ComponentOne

Word for WPF

ComponentOne, a division of GrapeCity
201 South Highland Avenue, Third Floor
Pittsburgh, PA 15206 USA

Website: http://www.componentone.com
Sales: sales@componentone.com
Telephone: 1.800.858.2739 or 1.412.681.4343 (Pittsburgh, PA USA Office)

Trademarks

Warranty

ComponentOne warrants that the media on which the software is delivered is free from defects in material and workmanship, assuming normal use, for a period of 90 days from the date of purchase. If a defect occurs during this time, you may return the defective media to ComponentOne, along with a dated proof of purchase, and ComponentOne will replace it at no charge. After 90 days, you can obtain a replacement for the defective media by sending it and a check for $2 5 (to cover postage and handling) to ComponentOne.

Except for the express warranty of the original media on which the software is delivered is set forth here, ComponentOne makes no other warranties, express or implied. Every attempt has been made to ensure that the information contained in this manual is correct as of the time it was written. ComponentOne is not responsible for any errors or omissions. ComponentOne's liability is limited to the amount you paid for the product. ComponentOne is not liable for any special, consequential, or other damages for any reason.

Copying and Distribution

Table of Contents

Word for WPF Overview

ComponentOne Studio introduces **Word for WPF** with rich API to create Word documents with advanced features. **Word for WPF** creates, reads, writes Word documents with Microsoft Word Open XML format (*.docx) and RTF Documents with Rich Text format (*.rtf) extension.

Word for WPF uses **C1.WPF.Word.C1WordDocument** class, which provides all advanced properties and methods required to generate both Microsoft Word and RTF Documents. The documents generated using **Word for WPF** can be easily stored in file system and exported in standard Microsoft Word document format.

Help with WPF Edition

For information on installing **ComponentOne Studio WPF and Silverlight Edition**, licensing, technical support, namespaces, and creating a project with the controls, please visit Getting Started with WPF Edition.

Word for WPF Key Features

The key features of **Word for WPF** are as follows:

- **Rich Object Model**
  Word for WPF provides a rich and powerful object model which is easily programmable. All you have to use is **C1WordDocument** class that provides all advanced properties and methods, to create both Microsoft Word and RTF Documents.
- **Advanced Library**
  Word for WPF as advanced methods to add images, text, graphics, quotes, and table in Word documents.
- **Different Paper Sizes**
  Create a document with multiple paper sizes and orientation using **Word for WPF**.
- **Add Tables**
  Word for WPF includes RTFTable object that helps you add data to table cells.
- **Add TOC**
  Word for WPF allows adding Table of Content (TOC) to a document containing headers along with the text. It also allows you to move to different pages within the document.
- **Hyperlinks and bookmarks**
  Word for WPF provides you bookmarks to navigate within the document and hyperlinks to navigate to different URLs.
- **Draw Text**
  Word for WPF allows you to draw text in different fonts and use font properties in your Word documents.
- **Text Flow**
  Word for WPF allows you to flow text into columns and pages in a word document.

Object Model Summary

Word for WPF provides a rich and powerful object model which is easily programmable.
The C1.WPF.Word.C1WordDocument class provides all advanced properties and methods to create both Microsoft Word and RTF Documents.

C1WordDocument Object
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 Object
A0, A1, A4, A10, B1, B5, HalfLetter, Legal, Letter
StringFormat
Alignment, Angle, LineAlignment, LineSpacing

Quick Start: Word for WPF

The quick start guide familiarizes you with **Word for WPF**. In this section, you learn to create a new WPF project in Visual Studio. You would require to add the C1Word reference (dll) and a button in the application to create and save a document.

Step 1 of 3: Setting up the Application

You begin with creating a WPF application in Visual Studio, adding **Word** reference and a button control to your application using the following steps:

1. Create a new WPF project in Visual Studio.
2. Add the **C1Word** reference (dll) to your application.
3. Add the following namespaces in the code view:

Visual Basic

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

C#

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

4. Switch to design view and add a Button control to the Form in your application to start using the **C1Word**. Set the **Content** property to a suitable text such as, **Text**, **Name** property to **btnText** and the **Click** event to **btnText_Click**.
5. Double-click the **btnText_Click** event from the properties window.
   The **btnText_Click** event gets created in the code view.

Step 2 of 3: Adding Simple Text

While you are still in code view in the Visual Studio project, add the following code within the **btnText_Click** event:

Visual Basic

```
' create document
Dim word = New C1WordDocument()
word.Clear()

' measure and show some text
Dim text = "Hello!! This is a sample text."
Dim font = New Font("Segoe UI Light", 20, RtfFontStyle.Italic)

' add paragraph
word.AddParagraph(text, font, Colors.BlueViolet,
RtfHorizontalAlignment.Justify)
```

```
' get stream to save to
Dim dlg = New SaveFileDialog()
dlg.FileName = "document"
dlg.DefaultExt = ".docx"
Dim dr = dlg.ShowDialog()
If Not dr.HasValue OrElse Not dr.Value Then
        Return
End If

' save document
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#

```
// create document
var word = new C1WordDocument();
word.Clear();

// measure and show some text
var text = "Hello!! This is a sample text.";
var font = new Font("Segoe UI Light", 20, RtfFontStyle.Italic);

// add paragraph
word.AddParagraph(text, font, Colors.BlueViolet,
RtfHorizontalAlignment.Justify);

// get stream to save to
var dlg = new SaveFileDialog();
dlg.FileName = "document";
dlg.DefaultExt = ".docx";
var dr = dlg.ShowDialog();
if (!dr.HasValue || !dr.Value) {
  return;
}

// save document
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);
```

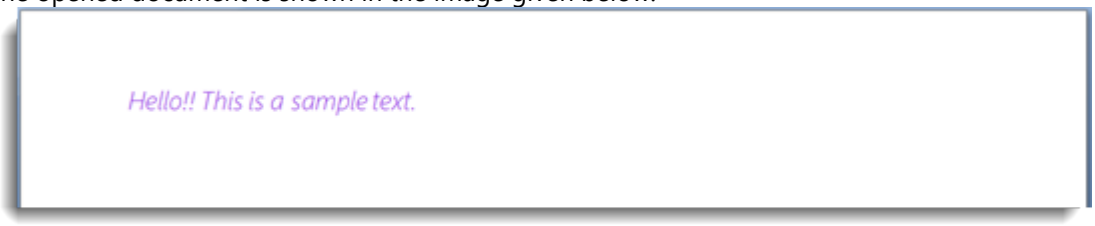
In the above code, a word document is created with some text is added to it using AddParagraph method. It allows you to save the created document to the location of your choice. You can select the desired location and if you want, you can change the name of the document as well. The document is then saved with the name, **document.rtf** or the name you provided it.

Step 3 of 3: Running the Application

In previous step, you added code to the **btnText_Click** event to create, add text and save the word document. In this step, you will run the application and view the document created. Follow the given steps to run the application:

1. Press **F5** to run the application.
2. Click Text Button to view the document you created and saved using **C1Word**.

The opened document is shown in the image given below:



Working with Word for WPF

Word for WPF comes with a rich API and object model that enables you to create word documents as well as RTF documents supported in Microsoft Word and other editors. Understand the working with **Word for WPF** in detail in the topics listed below.

Basic Level Working

Using **Word for WPF**, you can add simple illustrations such as images, graphics, quotes or more to your word document. **Word for WPF** enables you to add these illustrations with few lines of code. Following topics walk you through adding simple text and some basic illustrations.

Adding Text

You can add text to a word document using **C1Word**. You need to write the desired text and use AddParagraph method to add text. You can also set other properties, such as font style, family, color, and more for the text to be displayed in the word document using Font class and its properties. The implementation of adding text to a word document is given in the code below:

1. Create an object of C1WordDocument class with the following code:

Visual Basic

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

C#

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

2. Write the following code to add text to the document:

Visual Basic

```
Dim text = "Hello!! This is a sample text."
Dim font = New Font("Segoe UI Light", 20, RtfFontStyle.Italic)
word.AddParagraph(text, font, Colors.BlueViolet,
RtfHorizontalAlignment.Justify)
```

C#

```
var text = "Hello!! This is a sample text.";
var font = new Font("Segoe UI Light", 20, RtfFontStyle.Italic);
word.AddParagraph(text, font, Colors.BlueViolet,
RtfHorizontalAlignment.Justify);
```

The document will look similar to the image below:

Adding Images

You might need to insert images in your word document along with the text to enhance the overall appearance of your document. To add an image in your document, use the following code that loads an image and sketches it in the document:

Note that two classes named **WordUtils** and **DataAccess** are used in the code given below. These classes are available in the product sample located at the following location on your system:
Documents\ComponentOne Samples\WPF\WordCreator
You can use these classes in your application from the mentioned location.

Visual Basic

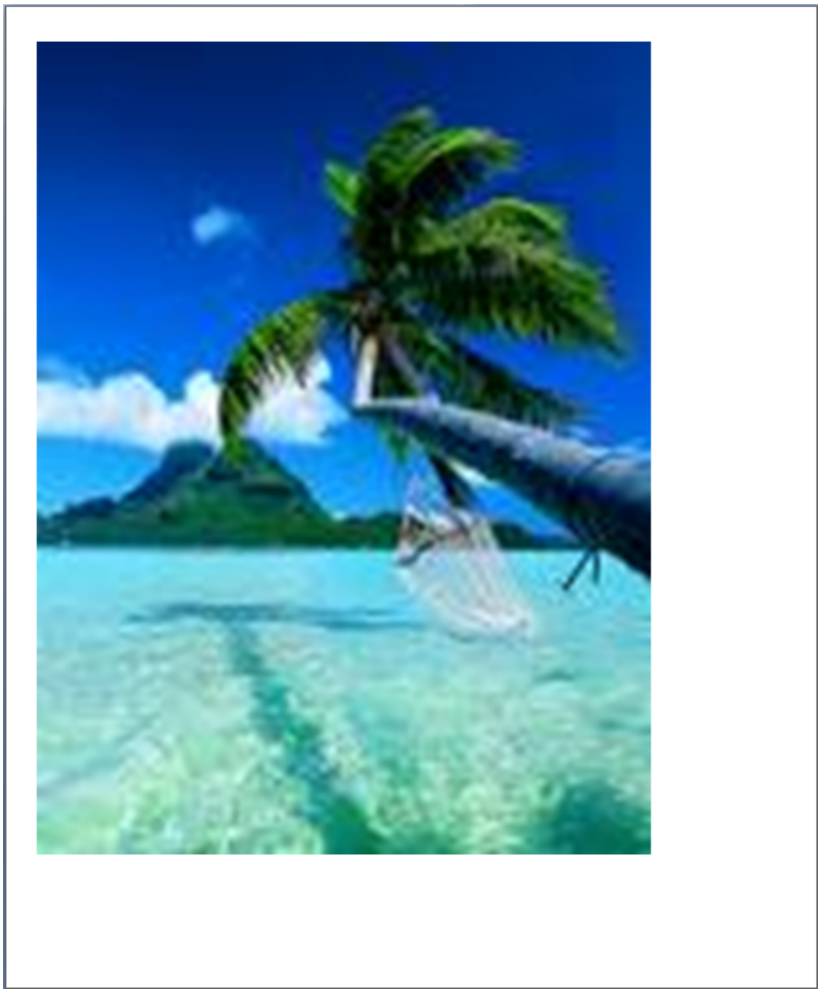
```
' calculate page rect (discounting margins)
Dim rcPage As Rect = WordUtils.PageRectangle(word)

' load image into writeable bitmap
Dim bi As New BitmapImage()
bi.BeginInit()
bi.StreamSource = DataAccess.GetStream("borabora.jpg")
bi.EndInit()
Dim wb = New WriteableBitmap(bi)

' center image on page preserving aspect ratio
word.DrawImage(wb, rcPage)
```

C#

```
// calculate page rect (discounting margins)
Rect rcPage = WordUtils.PageRectangle(word);

// load image into writeable bitmap
BitmapImage bi = new BitmapImage();
bi.BeginInit();
bi.StreamSource = DataAccess.GetStream("borabora.jpg");
bi.EndInit();
var wb = new WriteableBitmap(bi);

// center image on page preserving aspect ratio
word.DrawImage(wb, rcPage);
```

The image is loaded into writeable bitmap and DrawImage method is used to draw the image in the above code.

The output of the above code will look similar to the image given below:



Drawing Graphics

Adding graphics enhances the appearance of a document and make them visually appealing. You can add various types of shapes in your documents such as, Arc, Beizer, Ellipse, Line, Pie, polygon, PolygonLine, and Rectangle. Use the following code to add graphics such as pie, rectangles, polyline, and beziers along with the text:

Visual Basic

```
Dim rtf = New C1WordDocument()
' set up to draw
Dim rc As New Rect(100, 100, 300, 200)
Dim text As String = "Hello world of .NET Graphics and Word/RTF." & vbCr
& vbLf & "Nice to meet you."
Dim font As New Font("Times New Roman", 12, RtfFontStyle.Italic Or
RtfFontStyle.Underline)

' draw to pdf document
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)

' show a Bezier curve
Dim pts = New Point() {New Point(400, 200), New Point(420, 130), New
Point(500, 240), New Point(530, 120)}

' draw Bezier
rtf.DrawBeziers(New Pen(Colors.Blue, 4), pts)

' show Bezier control points
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

' title
rtf.DrawString("Simple Bezier", font, Colors.Black, New Rect(500, 150,
100, 100))
```

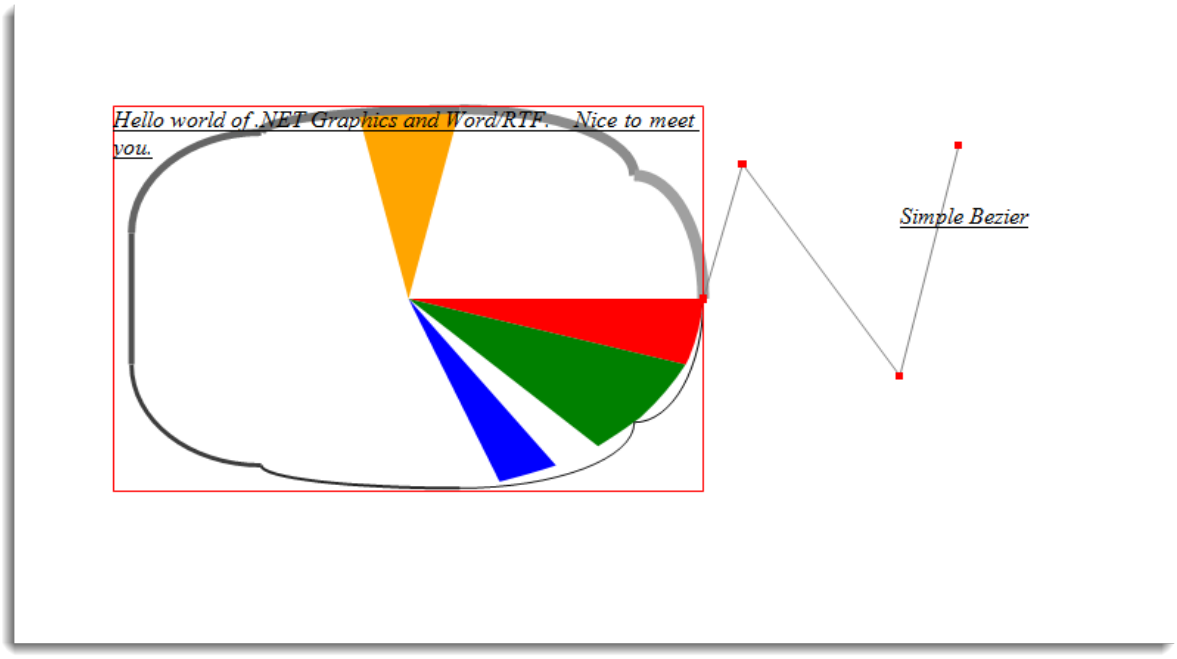
C#

```
var rtf = new C1WordDocument();
// set up to draw
Rect rc = new Rect(100, 100, 300, 200);
string text = "Hello world of .NET Graphics and Word/RTF.\r\nNice to
meet you.";
Font font = new Font("Times New Roman", 12, RtfFontStyle.Italic |
RtfFontStyle.Underline);

// draw to pdf document
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);

// show a Bezier curve
var pts = new Point[]
{
        new Point(400, 200), new Point(420, 130),
        new Point(500, 240), new Point(530, 120),
};

// draw Bezier
rtf.DrawBeziers(new Pen(Colors.Blue, 4), pts);

// show Bezier control points
rtf.DrawPolyline(Colors.Gray, pts);
foreach (Point pt in pts)
{
    rtf.FillRectangle(Colors.Red, pt.X - 2, pt.Y - 2, 4, 4);
}

// title
rtf.DrawString("Simple Bezier", font, Colors.Black, new Rect(500, 150,
100, 100));
```

In the above code, DrawRectangle, DrawArc, DrawPolyline, and DrawBeziers methods are used to draw graphics of different types such as, line, rectangle, pie, and bezier. Besides, DrawString method is used to draw and show text in the document.

The output of the above code will look similar to the image given below:



Adding Quotes

You can add quotes from another file in your document using **C1Word**. You can use following code to add quotes to your document from a text file:

Note that two classes named **WordUtils** and **DataAccess** are used in the code given below. These classes are available in the product sample located at the following location on your system:
Documents\ComponentOne Samples\WPF\WordCreator
You can use these classes in your application from the mentioned location.

Visual Basic

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

' add title
Dim rcTop = WordUtils.RenderParagraph(word, word.Info.Title, titleFont,
rcPage, rc)
rc = rcTop
' build document
For Each s As String In GetQuotes()
        Dim authorQuote As String() = s.Split(ControlChars.Tab)

        ' render header (author)
        Dim author = authorQuote(0)
        rc.Y += 25
        rc = WordUtils.RenderParagraph(word, author, hdrFont, rcPage,
rc, True)

        ' render body text (quote)
        Dim text As String = authorQuote(1)
        rc.X = rcPage.X + 36
        ' << indent body text by 1/2 inch
        rc.Width = rcPage.Width - 40
        rc = WordUtils.RenderParagraph(word, text, txtFont, rcPage, rc)
        rc.X = rcPage.X
        ' << restore indent
        rc.Width = rcPage.Width
        rc.Y += 12
        ' << add 12pt spacing after each quote
        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
```

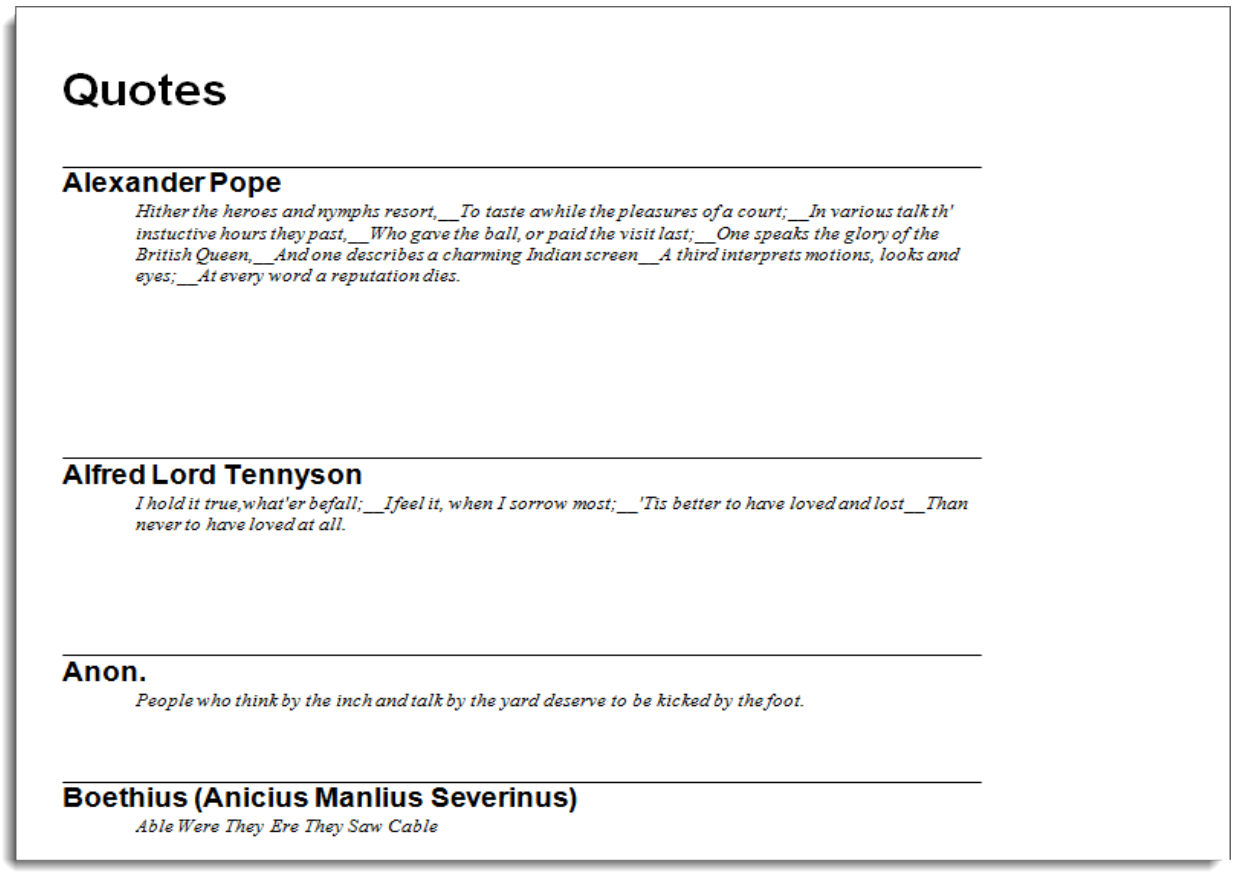
C#

```
// calculate page rect (discounting margins)
Rect rcPage = WordUtils.PageRectangle(word);
Rect rc = rcPage;

// initialize output parameters
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);

// add title
var rcTop = WordUtils.RenderParagraph(word, word.Info.Title, titleFont,
rcPage, rc);
rc = rcTop;

// build document
foreach (string s in GetQuotes())
{
    string[] authorQuote = s.Split('\t');

    // render header (author)
    var author = authorQuote[0];
    rc.Y += 25;
    rc = WordUtils.RenderParagraph(word, author, hdrFont, rcPage, rc,
true);

    // render body text (quote)
    string text = authorQuote[1];
    rc.X = rcPage.X + 36; // << indent body text by 1/2 inch
    rc.Width = rcPage.Width - 40;
    rc = WordUtils.RenderParagraph(word, text, txtFont, rcPage, rc);
    rc.X = rcPage.X; // << restore indent
    rc.Width = rcPage.Width;
    rc.Y += 12; // << add 12pt spacing after each quote
    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;
}
```

The above code reads the quotes from a text file and writes them in a document. It adds title to the document at first, then renders the header and body text, and writes the text in the document.

The output of the above code will look similar to the image given below:



Quotes

Alexander Pope

Hither the heroes and nymphs resort,__To taste awhile the pleasures of a court;__In various talk th' instuctive hours they past,__Who gave the ball, or paid the visit last;__One speaks the glory of the British Queen,__And one describes a charming Indian screen__A third interprets motions, looks and eyes;__At every word a reputation dies.

Alfred Lord Tennyson

I hold it true,what'er befall;__I feel it, when I sorrow most;__'Tis better to have loved and lost__Than never to have loved at all.

Anon.

People who think by the inch and talk by the yard deserve to be kicked by the foot.

Boethius (Anicius Manlius Severinus)

Able Were They Ere They Saw Cable

Advanced Level Working

Usually a word document contains text and adding images, illustrations, tables or metafiles to the document makes it look visually appealing and more descriptive. The listed topics will walk you through adding these advanced features to your word document using **Word for WPF**:

Inserting Table

Table is used in word documents to present data in a well formatted form with the help of rows and columns.

Note that two classes named **WordUtils** and **DataAccess** are used in the code given below. These classes are available in the product sample located at the following location on your system:
Documents\ComponentOne Samples\WPF\WordCreator
You can use these classes in your application from the mentioned location.

Word for WPF enables you to add structure to a word document using the following code:

Visual Basic

```
' get the data
Dim ds = DataAccess.GetDataSet()

' calculate page rect (discounting margins)
Dim rcPage As Rect = WordUtils.PageRectangle(word)
Dim rc As Rect = rcPage
```

```
' add title
Dim titleFont As New Font("Tahoma", 24, RtfFontStyle.Bold)
rc = WordUtils.RenderParagraph(word, word.Info.Title, titleFont, rcPage,
rc, False)

' render some tables
RenderTable(word, rc, rcPage, ds.Tables("Customers"), New String()
{"CompanyName", "ContactName", "Country", "Address", "Phone"})
```

C#

```
// get the data
var ds = DataAccess.GetDataSet();

// calculate page rect (discounting margins)
Rect rcPage = WordUtils.PageRectangle(word);
Rect rc = rcPage;

// add title
Font titleFont = new Font("Tahoma", 24, RtfFontStyle.Bold);
rc = WordUtils.RenderParagraph(word, word.Info.Title, titleFont, rcPage,
rc, false);

// render some tables
RenderTable(word, rc, rcPage, ds.Tables["Customers"], new string[]
{ "CompanyName", "ContactName", "Country", "Address", "Phone" });
```

The following code can be used to select the font for the content of the table and build the table:

Visual Basic

```
' select fonts
Dim hdrFont As New Font("Tahoma", 10, RtfFontStyle.Bold)
Dim txtFont As New Font("Tahoma", 8)

' build table
'word.AddBookmark(table.TableName, 0, rc.Y);
rc = WordUtils.RenderParagraph(word, "NorthWind " + table.TableName,
hdrFont, rcPage, rc, False)

' build table
rc = RenderTableHeader(word, hdrFont, rc, fields)
For Each dr As DataRow In table.Rows
        rc = RenderTableRow(word, txtFont, hdrFont, rcPage, rc, fields,
_
                dr)
Next

' done
Return rc
```

C#

```
// select fonts
Font hdrFont = new Font("Tahoma", 10, RtfFontStyle.Bold);
Font txtFont = new Font("Tahoma", 8);

// build table
//word.AddBookmark(table.TableName, 0, rc.Y);
rc = WordUtils.RenderParagraph(word, "NorthWind " + table.TableName,
hdrFont, rcPage, rc, false);

// build table
rc = RenderTableHeader(word, hdrFont, rc, fields);
foreach (DataRow dr in table.Rows)
{
     rc = RenderTableRow(word, txtFont, hdrFont, rcPage, rc, fields,
dr);
}

// done
return rc;
```

After building the table, you need to cell height and width of the table headers and rows, and then render them using the following code:

Visual Basic

```
Private Shared Function RenderTableHeader(word As C1WordDocument, font
As Font, rc As Rect, fields As String()) As Rect
        ' calculate cell width (same for all columns)
        Dim rcCell As Rect = rc
        rcCell.Width = rc.Width / fields.Length
        rcCell.Height = 0

        ' calculate cell height (max of all columns)
        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
        ' add 6 point margin
        ' render header cells
        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

        ' update rectangle and return it
        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
        ' calculate cell width (same for all columns)
        Dim rcCell As Rect = rc
        rcCell.Width = rc.Width / fields.Length
        rcCell.Height = 0

        ' calculate cell height (max of all columns)
        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)
        ' add 4 point margin
        rcCell.Height += 2

        ' break page if we have to
        If rcCell.Bottom > rcPage.Bottom Then
                word.PageBreak()
                rc = RenderTableHeader(word, hdrFont, rcPage, fields)
                rcCell.Y = rc.Y
        End If

        ' center vertically just to show how
        Dim fmt As New StringFormat()
        fmt.LineAlignment = VerticalAlignment.Center

        ' render data cells
        For Each field As String In fields

                ' get content
                Dim text As String = dr(field).ToString()

                ' set horizontal alignment
                Dim d As Double
                fmt.Alignment = If((Double.TryParse(text,
NumberStyles.Any, CultureInfo.CurrentCulture, d)),
HorizontalAlignment.Right, HorizontalAlignment.Left)

                ' render cell
                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
```

```
        ' update rectangle and return it
        Return WordUtils.Offset(rc, 0, rcCell.Height)
End Function
```

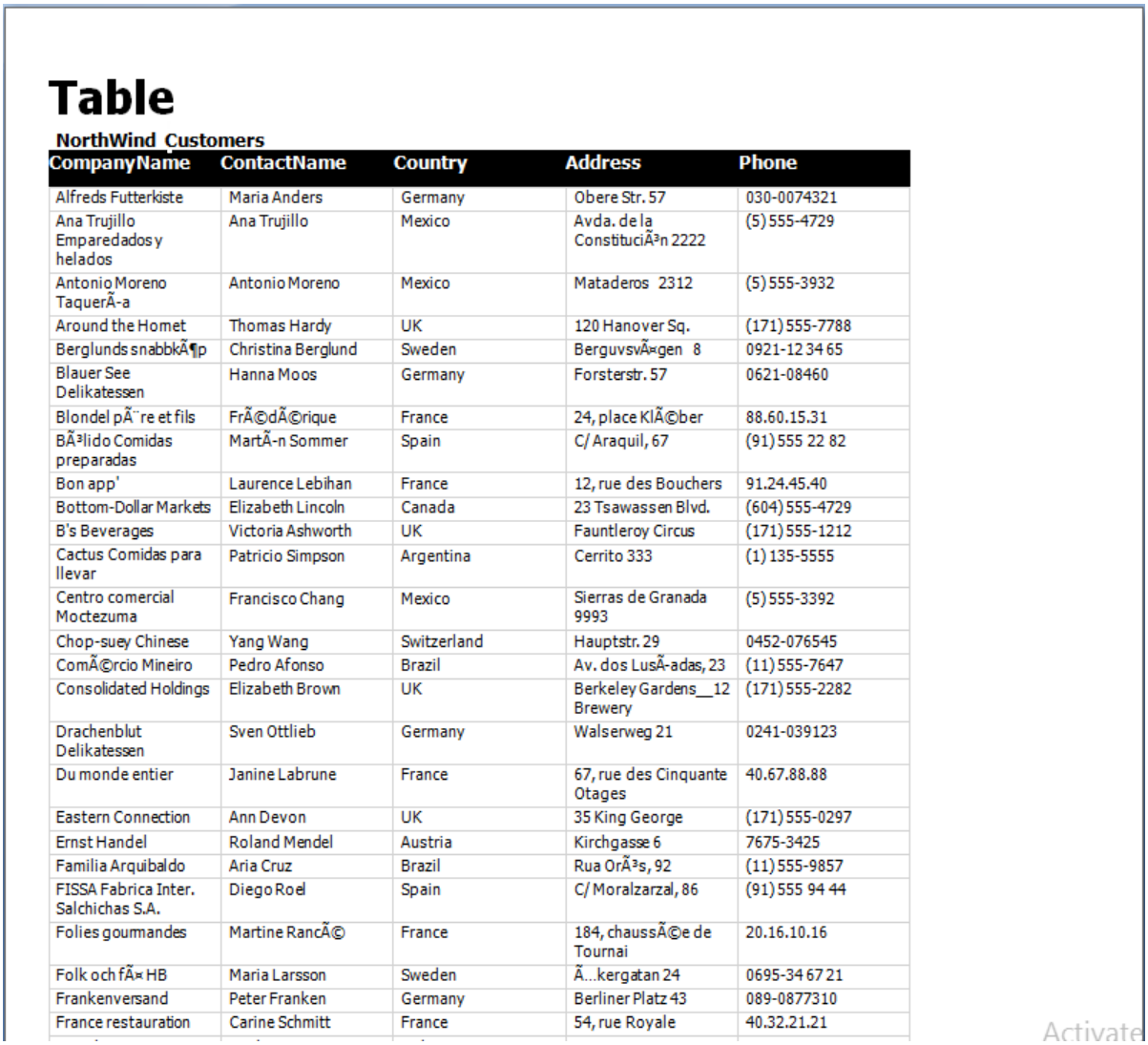
C#

```
static Rect RenderTableHeader(C1WordDocument word, Font font, Rect rc,
string[] fields)
{
// calculate cell width (same for all columns)
Rect rcCell = rc;
rcCell.Width = rc.Width / fields.Length;
rcCell.Height = 0;

// calculate cell height (max of all columns)
foreach (string field in fields)
{
     var height = word.MeasureString(field, font, rcCell.Width).Height;
     rcCell.Height = Math.Max(rcCell.Height, height);
}
rcCell.Height += 6; // add 6 point margin

// render header cells
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);
}

// update rectangle and return it
      return WordUtils.Offset(rc, 0, rcCell.Height);
}

static Rect RenderTableRow(C1WordDocument word, Font font, Font hdrFont,
Rect rcPage, Rect rc, string[] fields, DataRow dr)
{
// calculate cell width (same for all columns)
Rect rcCell = rc;
rcCell.Width = rc.Width / fields.Length;
rcCell.Height = 0;

// calculate cell height (max of all columns)
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); // add 4 point margin
rcCell.Height += 2;

// break page if we have to
if (rcCell.Bottom > rcPage.Bottom)
{
     word.PageBreak();
     rc = RenderTableHeader(word, hdrFont, rcPage, fields);
     rcCell.Y = rc.Y;
}

// center vertically just to show how
StringFormat fmt = new StringFormat();
fmt.LineAlignment = VerticalAlignment.Center;

// render data cells
foreach (string field in fields)
{

        // get content
    string text = dr[field].ToString();

    // set horizontal alignment
    double d;
    fmt.Alignment = (double.TryParse(text, NumberStyles.Any,
CultureInfo.CurrentCulture, out d))
    ? HorizontalAlignment.Right
    : HorizontalAlignment.Left;

// render cell
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);
}

// update rectangle and return it
return WordUtils.Offset(rc, 0, rcCell.Height);
}
```

The above code creates a table from the information taken from the Nwind database with proper indentation and alignment in a word document.

The output of the above code will look similar to the image given below:



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

Adding TOC

Word for WPF enables you to add TOC to a word document containing text with the respective headers. These headers can then be used to create a TOC.

Note that a class named **WordUtils** is used in the code given below. It is available in the product sample located at the following location on your system:
Documents\ComponentOne Samples\WPF\WordCreator
You can use this class in your application from the mentioned location.

Add the following lines of code inside the deriving Window class, to generate a random number and use a string reference:

Visual Basic

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

C#

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

The following code creates content and a TOC for a word document with text and headers in it:

Visual Basic

```
' create document
Dim word = New C1WordDocument()
word.Clear()
```

```
word.Info.Title = "Document with Table of Contents"

' add title
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

' create nonsense document
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
        ' create ith header (as a link target and outline entry)
        Dim header As String = String.Format("{0}. {1}", i + 1, BuildRandomTitle())
        rc = WordUtils.RenderParagraph(word, header, headerFont, rcPage, rc, True, _
                True)

        ' save bookmark to build TOC later
        Dim pageNumber As Integer = 1
        bkmk.Add(New String() {pageNumber.ToString(), header})

        ' create some text
        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

' start Table of Contents
word.PageBreak()
' start TOC on a new page
rc = WordUtils.RenderParagraph(word, "Table of Contents", titleFont, rcPage, rcPage, True)
rc.Y += 12
rc.X += 30
rc.Width -= 40

' render Table of Contents
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
        ' get bookmark info
        Dim page As String = entry(0)
        Dim header As String = entry(1)

        ' render header name and page number
        word.DrawString(header, bodyFont, Colors.Black, rc)
        word.DrawString(page, bodyFont, Colors.Black, rc, sfRight)


        ' connect the two with some dots (looks better than a dotted line)
        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

        ' move on to next entry
        rc = WordUtils.Offset(rc, 0, rc.Height)
        If rc.Bottom > rcPage.Bottom Then
                word.PageBreak()
                rc.Y = rcPage.Y
        End If
Next

' get stream to save to
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

' save document
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#

```
// create document
var word = new C1WordDocument();
word.Clear();
word.Info.Title = "Document with Table of Contents";

// add title
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;

// create nonsense document
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++)
{
    // create ith header (as a link target and outline entry)
    string header = string.Format("{0}. {1}", i + 1, BuildRandomTitle());
    rc = WordUtils.RenderParagraph(word, header, headerFont, rcPage, rc, true, true);

     // save bookmark to build TOC later
     int pageNumber = 1;
     bkmk.Add(new string[] { pageNumber.ToString(), header });

     // create some text
     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;
}

// start Table of Contents
word.PageBreak();                    // start TOC on a new page
rc = WordUtils.RenderParagraph(word, "Table of Contents", titleFont, rcPage, rcPage, true);
rc.Y += 12;
```

```
rc.X += 30;
rc.Width -= 40;

// render Table of Contents
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)
{
     // get bookmark info
     string page = entry[0];
     string header = entry[1];

      // render header name and page number
      word.DrawString(header, bodyFont, Colors.Black, rc);
      word.DrawString(page, bodyFont, Colors.Black, rc, sfRight);


      // connect the two with some dots (looks better than a dotted line)
      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;

      // move on to next entry
      rc = WordUtils.Offset(rc, 0, rc.Height);
      if (rc.Bottom > rcPage.Bottom)
      {
           word.PageBreak();
           rc.Y = rcPage.Y;
      }
}

// get stream to save to
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;
}

// save document
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);
```

The above code adds bookmark to the headers of the text in document. These bookmarks are then used to generate a TOC.

As you can see in the above code, we have used two methods, BuildRandomParagraph and BuildRandomTitle. These methods contain array of strings and return these strings in specified format as shown in the following code. Similarly, you can create your own methods and string values.

Visual Basic

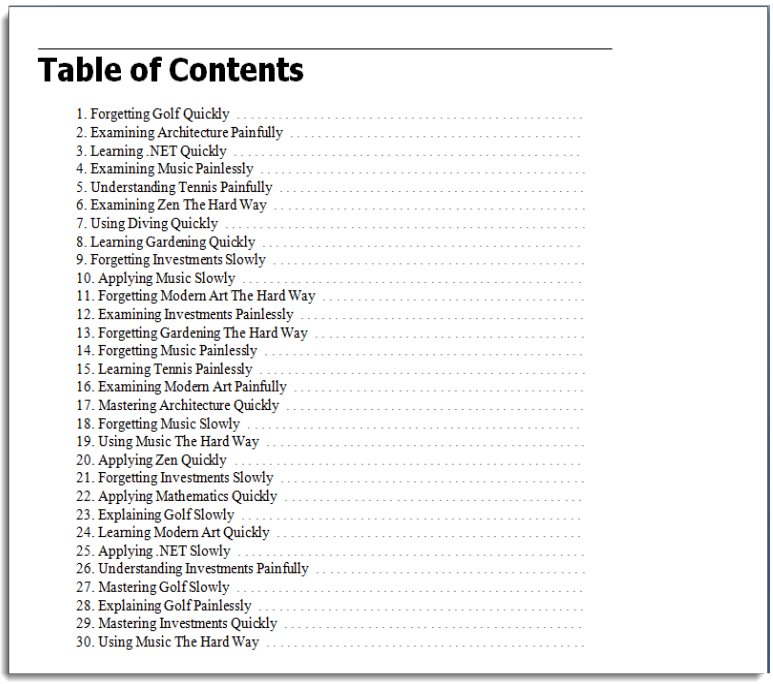
```
Private Shared Function BuildRandomTitle() As String
    Dim a1 As String() =
"Learning|Explaining|Mastering|Forgetting|Examining|Understanding|Applying|Using|Destroying".Split("|"C)

    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)]);
}
```

The output will look similar to the image given below:



Creating Word Document with different paper sizes

Word for WPF gives you the flexibility to create a word document with different paper sizes. You can use PaperKind enum to specify any of the available standard paper sizes.

Note that a class named **WordUtils** is used in the code given below. It is available in the product sample located at the following location on your system:
Documents\ComponentOne Samples\WPF\WordCreator
You can use these classes in your application from the mentioned location.

The implementation of PaperKind enum is given in the following code:

Visual Basic

```
' create one page with each paper size
Dim firstPage As Boolean = True
For Each fi As var In GetType(PaperKind).GetFields(BindingFlags.[Static]
Or BindingFlags.[Public])
        ' Silverlight/Phone doesn't have Enum.GetValues
        Dim pk As PaperKind = DirectCast(fi.GetValue(Nothing),
PaperKind)

        ' skip custom size
        If pk = PaperKind.[Custom] Then
                Continue For
        End If

        ' add new page for every page after the first one
        If Not firstPage Then
                word.PageBreak()
        End If
        firstPage = False
```

```
        ' set paper kind and orientation
        'rtf.PaperKind = pk;
        word.Landscape = Not word.Landscape

        ' draw some content on the page
        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#

```
// create one page with each paper size
bool firstPage = true;
foreach (var fi in typeof(PaperKind).GetFields(BindingFlags.Static |
BindingFlags.Public))
{
     // Silverlight/Phone doesn't have Enum.GetValues
     PaperKind pk = (PaperKind)fi.GetValue(null);

     // skip custom size
     if (pk == PaperKind.Custom) continue;

     // add new page for every page after the first one
     if (!firstPage) word.PageBreak();
     firstPage = false;

     // set paper kind and orientation
     //rtf.PaperKind = pk;
     word.Landscape = !word.Landscape;

     // draw some content on the page
     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);
}
```

Adding Text Flow

You can use text flow in a word document. Using **Word for WPF**, you can flow text into columns and pages of a document.

Note that a class named **WordUtils** is used in the code given below. It is available in the product sample located at the following location on your system:

Documents\ComponentOne Samples\WPF\WordCreator
You can use these classes in your application from the mentioned location.

The following code shows how the text flow feature can be used in **Word for WPF**:

Visual Basic

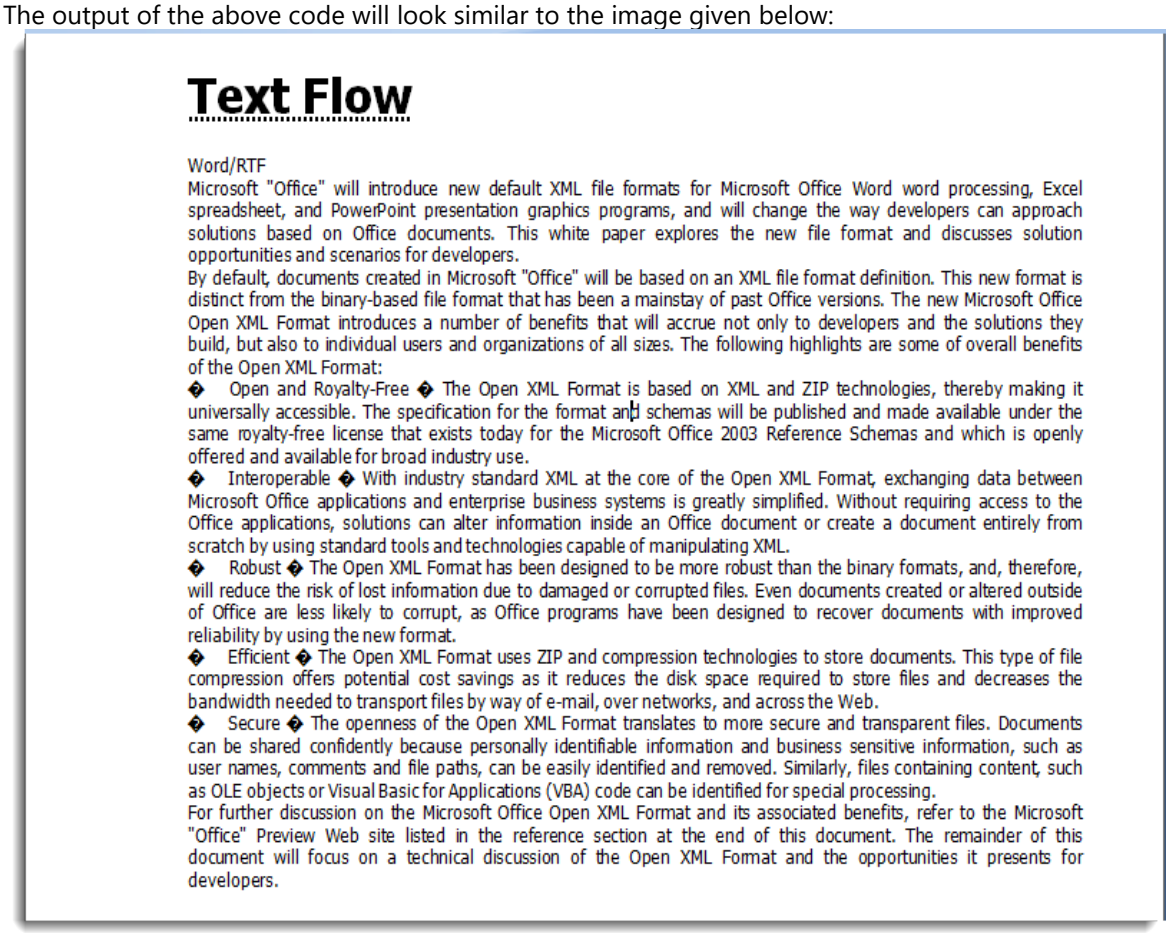
```
' load long string from resource file
Dim text As String = "Resource not found..."
Using sr = New StreamReader(DataAccess.GetStream("flow.txt"))
        text = sr.ReadToEnd()
End Using
text = text.Replace(vbTab, "    ")

' create pdf document
word.Info.Title = "Text Flow"
word.LineBreak()

' add title
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()

' render string spanning columns and pages
For Each s As var In text.Split(New String() {Environment.NewLine},
StringSplitOptions.None)
        word.AddParagraph(s, bodyFont, Colors.Black,
RtfHorizontalAlignment.Justify)
Next
```

C#

```
// load long string from resource file
string text = "Resource not found...";
using (var sr = new StreamReader(DataAccess.GetStream("flow.txt")))
{
    text = sr.ReadToEnd();
}
text = text.Replace("\t", "    ");

// create pdf document
word.Info.Title = "Text Flow";
word.LineBreak();

// add title
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();

// render string spanning columns and pages
foreach (var s in text.Split(new string[] { Environment.NewLine },
StringSplitOptions.None))
{
    word.AddParagraph(s, bodyFont, Colors.Black,
RtfHorizontalAlignment.Justify);
}
```

The output of the above code will look similar to the image given below:



Text Flow

Word/RTF
Microsoft "Office" will introduce new default XML file formats for Microsoft Office Word word processing, Excel spreadsheet, and PowerPoint presentation graphics programs, and will change the way developers can approach solutions based on Office documents. This white paper explores the new file format and discusses solution opportunities and scenarios for developers.
By default, documents created in Microsoft "Office" will be based on an XML file format definition. This new format is distinct from the binary-based file format that has been a mainstay of past Office versions. The new Microsoft Office Open XML Format introduces a number of benefits that will accrue not only to developers and the solutions they build, but also to individual users and organizations of all sizes. The following highlights are some of overall benefits of the Open XML Format:
• Open and Royalty-Free • The Open XML Format is based on XML and ZIP technologies, thereby making it universally accessible. The specification for the format and schemas will be published and made available under the same royalty-free license that exists today for the Microsoft Office 2003 Reference Schemas and which is openly offered and available for broad industry use.
• Interoperable • With industry standard XML at the core of the Open XML Format, exchanging data between Microsoft Office applications and enterprise business systems is greatly simplified. Without requiring access to the Office applications, solutions can alter information inside an Office document or create a document entirely from scratch by using standard tools and technologies capable of manipulating XML.
• Robust • The Open XML Format has been designed to be more robust than the binary formats, and, therefore, will reduce the risk of lost information due to damaged or corrupted files. Even documents created or altered outside of Office are less likely to corrupt, as Office programs have been designed to recover documents with improved reliability by using the new format.
• Efficient • The Open XML Format uses ZIP and compression technologies to store documents. This type of file compression offers potential cost savings as it reduces the disk space required to store files and decreases the bandwidth needed to transport files by way of e-mail, over networks, and across the Web.
• Secure • The openness of the Open XML Format translates to more secure and transparent files. Documents can be shared confidently because personally identifiable information and business sensitive information, such as user names, comments and file paths, can be easily identified and removed. Similarly, files containing content, such as OLE objects or Visual Basic for Applications (VBA) code can be identified for special processing.
For further discussion on the Microsoft Office Open XML Format and its associated benefits, refer to the Microsoft "Office" Preview Web site listed in the reference section at the end of this document. The remainder of this document will focus on a technical discussion of the Open XML Format and the opportunities it presents for developers.

Word for WPF Samples

Please be advised that this ComponentOne software tool is accompanied by various sample projects and/or demos which may make use of other development tools included with the ComponentOne Studio. Please refer to the pre-installed product samples through the following path:
Documents\ComponentOne Samples\WPF

The list of samples available for **Word for WPF** is as follows:

Sample	Description
WordCreator	This sample showcases main features of the C1.WPF.Word, such as adding text, images, graphics, quotes, table, TOC, text flow, and creating a document with different paper sizes.