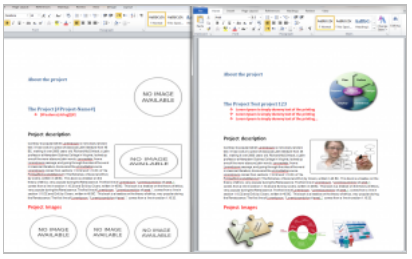
OpenXML Word templates processing

Office Open XML it's the Microsoft's zipped-xml based standard for processing Office documents. It allows to create, amend and process MS office files using e.g. .Net platform. In this article I will show you how to replace Word template with text and images using C# based application.

After creating new project we need to add two main references: DocumentFormat.OpenXml from DocumentFormat.OpenXml.dll and WindowsBase (.Net reference). Next, we need to prepare template with placeholders that our application will replace. For texts we will insert placeholder values in the the unique format that we can find parsing the document e.g. [#Project-Name#].

For the images we need to insert image placeholders (other images) and just remember their original names (e.g. myPicture2.jpg). This names needs to be provided in the parameter objects so we can find the placeholders by image name when parsing document.



The next step is to create parameters structure that we will use when processing the document. You may want to create your own parameters depending or your requirements.

```
1  public class WordParameter
2    {
3        public string Name { get; set; }
4        public string Text { get; set; }
5        public FileInfo Image { get; set; }
6    }
```

We will use them as follows when initiating the objects

```
1      var templ = new WordTemplate();
2      //add parameters
3      templ.WordParameters.Add(new WordParameter() { Name = "[#Project-Name#]", Text = "Test pr
4      templ.WordParameters.Add(new WordParameter() { Name = "[#Features (string[])#]", Text = "
5
6      //original image names to be replaced with the new ones
7      templ.WordParameters.Add(new WordParameter() { Name = "1.jpg", Image = new FileInfo(WordT
8      templ.WordParameters.Add(new WordParameter() { Name = "2.jpg", Image = new FileInfo(WordT
9      templ.WordParameters.Add(new WordParameter() { Name = "3.jpg", Image = new FileInfo(WordT
10     templ.WordParameters.Add(new WordParameter() { Name = "4.jpg", Image = new FileInfo(WordT
11     templ.WordParameters.Add(new WordParameter() { Name = "5.jpg", Image = new FileInfo(WordT
12
13     templ.ParseTemplate(); //create document from template
```

Having done that we can create our main function that parses the template and fills our placeholders with texts and images. Please see the inline comments within the function below.

```
1  public void ParseTemplate()
2  {
3      using (var templateFile = File.Open(templatePath, FileMode.Open, FileAccess.Read)) //read
4      {
5          using (var stream = new MemoryStream())
6          {
7              templateFile.CopyTo(stream); //copy template
8
9              using (var wordDoc = WordprocessingDocument.Open(stream, true)) //open word docum
10             {
11                 foreach (var paragraph in wordDoc.MainDocumentPart.Document.Descendants<Parag
12                 {
```

```
13                  ReplaceImages(wordDoc, paragraph); //replace images
14
15                  ReplaceText(paragraph); //replace text
16              }
17
18              wordDoc.MainDocumentPart.Document.Save(); //save document changes we've made
19          }
20          stream.Seek(0, SeekOrigin.Begin);//scroll to stream start point
21
22          //save file or overwrite it
23          var outPath = WordTemplate.GetRootPath() + @"\Output\DocumentOutput.docx";
24
25          using (var fileStream = File.Create(outPath))
26          {
27              stream.CopyTo(fileStream);
28          }
29      }
30  }
31 }
```

Function that replaces the images. It gets all Blip objects from the paragraph and changes it's embed ID that points to the image.

The cool thing about it is fact that you can give your own styles and transformation to the image template and it will be preserved and applied to the new image


Please see the inline comments.

```
1   void ReplaceImages(WordprocessingDocument wordDoc, Paragraph paragraph)
2   {
3       // get all images in paragraph
4       var imagesToReplace = paragraph.Descendants<A.Blip>().ToList();
5
6       if (imagesToReplace.Any())
7       {
8           var index = 0;//image index within paragraph
9
10          //find all original image names in paragraph
11          var paragraphImageNames = paragraph.Descendants<DocumentFormat.OpenXml.Drawing.Pictur
12
13          //check all images in the paragraph and replace them if it matches our parameter
14          foreach (var imagePlaceHolder in paragraphImageNames)
15          {
16              //check if we have image parameter that matches paragraph image
17              foreach (var param in WordParameters)
18              {
19                  //replace it if found by original image name
20                  if (param.Image != null && param.Image.Name.ToLower() == imagePlaceHolder.Nam
21                  {
22                      var imagePart = wordDoc.MainDocumentPart.AddImagePart(ImagePartType.Jpeg)
23                      using (FileStream imgStream = new FileStream(param.Image.FullName, FileMo
24                      {
25                          imagePart.FeedData(imgStream); //feed it with data
26                      }
27
28                      var relID = wordDoc.MainDocumentPart.GetIdOfPart(imagePart); // get relat
29
30                      imagesToReplace.Skip(index).First().Embed = relID; //assign new relID, sk
31                  }
32              }
33              index += 1;
34          }
35      }
36  }
```

When replacing the texts we check if paragraph contains the text that matches our parameter. If yes, then we check if to include one or multiple lines of text.
Next we create new parameter by copying the old parameter's OuterXML (this preserves the styles). We also need to replace text that is stored in our parameter.

```
1   void ReplaceText(Paragraph paragraph)
2   {
3       var parent = paragraph.Parent; //get parent element - to be used when removing placeholde
4       var dataParam = new WordParameter();
5
6       if (ContainsParam(paragraph, ref dataParam)) //check if paragraph is on our parameter lis
7       {
8           //insert text list
9           if (dataParam.Name.Contains("string[]")) //check if param is a list
```

```
        {
            var arrayText = dataParam.Text.Split(Environment.NewLine.ToCharArray()); //in our

            if (arrayText is IEnumerable) //enumerate if we can
            {
                foreach (var itemData in arrayText)
                {
                    Paragraph bullet = CloneParaGraphWithStyles(paragraph, dataParam.Name, it
                    parent.InsertBefore(bullet, paragraph); //insert new element
                }
            }
            paragraph.Remove();//delete placeholder
        }
        else
        {
            //insert text line
            var param = CloneParaGraphWithStyles(paragraph, dataParam.Name, dataParam.Text);
            parent.InsertBefore(param, paragraph);//insert new element

            paragraph.Remove();//delete placeholder
        }
    }
}
```

Creating the new paragraph object preserving the styles

```
public static Paragraph CloneParaGraphWithStyles(Paragraph sourceParagraph, string paramKey,
{
    var xmlSource = sourceParagraph.OuterXml;

    xmlSource = xmlSource.Replace(paramKey.Trim(), text.Trim());

    return new Paragraph(xmlSource);
}
```

Please note that when replacing images, the image placeholder names must be found in the document. Images that don't match the parameter name, wont be replaced.

Having done that we can finally test our application. I have included complete working application for your tests.

All constructive comments/ questions are welcome!

(3 votes, average: **5.00** out of 5)

Word Templates - complete working solution for Office 2007/2010
download | *3.38 MB*

*Written by **dotNet Geek** on May 12, 2012*
Categories: OpenXML, Win Forms

Share on Twitter Facebook

1 Comment

Related Posts

- Polymorphism in practice (C# sample)

1 Comment

Ale says:
February 14, 2014 at 9:09 am

Thank you so much for the beautiful and useful article!

It works perfectly.

The fact that it maintains the styles of the images and texts is exceptional.

It would be useful to be able to also replace the contents of the textbox.
Do you have any idea about it?

Thank you again.

Reply

Leave a Reply

Your email address will not be published.

Name

Email

Website

Comment

Post Comment

« Finding non empty data periods T-SQL – Database driven rotating image gallery with jQuery »