに **btnText_Click** イベントが作成されます。

手順 2：単純なテキストの追加

Visual Studio プロジェクトでコードビューを表示したまま、**btnText_Click** イベントに次のコードを追加します。

Visual Basic

```
' 保存先のストリームを取得します
Dim dlg = New SaveFileDialog()
dlg.FileName = "document"
dlg.DefaultExt = ".docx"
dlg.Filter = "RTF files (*.rtf)|*.rtf|MS Word (Open XML) files
(*.docx)|*.docx"
Dim dr = dlg.ShowDialog()
If Not dr.HasValue OrElse Not dr.Value Then
        Return
End If

' sender ボタンを取得します
Dim btn = TryCast(sender, Button)
```

```
' ドキュメントを作成します
Dim word = New C1WordDocument()
word.Clear()

' ドキュメント情報を設定します
Dim di = word.Info
di.Author = "ComponentOne"
di.Subject = "C1.WPF.Word sample."
di.Title = DirectCast(btn.Content, String)

' テキストを測定および表示します
Dim text = "こんにちは!! これはサンプルテキストです。"
Dim font = New Font("Segoe UI Light", 20, RtfFontStyle.Italic)

' 段落を追加します
word.AddParagraph(text, font, Colors.BlueViolet,
RtfHorizontalAlignment.Justify)

Using stream = dlg.OpenFile()
        word.Save(stream, If(dlg.FileName.ToLower().EndsWith("docx"),
FileFormat.OpenXml, FileFormat.Rtf))
        MessageBox.Show("Word Document saved to " + dlg.SafeFileName)
End Using
```

C#

```
// 保存先のストリームを取得します
var dlg = new SaveFileDialog();
dlg.FileName = "document";
dlg.DefaultExt = ".docx";
dlg.Filter= "RTF files (*.rtf)|*.rtf|MS Word (Open XML) files
(*.docx)|*.docx";
var dr = dlg.ShowDialog();
if (!dr.HasValue || !dr.Value)
{
    return;
}

// sender ボタンを取得します
var btn = sender as Button;

// ドキュメントを作成します
var word = new C1WordDocument();
word.Clear();

// ドキュメント情報を設定します
var di = word.Info;
di.Author = "ComponentOne";
di.Subject = "C1.WPF.Word sample.";
di.Title = (string)btn.Content;

// テキストを測定および表示します
```

```
var text = "こんにちは!! これはサンプルテキストです。";
var font = new Font("Segoe UI Light", 20, RtfFontStyle.Italic);

// 段落を追加します
word.AddParagraph(text, font, Colors.BlueViolet,
RtfHorizontalAlignment.Justify);

using (var stream = dlg.OpenFile())
{
     word.Save(stream, dlg.FileName.ToLower().EndsWith("docx") ?
FileFormat.OpenXml : FileFormat.Rtf);
     MessageBox.Show("Word Document saved to " + dlg.SafeFileName);
}
```

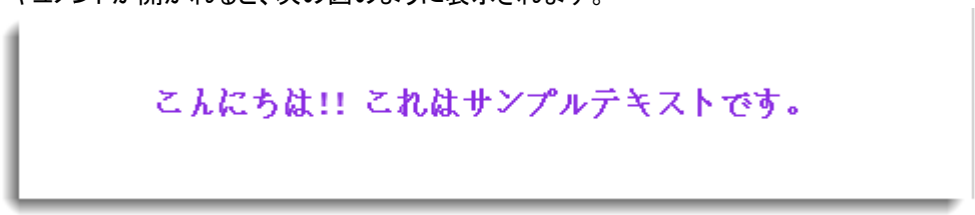
上記のコードは、Word ドキュメントを作成し、**AddParagraph** メソッドを使用してテキストを追加します。作成したドキュメントは、選択した場所に保存できます。目的の場所を選択し、必要に応じてドキュメントの名前も変更できます。ドキュメントは、**document.rtf** という名前または指定した名前で保存されます。

手順 3：アプリケーションの実行

前の手順では、テキストを作成して追加し、Word ドキュメントを保存するコードを **btnText_Click** イベントに追加しました。この手順では、アプリケーションを実行して、作成したドキュメントを表示します。次の手順に従って、アプリケーションを実行します。

1. **[F5]**キーを押してアプリケーションを実行します。
2. ボタンをクリックして、**C1Word** を使用して作成および保存したドキュメントを表示します。

ドキュメントが開かれると、次の図のように表示されます。



Word for WPF の操作

Word for WPF は機能豊富な API とオブジェクトモデルを備えており、Word ドキュメントのほかに、Microsoft Word や他のエディタでサポートされている RTF ドキュメントも作成できます。**Word for WPF** の詳細な仕組みを次のトピックに示します。

基礎レベルの操作

Word for WPF を使用して、画像、グラフィック、引用文などの単純なイラストを Word ドキュメントに追加できます。**Word for WPF** を使用すると、このようなイラストを数行のコードで追加できます。次のトピックでは、単純なテキストといくつかの基本的なイラストを追加する方法について説明します。

テキストの追加

C1Word を使用して、Word ドキュメントにテキストを追加できます。目的のテキストを記述し、**AddParagraph** メソッドを使用してそのテキストを追加する必要があります。Word ドキュメントに表示するテキストに対して、**Font** クラスとそのプロパティを使用して、フォントのスタイル、ファミリ、色などのプロパティを設定することもできます。Word ドキュメントへのテキストの追加は、次のコードで実装されます。

1. 次のコードで、**C1WordDocument** クラスのオブジェクトを作成します。

Visual Basic

```
Dim word = New C1WordDocument()
```

C#

```
var word = new C1WordDocument();
```

2. ドキュメントにテキストを追加するには、次のコードを記述します。

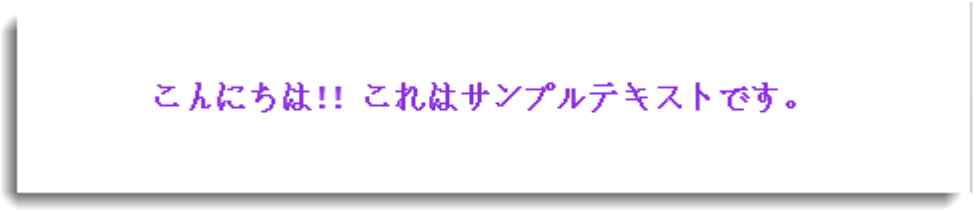
Visual Basic

```
Dim text = "こんにちは!! これはサンプルテキストです。"
Dim font = New Font("Segoe UI Light", 20, RtfFontStyle.Italic)
word.AddParagraph(text, font, Colors.BlueViolet,
RtfHorizontalAlignment.Justify)
```

C#

```
var text = "こんにちは!! これはサンプルテキストです。";
var font = new Font("Segoe UI Light", 20, RtfFontStyle.Italic);
word.AddParagraph(text, font, Colors.BlueViolet,
RtfHorizontalAlignment.Justify);
```

ドキュメントは、次の図のように表示されます。



画像の追加

Word ドキュメントにテキストに加えて画像を挿入して、全体的に見栄えをよくしたい場合があります。ドキュメントに画像を追加するには、次のコードを使用します。これは、画像をロードしてドキュメント内にスケッチします。

以下のコードでは、**WordUtils** および **DataAccess** という 2 つのクラスが使用されています。これらのクラスは、システムの次の場所にある製品サンプル内に置かれています。
Documents\ComponentOne Samples\WPF\WordCreator
これらのクラスを上記の場所からアプリケーションで使用できます。

Visual Basic

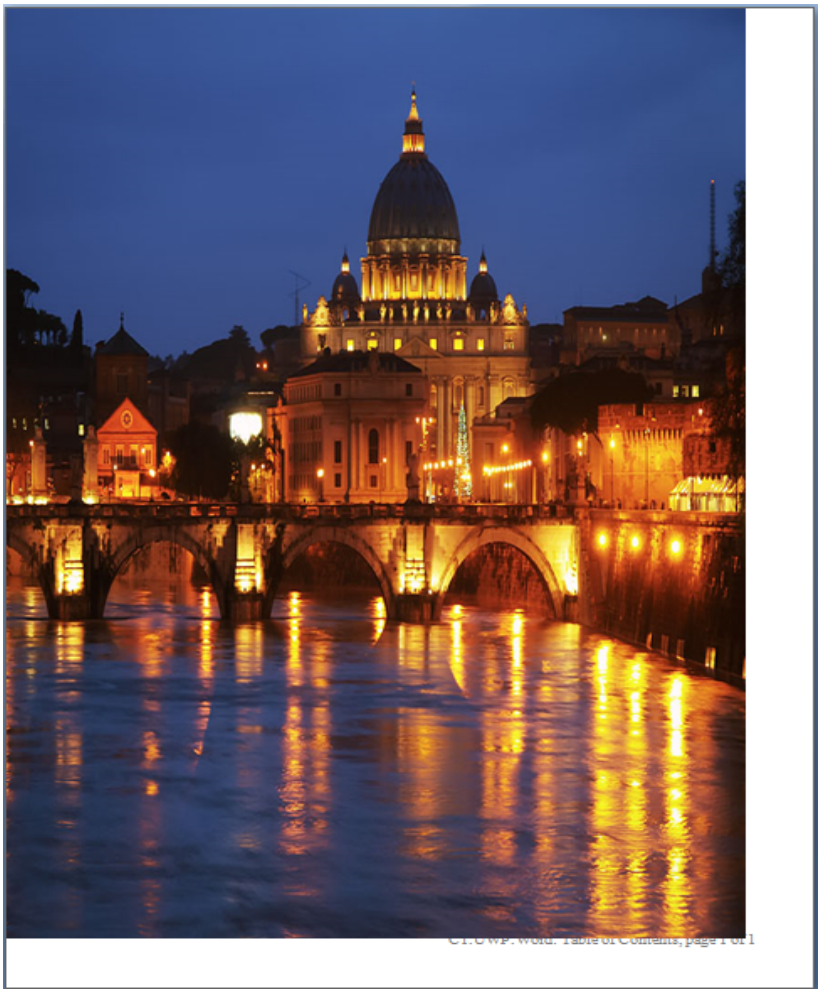
```
' ページ四角形を計算します(マージンを差し引いて)
Dim rcPage As Rect = WordUtils.PageRectangle(word)

' 書き込み可能なビットマップに画像をロードします
Dim bi As New BitmapImage()
bi.BeginInit()
bi.StreamSource = DataAccess.GetStream("borabora.jpg")
bi.EndInit()
Dim wb = New WriteableBitmap(bi)

' アスペクト比を維持して画像をページの中央に配置します
word.DrawImage(wb, rcPage)
```

C#

```
// ページ四角形を計算します(マージンを差し引いて)
Rect rcPage = WordUtils.PageRectangle(word);

// 書き込み可能なビットマップに画像をロードします
BitmapImage bi = new BitmapImage();
bi.BeginInit();
bi.StreamSource = DataAccess.GetStream("borabora.jpg");
bi.EndInit();
var wb = new WriteableBitmap(bi);

// アスペクト比を維持して画像をページの中央に配置します
word.DrawImage(wb, rcPage);
```

上記のコードで、画像は書き込み可能なビットマップにロードされ、**DrawImage** メソッドを使用して描画されます。

上記のコードの出力は、次の図のようになります。



グラフィックの描画

グラフィックを追加することで、ドキュメントの見栄えがよくなり、視覚に訴えることができます。ドキュメントに、円弧、ベジェ、楕円、直線、円、多角形、折れ線、四角形などのさまざまなタイプの図形を追加できます。テキストに加えて円、四角形、ポリライン、ベジェなどのグラフィックを追加するには、次のコードを使用します。

Visual Basic

```
Dim rtf = New C1WordDocument()
' 描画の設定を行います
Dim rc As New Rect(100, 100, 300, 200)
Dim text As String = "Hello world of .NET Graphics and Word/RTF." & vbCr
& vbLf & "よろしくお願いします。"
Dim font As New Font("Times New Roman", 12, RtfFontStyle.Italic Or
RtfFontStyle.Underline)

' PDF ドキュメントに描画します
Dim penWidth As Integer = 0
Dim penRGB As Byte = 0
rtf.FillPie(Colors.Red, rc, 0, 20F)
rtf.FillPie(Colors.Green, rc, 20F, 30F)
rtf.FillPie(Colors.Blue, rc, 60F, 12F)
rtf.FillPie(Colors.Orange, rc, -80F, -20F)
For startAngle As Single = 0 To 359 Step 40
```

```
        Dim penColor As Color = Color.FromArgb(&Hff, penRGB, penRGB,
penRGB)
        Dim pen As New Pen(penColor,
System.Math.Max(System.Threading.Interlocked.Increment(penWidth),penWidth
- 1))
        penRGB = CByte(penRGB + 20)
        rtf.DrawArc(pen, rc, startAngle, 40F)
Next
rtf.DrawRectangle(Colors.Red, rc)
rtf.DrawString(text, font, Colors.Black, rc)

' ベジェ曲線を表示します
Dim pts = New Point() {New Point(400, 200), New Point(420, 130), New
Point(500, 240), New Point(530, 120)}

' ベジェを描画します
rtf.DrawBeziers(New Pen(Colors.Blue, 4), pts)

' ベジェ制御点を表示します
rtf.DrawPolyline(Colors.Gray, pts)
For Each pt As Point In pts
        rtf.FillRectangle(Colors.Red, pt.X - 2, pt.Y - 2, 4, 4)
Next

' タイトル
rtf.DrawString("単純なベジェ", font, Colors.Black, New Rect(500, 150, 100,
100))
```

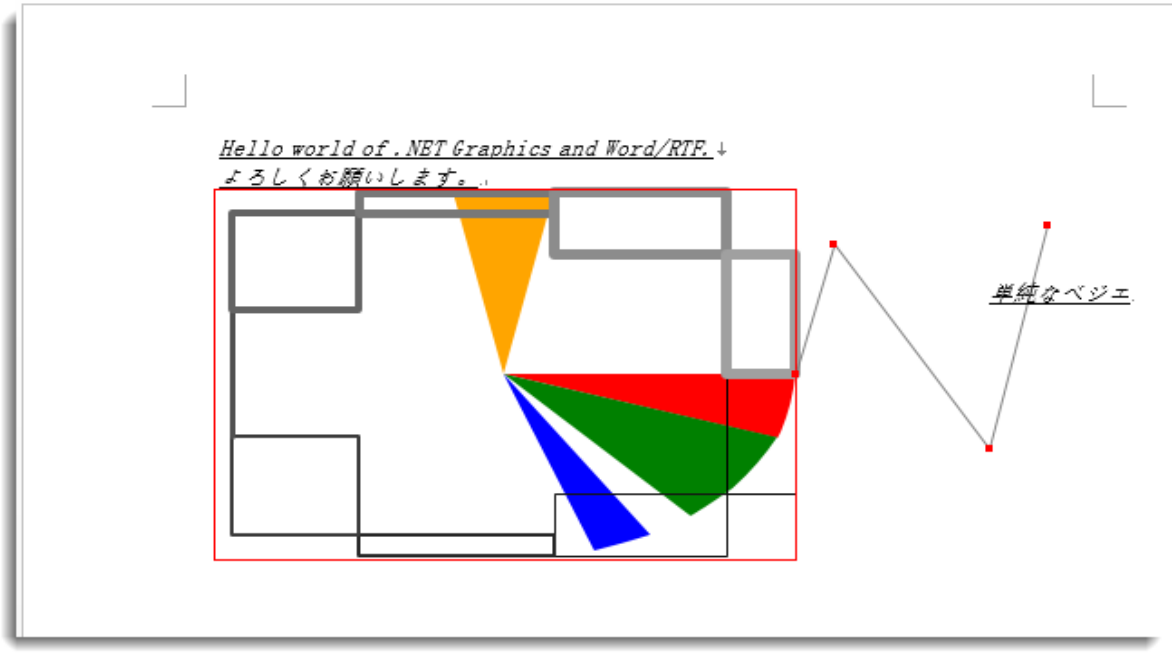
C#

```
var rtf = new C1WordDocument();
// 描画の設定を行います
Rect rc = new Rect(100, 100, 300, 200);
string text = "Hello world of .NET Graphics and Word/RTF.\r\nよろしくお願いし
ます。";
Font font = new Font("Times New Roman", 12, RtfFontStyle.Italic |
RtfFontStyle.Underline);

// PDF ドキュメントに描画します
int penWidth = 0;
byte penRGB = 0;
rtf.FillPie(Colors.Red, rc, 0, 20f);
rtf.FillPie(Colors.Green, rc, 20f, 30f);
rtf.FillPie(Colors.Blue, rc, 60f, 12f);
rtf.FillPie(Colors.Orange, rc, -80f, -20f);
for (float startAngle = 0; startAngle < 360; startAngle += 40)
{
     Color penColor = Color.FromArgb(0xff, penRGB, penRGB, penRGB);
     Pen pen = new Pen(penColor, penWidth++);
     penRGB = (byte)(penRGB + 20);
     rtf.DrawArc(pen, rc, startAngle, 40f);
}
```

```
rtf.DrawRectangle(Colors.Red, rc);
rtf.DrawString(text, font, Colors.Black, rc);

// ベジェ曲線を表示します
var pts = new Point[]
{
      new Point(400, 200), new Point(420, 130),
      new Point(500, 240), new Point(530, 120),
};

// ベジェを描画します
rtf.DrawBeziers(new Pen(Colors.Blue, 4), pts);

// ベジェ制御点を表示します
rtf.DrawPolyline(Colors.Gray, pts);
foreach (Point pt in pts)
{
    rtf.FillRectangle(Colors.Red, pt.X - 2, pt.Y - 2, 4, 4);
}

// タイトル
rtf.DrawString("単純なベジェ", font, Colors.Black, new Rect(500, 150, 100,
100));
```

上記のコードでは、**DrawRectangle**、**DrawArc**、**DrawPolyline**、および **DrawBeziers** メソッドを使用してさまざまなタイプのグラフィック（直線、四角形、円、ベジェなど）を描画しています。また、**DrawString** メソッドを使用してテキストを描画してドキュメントに表示しています。

上記のコードの出力は、次の図のようになります。



引用文の追加

C1Word を使用して、別のファイルからの引用文をドキュメントに追加できます。次のコードを使用して、テキストファイルからの引用文をドキュメントに追加します。

以下のコードでは、**WordUtils** および **DataAccess** という 2 つのクラスが使用されています。これらのクラスは、システムの次の場所にある製品サンプル内に置かれています。
Documents\ComponentOne Samples\WPF\WordCreator
これらのクラスを上記の場所からアプリケーションで使用できます。

Visual Basic

```
Dim titleFont As New Font("Arial", 24, RtfFontStyle.Bold)
Dim txtFont As New Font("Times New Roman", 10, RtfFontStyle.Italic)

' タイトルを追加します
Dim rcTop = WordUtils.RenderParagraph(word, word.Info.Title, titleFont,
rcPage, rc)
rc = rcTop
' ドキュメントを構築します
For Each s As String In GetQuotes()
        Dim authorQuote As String() = s.Split(ControlChars.Tab)

        ' ヘッダーをレンダリングします（作成者）
        Dim author = authorQuote(0)
        rc.Y += 25
        rc = WordUtils.RenderParagraph(word, author, hdrFont, rcPage,
rc, True)

        ' 本文をレンダリングします（引用文）
        Dim text As String = authorQuote(1)
        rc.X = rcPage.X + 36
        ' << 本文を 1/2 インチインデントします
        rc.Width = rcPage.Width - 40
        rc = WordUtils.RenderParagraph(word, text, txtFont, rcPage, rc)
        rc.X = rcPage.X
        ' << インデントを元に戻します
        rc.Width = rcPage.Width
        rc.Y += 12
        ' << 各引用文の後に 12pt のスペースを追加します
        If rc.Y > rcPage.Height Then
                word.PageBreak()
                rc = rcTop
        End If
Next

Private Shared Function GetQuotes() As List(Of String)
        Dim list = New List(Of String)()

        Using sr = New StreamReader(DataAccess.GetStream("quotes.txt"))
                Dim quotes = sr.ReadToEnd()
                For Each quote As String In quotes.Split("*"C)
                        Dim pos As Integer = quote.IndexOf(vbCr & vbLf)
                        If pos > -1 Then
                                Dim q = String.Format("{0}" & vbTab & "
{1}", quote.Substring(0, pos), quote.Substring(pos + 2).Trim())
                                list.Add(q)
                        End If
```

```
                Next
        End Using

        Return list
End Function
```

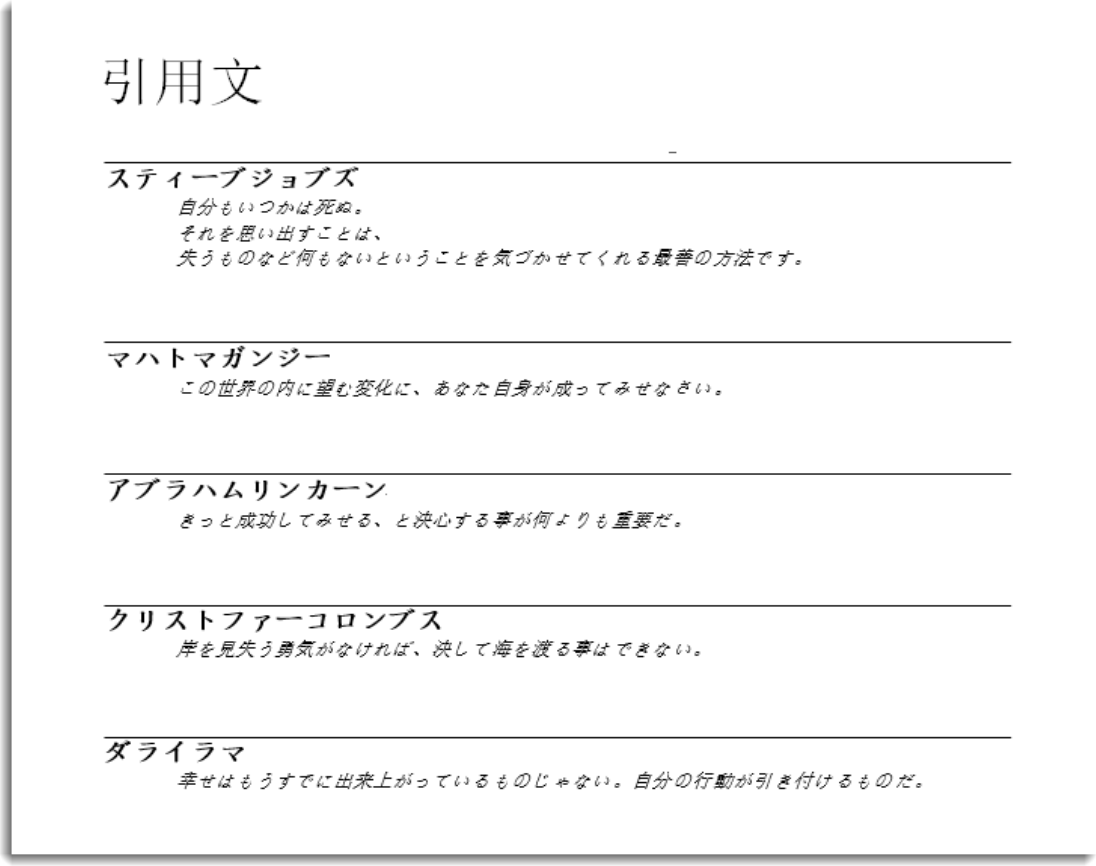
C#

```
// ページ四角形を計算します(マージンを差し引いて)
Rect rcPage = WordUtils.PageRectangle(word);
Rect rc = rcPage;

// 出力パラメータを初期化します
Font hdrFont = new Font("Arial", 14, RtfFontStyle.Bold);
Font titleFont = new Font("Arial", 24, RtfFontStyle.Bold);
Font txtFont = new Font("Times New Roman", 10, RtfFontStyle.Italic);

// タイトルを追加します
var rcTop = WordUtils.RenderParagraph(word, word.Info.Title, titleFont,
rcPage, rc);
rc = rcTop;

// ドキュメントを構築します
foreach (string s in GetQuotes())
{
    string[] authorQuote = s.Split('\t');

    // ヘッダーをレンダリングします(作成者)
    var author = authorQuote[0];
    rc.Y += 25;
    rc = WordUtils.RenderParagraph(word, author, hdrFont, rcPage, rc,
true);

    // 本文をレンダリングします(引用文)
    string text = authorQuote[1];
    rc.X = rcPage.X + 36; // << 本文を 1/2 インチインデントします
    rc.Width = rcPage.Width - 40;
    rc = WordUtils.RenderParagraph(word, text, txtFont, rcPage, rc);
    rc.X = rcPage.X; // << インデントを元に戻します
    rc.Width = rcPage.Width;
    rc.Y += 12; // << 各引用文の後に 12pt のスペースを追加します
    if (rc.Y > rcPage.Height)
    {
        word.PageBreak();
        rc = rcTop;
     }
}

static List <string> GetQuotes()
{
    var list = new List <string>();

    using (var sr = new
```

```
StreamReader(DataAccess.GetStream("quotes.txt")))
    {
        var quotes = sr.ReadToEnd();
        foreach (string quote in quotes.Split('*'))
        {
            int pos = quote.IndexOf("\r\n");
            if (pos > -1)
            {
                var q = string.Format("{0}\t{1}", quote.Substring(0,
pos), quote.Substring(pos + 2).Trim());
                list.Add(q);
            }
        }
}

return list;
}
```

上記のコードは、テキストファイルから引用文を読み込んでドキュメントに書き出します。まずタイトルをドキュメントに追加し、次にヘッダーと本文をレンダリングしてから、ドキュメントにテキストを書き込みます。

上記のコードの出力は、次の図のようになります。



上級レベルの操作

通常、Word ドキュメントにはテキストが入りますが、これに画像、イラスト、表、メタファイルなどを追加することで、見栄えをよくし、かつわかりやすくすることができます。次に示すトピックでは、**Word for WPF** を使用して、これらの高度な機能を Word ドキュメントに追加する方法を説明します。

表の挿入

表は、Word ドキュメントでデータをいくつかの行と列に整えて表示するために使用されます。

以下のコードでは、**WordUtils** および **DataAccess** という 2 つのクラスが使用されています。これらのクラスは、システムの次の場所にある製品サンプル内に置かれています。
Documents\ComponentOne Samples\WPF\WordCreator
これらのクラスを上記の場所からアプリケーションで使用できます。

Word for WPF を使用すると、次のコードで構造を Word ドキュメントに追加できます。

Visual Basic

```
' データを取得します
Dim ds = DataAccess.GetDataSet()

' ページ四角形を計算します(マージンを差し引いて)
Dim rcPage As Rect = WordUtils.PageRectangle(word)
Dim rc As Rect = rcPage

' タイトルを追加します
Dim titleFont As New Font("Tahoma", 24, RtfFontStyle.Bold)
rc = WordUtils.RenderParagraph(word, word.Info.Title, titleFont, rcPage,
rc, False)

' いくつかのテーブルをレンダリングします
RenderTable(word, rc, rcPage, ds.Tables("Customers"), New String()
{"CompanyName", "ContactName", "Country", "Address", "Phone"})
```

C#

```
// データを取得します
var ds = DataAccess.GetDataSet();

// ページ四角形を計算します(マージンを差し引いて)
Rect rcPage = WordUtils.PageRectangle(word);
Rect rc = rcPage;

// タイトルを追加します
Font titleFont = new Font("Tahoma", 24, RtfFontStyle.Bold);
rc = WordUtils.RenderParagraph(word, word.Info.Title, titleFont, rcPage,
rc, false);

// いくつかのテーブルをレンダリングします
RenderTable(word, rc, rcPage, ds.Tables["Customers"], new string[]
{ "CompanyName", "ContactName", "Country", "Address", "Phone" });
```

テーブルのコンテンツのフォントを選択し、テーブルを構築するには、次のコードを使用できます。

Visual Basic

```
' フォントを選択します
Dim hdrFont As New Font("Tahoma", 10, RtfFontStyle.Bold)
Dim txtFont As New Font("Tahoma", 8)
```

```
' テーブルを構築します
'word.AddBookmark(table.TableName, 0, rc.Y);
rc = WordUtils.RenderParagraph(word, "NorthWind " + table.TableName,
hdrFont, rcPage, rc, False)

' テーブルを構築します
rc = RenderTableHeader(word, hdrFont, rc, fields)
For Each dr As DataRow In table.Rows
        rc = RenderTableRow(word, txtFont, hdrFont, rcPage, rc, fields,
_
                dr)
Next

' 終了
Return rc
```

C#

```
// フォントを選択します
Font hdrFont = new Font("Tahoma", 10, RtfFontStyle.Bold);
Font txtFont = new Font("Tahoma", 8);

// テーブルを構築します
//word.AddBookmark(table.TableName, 0, rc.Y);
rc = WordUtils.RenderParagraph(word, "NorthWind " + table.TableName,
hdrFont, rcPage, rc, false);

// テーブルを構築します
rc = RenderTableHeader(word, hdrFont, rc, fields);
foreach (DataRow dr in table.Rows)
{
     rc = RenderTableRow(word, txtFont, hdrFont, rcPage, rc, fields,
dr);
}

// 終了
return rc;
```

テーブルを構築したら、次のコードを使用して、テーブルヘッダーと行のセルの高さと幅を計算し、それらをレンダリングする必要があります。

Visual Basic

```
Private Shared Function RenderTableHeader(word As C1WordDocument, font
As Font, rc As Rect, fields As String()) As Rect
        ' セルの幅を計算します(すべての列で同じ)
        Dim rcCell As Rect = rc
        rcCell.Width = rc.Width / fields.Length
        rcCell.Height = 0

        ' セルの高さを計算します(すべての列の最大値)
        For Each field As String In fields
```

```
                Dim height = word.MeasureString(field, font,
rcCell.Width).Height
                rcCell.Height = Math.Max(rcCell.Height, height)
        Next
        rcCell.Height += 6
        ' 6 ポイントのマージンを追加します
        ' ヘッダーセルをレンダリングします
        Dim fmt = New StringFormat()
        fmt.LineAlignment = VerticalAlignment.Center
        For Each field As String In fields
                word.FillRectangle(Colors.Black, rcCell)
                word.DrawString(field, font, Colors.White, rcCell, fmt)
                rcCell = WordUtils.Offset(rcCell, rcCell.Width, 0)
        Next

        ' 四角形を更新して返します
        Return WordUtils.Offset(rc, 0, rcCell.Height)
End Function

Private Shared Function RenderTableRow(word As C1WordDocument, font As
Font, hdrFont As Font, rcPage As Rect, rc As Rect, fields As String(), _
        dr As DataRow) As Rect
        ' セルの幅を計算します(すべての列で同じ)
        Dim rcCell As Rect = rc
        rcCell.Width = rc.Width / fields.Length
        rcCell.Height = 0

        ' セルの高さを計算します(すべての列の最大値)
        rcCell = WordUtils.Inflate(rcCell, -4, 0)
        For Each field As String In fields
                Dim text As String = dr(field).ToString()
                Dim height = word.MeasureString(text, font,
rcCell.Width).Height
                rcCell.Height = Math.Max(rcCell.Height, height)
        Next
        rcCell = WordUtils.Inflate(rcCell, 4, 0)
        ' 4 ポイントのマージンを追加します
        rcCell.Height += 2

        ' 必要なら、改行します
        If rcCell.Bottom > rcPage.Bottom Then
                word.PageBreak()
                rc = RenderTableHeader(word, hdrFont, rcPage, fields)
                rcCell.Y = rc.Y
        End If

        ' 垂直方向に中央揃えにします
        Dim fmt As New StringFormat()
        fmt.LineAlignment = VerticalAlignment.Center

        ' データセルをレンダリングします
        For Each field As String In fields
```

```
                ' コンテンツを取得します
                Dim text As String = dr(field).ToString()

                ' 水平方向の配置を設定します
                Dim d As Double
                fmt.Alignment = If((Double.TryParse(text,
NumberStyles.Any, CultureInfo.CurrentCulture, d)),
HorizontalAlignment.Right, HorizontalAlignment.Left)

                ' セルをレンダリングします
                word.DrawRectangle(Colors.LightGray, rcCell)
                rcCell = WordUtils.Inflate(rcCell, -4, 0)
                word.DrawString(text, font, Colors.Black, rcCell, fmt)
                rcCell = WordUtils.Inflate(rcCell, 4, 0)
                rcCell = WordUtils.Offset(rcCell, rcCell.Width, 0)
        Next

        ' 四角形を更新して返します
        Return WordUtils.Offset(rc, 0, rcCell.Height)
End Function
```

C#

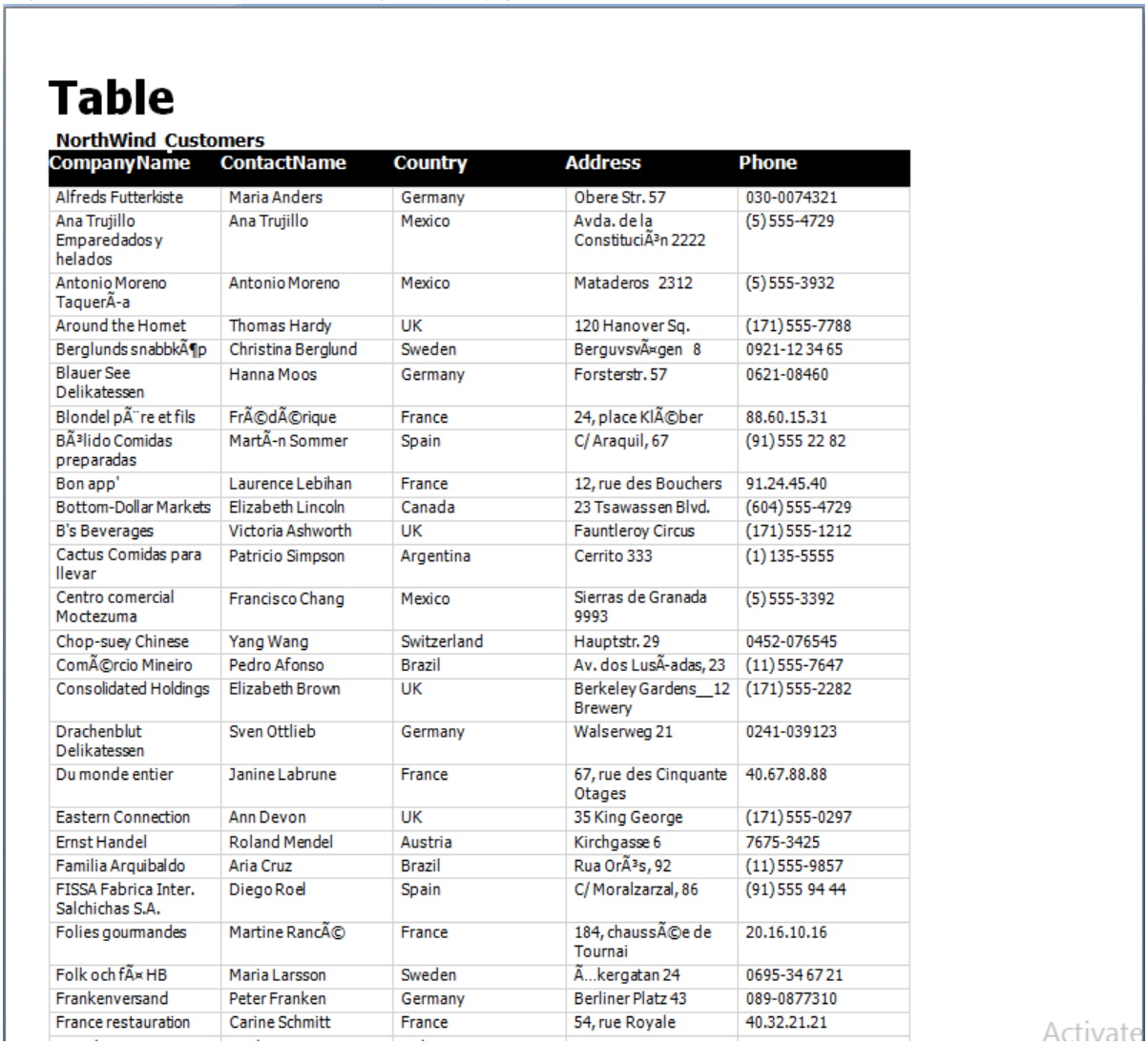
```
static Rect RenderTableHeader(C1WordDocument word, Font font, Rect rc,
string[] fields)
{
// セルの幅を計算します(すべての列で同じ)
Rect rcCell = rc;
rcCell.Width = rc.Width / fields.Length;
rcCell.Height = 0;

// セルの高さを計算します(すべての列の最大値)
foreach (string field in fields)
{
     var height = word.MeasureString(field, font, rcCell.Width).Height;
     rcCell.Height = Math.Max(rcCell.Height, height);
}
rcCell.Height += 6; // 6 ポイントのマージンを追加します

// ヘッダーセルをレンダリングします
var fmt = new StringFormat();
fmt.LineAlignment = VerticalAlignment.Center;
foreach (string field in fields)
{
     word.FillRectangle(Colors.Black, rcCell);
     word.DrawString(field, font, Colors.White, rcCell, fmt);
     rcCell = WordUtils.Offset(rcCell, rcCell.Width, 0);
}

// 四角形を更新して返します
      return WordUtils.Offset(rc, 0, rcCell.Height);
}
```

```
static Rect RenderTableRow(C1WordDocument word, Font font, Font hdrFont,
Rect rcPage, Rect rc, string[] fields, DataRow dr)
{
// セルの幅を計算します（すべての列で同じ）
Rect rcCell = rc;
rcCell.Width = rc.Width / fields.Length;
rcCell.Height = 0;

// セルの高さを計算します（すべての列の最大値）
rcCell = WordUtils.Inflate(rcCell, -4, 0);
foreach (string field in fields)
{
     string text = dr[field].ToString();
     var height = word.MeasureString(text, font, rcCell.Width).Height;
     rcCell.Height = Math.Max(rcCell.Height, height);
}
rcCell = WordUtils.Inflate(rcCell, 4, 0); // 4 ポイントのマージンを追加します
rcCell.Height += 2;

// 必要なら、改行します
if (rcCell.Bottom > rcPage.Bottom)
{
     word.PageBreak();
     rc = RenderTableHeader(word, hdrFont, rcPage, fields);
     rcCell.Y = rc.Y;
}

// 垂直方向に中央揃えにします
StringFormat fmt = new StringFormat();
fmt.LineAlignment = VerticalAlignment.Center;

// データセルをレンダリングします
foreach (string field in fields)
{

         // コンテンツを取得します
    string text = dr[field].ToString();

    // 水平方向の配置を設定します
    double d;
    fmt.Alignment = (double.TryParse(text, NumberStyles.Any,
CultureInfo.CurrentCulture, out d))
    ? HorizontalAlignment.Right
    : HorizontalAlignment.Left;

// セルをレンダリングします
word.DrawRectangle(Colors.LightGray, rcCell);
rcCell = WordUtils.Inflate(rcCell, -4, 0);
word.DrawString(text, font, Colors.Black, rcCell, fmt);
rcCell = WordUtils.Inflate(rcCell, 4, 0);
rcCell = WordUtils.Offset(rcCell, rcCell.Width, 0);
```

```
}

// 四角形を更新して返します
return WordUtils.Offset(rc, 0, rcCell.Height);
}
```

上記のコードは Nwind データベースから取得した情報に基づき、適切なインデントと配置を使用して Word ドキュメントにテーブルを作成します。

上記のコードの出力は、次の図のようになります。



Table

NorthWind_Customers

CompanyName	ContactName	Country	Address	Phone
Alfreds Futterkiste	Maria Anders	Germany	Obere Str. 57	030-0074321
Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico	Avda. de la ConstituciÃ³n 2222	(5) 555-4729
Antonio Moreno TaquerÃ-a	Antonio Moreno	Mexico	Mataderos 2312	(5) 555-3932
Around the Horn	Thomas Hardy	UK	120 Hanover Sq.	(171) 555-7788
Berglunds snabbkÃ¶p	Christina Berglund	Sweden	BerguvsvÃ¤gen 8	0921-12 34 65
Blauer See Delikatessen	Hanna Moos	Germany	Forsterstr. 57	0621-08460
Blondel pÃ¨re et fils	FrÃ©dÃ©rique	France	24, place KlÃ©ber	88.60.15.31
BÃ³lido Comidas preparadas	MartÃ-n Sommer	Spain	C/ Araquil, 67	(91) 555 22 82
Bon app'	Laurence Lebihan	France	12, rue des Bouchers	91.24.45.40
Bottom-Dollar Markets	Elizabeth Lincoln	Canada	23 Tsawassen Blvd.	(604) 555-4729
B's Beverages	Victoria Ashworth	UK	Fauntleroy Circus	(171) 555-1212
Cactus Comidas para llevar	Patricio Simpson	Argentina	Cerrito 333	(1) 135-5555
Centro comercial Moctezuma	Francisco Chang	Mexico	Sierras de Granada 9993	(5) 555-3392
Chop-suey Chinese	Yang Wang	Switzerland	Hauptstr. 29	0452-076545
ComÃ©rcio Mineiro	Pedro Afonso	Brazil	Av. dos LusÃ-adas, 23	(11) 555-7647
Consolidated Holdings	Elizabeth Brown	UK	Berkeley Gardens 12 Brewery	(171) 555-2282
Drachenblut Delikatessen	Sven Ottlieb	Germany	Walserweg 21	0241-039123
Du monde entier	Janine Labrune	France	67, rue des Cinquante Otages	40.67.88.88
Eastern Connection	Ann Devon	UK	35 King George	(171) 555-0297
Ernst Handel	Roland Mendel	Austria	Kirchgasse 6	7675-3425
Familia Arquibaldo	Aria Cruz	Brazil	Rua OrÃ³s, 92	(11) 555-9857
FISSA Fabrica Inter. Salchichas S.A.	Diego Roel	Spain	C/ Moralzarzal, 86	(91) 555 94 44
Folies gourmandes	Martine RancÃ©	France	184, chaussÃ©e de Tournai	20.16.10.16
Folk och fÃ¤ HB	Maria Larsson	Sweden	Ã…kergatan 24	0695-34 67 21
Frankenversand	Peter Franken	Germany	Berliner Platz 43	089-0877310
France restauration	Carine Schmitt	France	54, rue Royale	40.32.21.21

TOC の追加

Word for WPF では、テキストと見出しを含む Word ドキュメントに目次(TOC)を追加できます。これらの見出しを使用して TOC を作成します。次のコードは、テキストと見出しを含む Word ドキュメントの TOC を作成します。

次のコードでは、**WordUtils** というクラスを使用しています。このクラスは、システムの次の場所にある製品サンプル内に置かれています。
Documents\ComponentOne Samples\WPF\WordCreator
これらのクラスを上記の場所からアプリケーションで使用できます。

派生しているWindowクラス内に次のコード行を追加して、乱数を生成し、文字列参照を使用します。

Visual Basic

```
Shared rnd As New Random()
Shared _extension As String = ".rtf"
```

C#

```
static Random rnd = new Random();
static string _extension = ".rtf";
```

次のコードは、テキストとヘッダーを含むWord文書のコンテンツとTOCを作成します。

Visual Basic

```
' ドキュメントを作成します
Dim word = New C1WordDocument()
word.Clear()
word.Info.Title = "Document with Table of Contents"

' タイトルを追加します
Dim titleFont As New Font("Tahoma", 24, RtfFontStyle.Bold)
Dim rcPage As Rect = WordUtils.PageRectangle(word)
Dim rc As Rect = WordUtils.RenderParagraph(word, word.Info.Title,
             _titleFont, rcPage, rcPage, False)
rc.Y += 12

' ナンセンス文書を作成します
Dim bkmk = New List()
Dim headerFont As New Font("Arial", 14, RtfFontStyle.Bold)
Dim bodyFont As New Font("Times New Roman", 11)
For i As Integer = 0 To 29
        ' i番目のヘッダを作成します(リンク先とアウトラインエントリとして)
        Dim header As String = String.Format("{0}. {1}", i + 1, BuildRandomTitle())
        rc = WordUtils.RenderParagraph(word, header, headerFont, rcPage, rc, True, _
                True)

        ' 後で TOC を構築するためにブックマークを保存します
        Dim pageNumber As Integer = 1
        bkmk.Add(New String() {pageNumber.ToString(), header})

        ' テキストを作成します
        rc.X += 36
        rc.Width -= 36

        For j As Integer = 0 To 3 + (rnd.[Next](20) - 1)
                Dim text As String = BuildRandomParagraph()
                rc = WordUtils.RenderParagraph(word, text, bodyFont, rcPage, rc)
                rc.Y += 6
        Next
        rc.X -= 36
        rc.Width += 36
        rc.Y += 20
Next

' TOC を開始します
word.PageBreak()
' 新しいページで TOC を開始します
rc = WordUtils.RenderParagraph(word, "WPFのコントロール", titleFont, rcPage, rcPage, True)
rc.Y += 12
rc.X += 30
rc.Width -= 40

' TOC をレンダリングします
Dim dottedPen As New C1.WPF.Word.Pen(Colors.Gray, 1.5F)
dottedPen.DashStyle = C1.WPF.Word.DashStyle.Dot
Dim sfRight As New StringFormat()
sfRight.Alignment = HorizontalAlignment.Right
rc.Height = bodyFont.Size * 1.2

For Each entry As String() In bkmk
        ' ブックマーク情報を取得します
        Dim page As String = entry(0)
        Dim header As String = entry(1)

        ' 見出し名とページ番号をレンダリングします
        word.DrawString(header, bodyFont, Colors.Black, rc)
        word.DrawString(page, bodyFont, Colors.Black, rc, sfRight)
```

```
        ' 2つをいくつかの点で結びます(点線よりも見やすい)
        Dim dots As String = ". "
        Dim wid = word.MeasureString(dots, bodyFont).Width
        Dim x1 = rc.X + word.MeasureString(header, bodyFont).Width + 8
        Dim x2 = rc.Right - word.MeasureString(page, bodyFont).Width - 8
        Dim x = rc.X
        rc.X = x1
        While rc.X < x2
                word.DrawString(dots, bodyFont, Colors.Gray, rc)
                rc.X += wid
        End While
        rc.X = x

        ' 次のエントリに移動します
        rc = WordUtils.Offset(rc, 0, rc.Height)
        If rc.Bottom > rcPage.Bottom Then
                word.PageBreak()
                rc.Y = rcPage.Y
        End If
Next

' 保存するストリームを取得します
Dim dlg = New SaveFileDialog()
dlg.FileName = "Sample document"
dlg.DefaultExt = ".docx"
dlg.Filter = WordUtils.GetFileFilter(_extension)
Dim dr = dlg.ShowDialog()
If Not dr.HasValue OrElse Not dr.Value Then
        Return
End If

' 文書を保存します
Using stream = dlg.OpenFile()
        word.Save(stream, If(dlg.FileName.ToLower().EndsWith(".docx"),
_FileFormat.OpenXml, FileFormat.Rtf))
End Using
MessageBox.Show("Word Document saved to " + dlg.SafeFileName)
```

C#

```
// ドキュメントを作成しますvar word = new C1WordDocument();
word.Clear();
word.Info.Title = "Document with Table of Contents";

// タイトルを追加しますFont titleFont = new Font("Tahoma", 24, RtfFontStyle.Bold);
Rect rcPage = WordUtils.PageRectangle(word);
Rect rc = WordUtils.RenderParagraph(word, word.Info.Title, titleFont, rcPage, rcPage, false);
rc.Y += 12;

// ナンセンス文書を作成しますvar bkmk = new List();
Font headerFont = new Font("Arial", 14, RtfFontStyle.Bold);
Font bodyFont = new Font("Times New Roman", 11);
for (int i = 0; i < 30; i++)
{
    // i番目のヘッダを作成します(リンク先とアウトラインエントリとして)

    string header = string.Format("{0}. {1}", i + 1, BuildRandomTitle());
    rc = WordUtils.RenderParagraph(word, header, headerFont, rcPage, rc, true, true);

     // 後で TOC を構築するためにブックマークを保存します
     int pageNumber = 1;
     bkmk.Add(new string[] { pageNumber.ToString(), header });

     // テキストを作成します
     rc.X += 36;
     rc.Width -= 36;

     for (int j = 0; j < 3 + rnd.Next(20); j++)
     {
          string text = BuildRandomParagraph();
          rc = WordUtils.RenderParagraph(word, text, bodyFont, rcPage, rc);
          rc.Y += 6;
      }
      rc.X -= 36;
```

```
        rc.Width += 36;
        rc.Y += 20;
}

// TOC を開始します
word.PageBreak();

// 新しいページで TOC を開始します
rc = WordUtils.RenderParagraph(word, "WPFのコントロール", titleFont, rcPage, rcPage, true);
rc.Y += 12;
rc.X += 30;
rc.Width -= 40;

// TOC をレンダリングします
C1.WPF.Word.Pen dottedPen = new C1.WPF.Word.Pen(Colors.Gray, 1.5f);
dottedPen.DashStyle = C1.WPF.Word.DashStyle.Dot;
StringFormat sfRight = new StringFormat();
sfRight.Alignment = HorizontalAlignment.Right;
rc.Height = bodyFont.Size * 1.2;

foreach (string[] entry in bkmk)
{
     // ブックマーク情報を取得します
     string page = entry[0];
     string header = entry[1];

      // 見出し名とページ番号をレンダリングします
      word.DrawString(header, bodyFont, Colors.Black, rc);
      word.DrawString(page, bodyFont, Colors.Black, rc, sfRight);


      // 2つをいくつかの点で結びます（点線よりも見やすい）
      string dots = ". ";
      var wid = word.MeasureString(dots, bodyFont).Width;
      var x1 = rc.X + word.MeasureString(header, bodyFont).Width + 8;
      var x2 = rc.Right - word.MeasureString(page, bodyFont).Width - 8;
      var x = rc.X;
      for (rc.X = x1; rc.X < x2; rc.X += wid)
      {
           word.DrawString(dots, bodyFont, Colors.Gray, rc);
      }
      rc.X = x;

      // 次のエントリに移動します

      rc = WordUtils.Offset(rc, 0, rc.Height);
      if (rc.Bottom > rcPage.Bottom)
      {
           word.PageBreak();
           rc.Y = rcPage.Y;
      }
}

// 保存するストリームを取得します
var dlg = new SaveFileDialog();
dlg.FileName = "Sample document";
dlg.DefaultExt = ".docx";
dlg.Filter = WordUtils.GetFileFilter(_extension);
var dr = dlg.ShowDialog();
if (!dr.HasValue || !dr.Value)
{
     return;
}

// ドキュメントを保存します
using (var stream = dlg.OpenFile())
{
     word.Save(stream, dlg.FileName.ToLower().EndsWith(".docx") ? FileFormat.OpenXml :
FileFormat.Rtf);
}
MessageBox.Show("Word Document saved to " + dlg.SafeFileName);
```

上記のコードは、ドキュメント内のテキストの見出しにブックマークを追加します。次に、これらのブックマークを使用して TOC を生成します。

上記のコードでは、BuildRandomParagraphとBuildRandomTitleの2つのメソッドを使用しています。これらのメソッドには、次のコードのように指定された形式で文字列の配列およびreturn文字列がふくまれています。同様に、独自のメソッドや文字列値を作成することもできます。

Visual Basic

```
Private Shared Function BuildRandomTitle() As String
    Dim a1 As String() =
"Learning|Explaining|Mastering|Forgetting|Examining|Understanding|Applying|Using|Destroying".Split("|"C)

    Dim a2 As String() = "Music|Tennis|Golf|Zen|Diving|Modern
Art|Gardening|Architecture|Mathematics|Investments|.NET|Java".Split("|"C)
    Dim a3 As String() = "Quickly|Painlessly|The Hard Way|Slowly|Painfully|With Panache".Split("|"C)
    Return String.Format("{0} {1} {2}", a1(rnd.[Next](a1.Length - 1)), a2(rnd.[Next](a2.Length - 1)),
a3(rnd.[Next](a3.Length - 1)))
End Function

Private Shared Function BuildRandomParagraph() As String
    Dim sb As New StringBuilder()
    For i As Integer = 0 To 5 + (rnd.[Next](10) - 1)
        sb.AppendFormat(BuildRandomSentence())
    Next
    Return sb.ToString()
End Function

Private Shared Function BuildRandomSentence() As String
    Dim a1 As String() = "Artists|Movie stars|Musicians|Politicians|Computer programmers|Modern
thinkers|Gardeners|Experts|Some people|Hockey players".Split("|"C)
    Dim a2 As String() = "know|seem to think about|care about|often discuss|dream
about|hate|love|despise|respect|long for|pay attention to|embrace".Split("|"C)
    Dim a3 As String() = "the movies|chicken soup|tea|many things|sushi|my car|deep thoughs|tasteless
jokes|vaporware|cell phones|hot dogs|ballgames".Split("|"C)
    Dim a4 As String() = "incessantly|too much|easily|without
reason|rapidly|sadly|randomly|vigorously|more than usual|with enthusiasm|shamelessly|on
Tuesdays".Split("|"C)
    Return String.Format("{0} {1} {2} {3}. ", a1(rnd.[Next](a1.Length - 1)), a2(rnd.[Next](a2.Length -
1)), a3(rnd.[Next](a3.Length - 1)), a4(rnd.[Next](a4.Length - 1)))
End Function
```

C#

```
static string BuildRandomTitle() {
  string[] a1 =
"Learning|Explaining|Mastering|Forgetting|Examining|Understanding|Applying|Using|Destroying".Split('|');

  string[] a2 = "Music|Tennis|Golf|Zen|Diving|Modern
Art|Gardening|Architecture|Mathematics|Investments|.NET|Java".Split('|');
  string[] a3 = "Quickly|Painlessly|The Hard Way|Slowly|Painfully|With Panache".Split('|');
  return string.Format("{0} {1} {2}", a1[rnd.Next(a1.Length - 1)], a2[rnd.Next(a2.Length - 1)],
a3[rnd.Next(a3.Length - 1)]);
}

static string BuildRandomParagraph() {
  StringBuilder sb = new StringBuilder();
  for (int i = 0; i < 5 + rnd.Next(10); i++) {
    sb.AppendFormat(BuildRandomSentence());
  }
  return sb.ToString();
}

static string BuildRandomSentence() {
  string[] a1 = "Artists|Movie stars|Musicians|Politicians|Computer programmers|Modern
thinkers|Gardeners|Experts|Some people|Hockey players".Split('|');
  string[] a2 = "know|seem to think about|care about|often discuss|dream
about|hate|love|despise|respect|long for|pay attention to|embrace".Split('|');
  string[] a3 = "the movies|chicken soup|tea|many things|sushi|my car|deep thoughs|tasteless
jokes|vaporware|cell phones|hot dogs|ballgames".Split('|');
  string[] a4 = "incessantly|too much|easily|without reason|rapidly|sadly|randomly|vigorously|more
than usual|with enthusiasm|shamelessly|on Tuesdays".Split('|');
  return string.Format("{0} {1} {2} {3}. ", a1[rnd.Next(a1.Length - 1)], a2[rnd.Next(a2.Length - 1)],
a3[rnd.Next(a3.Length - 1)], a4[rnd.Next(a4.Length - 1)]);
}
```

上記のコードの出力は、次の図のようになります。

さまざまな用紙サイズの Word ドキュメントを作成

Word for WPF では、さまざまな用紙サイズで Word ドキュメントを作成できます。**PaperKind** 列挙を使用して、利用可能な標準用紙サイズを指定できます。

次のコードでは、**WordUtils** という名前のクラスを使用します。このクラスは、システムの次の場所にある製品サンプル内に置かれています。
Documents\ComponentOne Samples\WPF\WordCreator
これらのクラスを上記の場所からアプリケーションで使用できます。

PaperKind 列挙は、次のコードで実装されます。

Visual Basic

```
' 各用紙サイズにつき 1 つのページを作成します
Dim firstPage As Boolean = True
For Each fi As var In GetType(PaperKind).GetFields(BindingFlags.[Static]
Or BindingFlags.[Public])
        ' Silverlight/Phone には Enum.GetValues はありません
        Dim pk As PaperKind = DirectCast(fi.GetValue(Nothing),
PaperKind)

        ' カスタムサイズはスキップします
        If pk = PaperKind.[Custom] Then
                Continue For
        End If

        ' 最初のページ以降のすべてのページに新しいページを追加します
        If Not firstPage Then
                word.PageBreak()
        End If
        firstPage = False
```

```
        ' 用紙の種類と向きを設定します
        'rtf.PaperKind = pk;
        word.Landscape = Not word.Landscape

        ' コンテンツをページに描画します
        rc = WordUtils.PageRectangle(word)
        rc = WordUtils.Inflate(rc, -6, -6)
        Dim text As String = String.Format("PaperKind: [{0}];" & vbCr &
vbLf & "Landscape: [{1}];" & vbCr & vbLf & "Font: [Tahoma 18pt]", pk,
word.Landscape)
        word.DrawString(text, font, Colors.Black, rc, sf)
        word.DrawRectangle(Colors.Black, rc)
Next
```

C#

```
// 各用紙サイズにつき 1 つのページを作成します
bool firstPage = true;
foreach (var fi in typeof(PaperKind).GetFields(BindingFlags.Static |
BindingFlags.Public))
{
     // Silverlight/Phone には Enum.GetValues はありません
     PaperKind pk = (PaperKind)fi.GetValue(null);

     // カスタムサイズはスキップします
     if (pk == PaperKind.Custom) continue;

     // 最初のページ以降のすべてのページに新しいページを追加します
     if (!firstPage) word.PageBreak();
     firstPage = false;

     // 用紙の種類と向きを設定します
     //rtf.PaperKind = pk;
     word.Landscape = !word.Landscape;

     // コンテンツをページに描画します
     rc = WordUtils.PageRectangle(word);
     rc = WordUtils.Inflate(rc, -6, -6);
     string text = string.Format("PaperKind: [{0}];\r\nLandscape:
[{1}];\r\nFont: [Tahoma 18pt]", pk, word.Landscape);
     word.DrawString(text, font, Colors.Black, rc, sf);
     word.DrawRectangle(Colors.Black, rc);
}
```

テキストフローの追加

Word ドキュメントでテキストフローを使用できます。**Word for WPF** を使用して、ドキュメントの段やページにテキストをフロー挿入できます。

次のコードでは、**WordUtils** という名前のクラスを使用します。このクラスは、システムの次の場所にある製品サンプル内に置かれています。
Documents\ComponentOne Samples\WPF\WordCreator
これらのクラスを上記の場所からアプリケーションで使用できます。

次のコードは、**Word for WPF** でテキストフロー機能を使用する方法を示します。

Visual Basic

```
' 長い文字列をリソースファイルからロードします
Dim text As String = "Resource not found..."
Using sr = New StreamReader(DataAccess.GetStream("flow.txt"))
        text = sr.ReadToEnd()
End Using
text = text.Replace(vbTab, "    ")

' PDF ドキュメントを作成します
word.Info.Title = "テキストフロー"
word.LineBreak()

' タイトルを追加します
Dim titleFont As New Font("Tahoma", 24, RtfFontStyle.Bold)
Dim bodyFont As New Font("Tahoma", 9)
Dim rcPage As Rect = WordUtils.PageRectangle(word)
Dim paragraph = New RtfParagraph()
Dim title = New RtfString(word.Info.Title, titleFont,
RtfUnderlineStyle.Dotted)
paragraph.Add(title)
word.Add(paragraph)
word.LineBreak()
word.LineBreak()

' 文字列を複数の段とページにまたがってレンダリングします
For Each s As var In text.Split(New String() {Environment.NewLine},
StringSplitOptions.None)
        word.AddParagraph(s, bodyFont, Colors.Black,
RtfHorizontalAlignment.Justify)
Next
```

C#

```
// 長い文字列をリソースファイルからロードします
string text = "Resource not found...";
using (var sr = new StreamReader(DataAccess.GetStream("flow.txt")))
{
    text = sr.ReadToEnd();
}
text = text.Replace("\t", "    ");

// PDF ドキュメントを作成します
word.Info.Title = "テキストフロー";
word.LineBreak();

// タイトルを追加します
Font titleFont = new Font("Tahoma", 24, RtfFontStyle.Bold);
Font bodyFont = new Font("Tahoma", 9);
Rect rcPage = WordUtils.PageRectangle(word);
var paragraph = new RtfParagraph();
```

```
var title = new RtfString(word.Info.Title, titleFont,
RtfUnderlineStyle.Dotted);
paragraph.Add(title);
word.Add(paragraph);
word.LineBreak();
word.LineBreak();

// 文字列を複数の段とページにまたがってレンダリングします
foreach (var s in text.Split(new string[] { Environment.NewLine },
StringSplitOptions.None))
{
    word.AddParagraph(s, bodyFont, Colors.Black,
RtfHorizontalAlignment.Justify);
}
```

上記のコードの出力は、次の図のようになります。

テキストフロー

Wordの最初のバージョンを発生したのは、ビル・ゲイツの個人的な技術アドバイザーでもあったりチャート・ブロディである。1981年にリチャード・ブロディがマイクロソフトに入社、一年後にブロディはWordのプログラミングを任された（Wordは成功を収めたものの、のちにブロディはマイクロソフトを退社し、著述業へ転身した）。1983年5月、Multi - Tool Wordの名前でXenix向けに発売された。この最初のWordは、同社初のグラフィカルユーザインタフェースを採用した製品であり、Microsoft Mouseという名前のマウス製品が同時発売となった。初期のWindowsは、この初代Wordで採用されていたインターフェイスを採用しており、このWordを開発する際に構築された開発ライブラリ名がWindowsと呼ばれていたとされる。翌84年1月にはMicrosoft Word 1.0 for Macを発表した。日本市場においてワープロソフトと言えば、MS - DOS時代からジャストシステムの一太郎が絶対的なシェアを持っており、英語文化圏で開発されたWordは文字数指定や縦書きといった日本語特有の文化に対応した機能を持っておらず、且つ、Microsoft製のWindows用の日本語入力ソフトであるMicrosoft IMEは未熟だったため、Wordは苦戦を強いられていた。また、英語文化圏でもコーレル（当時はノベル社）のWordPerfectがシェアを50 % 以上とっており、現在にあるその地位にはいなかった。ただ、Mac版は日本語化が遅れたため日本国内ではエルゴソフトのEGWORDに押されていたものの、英語文化圏においてクラリス社のMacWriteやNisus社のNisus Writerと並ぶ人気ワープロソフトであった。その後、競合製品の機能を積極的に取り込んだほか、スタイルシートなどのオリジナルの機能も追加して高機能化を推し進めた（このWordオリジナルの機能は逆に競合製品に取り込まれている）。また、日本語独自機能はマイクロソフト（日本法人）が主体として開発するようになり、日本語処理を強化していった。競合他社への情報提供の時間差を利用して自社製OSであるWindows 95の発売と同時に対応バージョンのWord 95を発売し、Excelの人気をテコにバンドルしたセットでPCメーカーにプリインストール販売戦略を推進することでシェアを高めていった。その結果、ライバルのWordPerfectのシェアが当時50%あったものが、コーレル売却時には10%になったため、当時のWordPerfectの開発元であったノベル社はMicrosoftを独占禁止法違反でユタ州連邦地方裁判所に提訴している。ノベル社の主張は、同社が「WordPerfect」と「Quattro Pro」を所有していた期間にMicrosoft社がオフィス向けアプリケーション市場の競争を排除する行為によってノベル社に損害を与えたというものである。現在、シェアはWordが圧倒的に優勢となっている。また、日本国内においても、Microsoft Officeのバンドル・プリインストールの際はWordとExcelをセットで販売する方針を強化し、一太郎とExcelといった組み合わせを認めない、と行った手法が横行した。これには1998年11月に公正取引委員会より抱き合わせ販売にあたるとして排除措置命令が出された。98年当時にはすでに「Word 97」の日本語版としての「Word 98」が発売されるほどにまで製品基盤が強化されており、この戦略が定着したものとなっていた。この時、この戦略をなぞる形で「Personal business Edition」が発売されている。Windows用ではWord95、97、98、2000、2002、2003、2007、2010、2013を経て、2015年現在「Word 2016」が最新版である。なお、Word 98は当時評判の悪かった日本語処理の向上、およびライバル製品（一太郎）の存在する日本市場上の戦略により投入された、欧米では発売されていない独自のバージョンである。またWord 98は大韓民国においても朝鮮語版が発売されている（発売の背景は不明）。