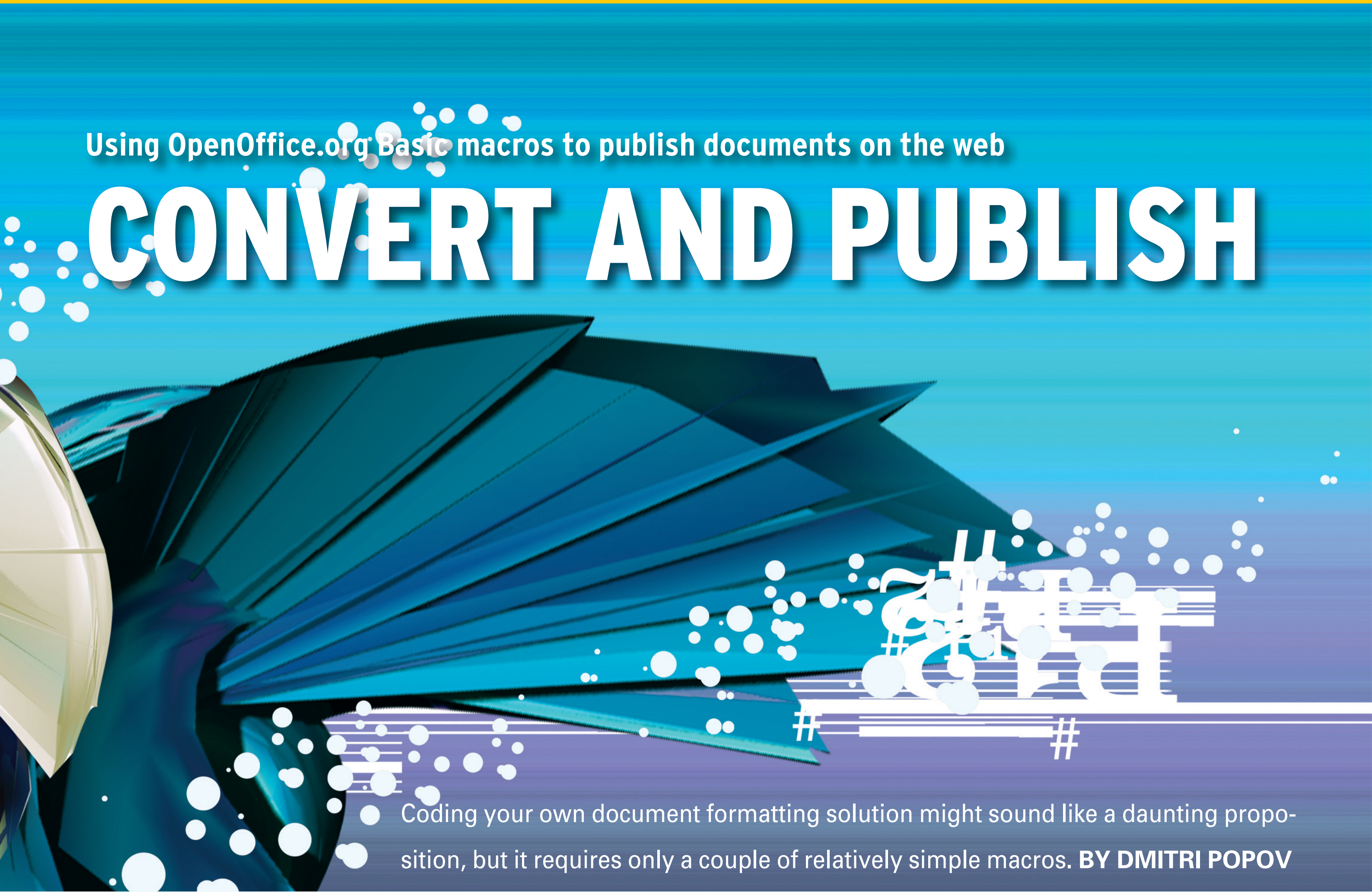


Using OpenOffice.org Basic macros to publish documents on the web

CONVERT AND PUBLISH

Coding your own document formatting solution might sound like a daunting proposition, but it requires only a couple of relatively simple macros. BY DMITRI POPOV

When it comes to publishing your OpenOffice.org Writer documents on the web, you have several options. If you use MediaWiki as your web publishing platform, you can use the Sun Wiki Publisher to convert documents into wiki pages. OpenOffice also lets you save pages in HTML format for use with content management systems that support HTML formatting (the HTML output produced by OpenOffice is far from perfect, though). But what if you are using a publishing platform that uses its own markup? In this case, you might want to consider creating a DIY OpenOffice Basic solution.

To begin, start with a macro that formats the Writer document with a specific markup. For this exercise, I'll use DokuWiki syntax, but you can adapt the final macro to other markup systems. For simplicity, I focus on the core formatting options: headings (from Heading 1 to Heading 5), bold, italic, and underlined, as well as hyperlinks. The way the macro works is simple: It finds all occurrences of formatted text and replaces them with appropriate DokuWiki markup: **text in bold** becomes **text in bold**, *text in italic* becomes //text in italic//, and so on. Because the macro performs a number of similar find and replace operations, it is a perfect candidate for functions. In this case, I need three functions for converting headings, URLs, and formatted text. The function that does the latter is shown in Listing 1.

The function requires three parameters: *SearchAttrName*, *SearchAttrValue*, and *ReplaceStr*. The *SearchAttrName* and *SearchAttrValue* variables find specific formatting in the document. For example, to find text fragments in bold, you should assign the *CharWeight* value to the *SearchAttrName* variable and the *com.sun.star.awt.FontWeight.BOLD* value to the *SearchAttrValue* variable. The *ReplaceStr* variable specifies the appropriate formatting tags wrapped around the strings found. In DokuWiki, the ** tag is used to mark bold text, so in this case, the required value is **&**. Note that the function also uses the statement *ReplaceObj.SearchRegular Expression = true* to enable the regular expressions in the search, and it uses the

Listing 1: Converting Formatted Text

```
Function MarkupTextFunc (SearchAttrName, SearchAttrValue, ReplaceStr)
Dim SearchAttributes(0) As New com.sun.star.beans.PropertyValue
ThisDoc=ThisComponent
SearchAttributes(0).Name=SearchAttrName
SearchAttributes(0).Value=SearchAttrValue
ReplaceObj=ThisDoc.createReplaceDescriptor
ReplaceObj.SearchRegularExpression=true
ReplaceObj.searchStyles=false
ReplaceObj.searchAll=true
ReplaceObj.SetSearchAttributes(SearchAttributes)
ReplaceObj.SearchString=".*"
ReplaceObj.ReplaceString=ReplaceStr
ThisDoc.replaceAll(ReplaceObj)
End Function
```

".*" regular expression to perform the specified search in the document text.

To format headings, I need another function. Headings in Writer documents are usually formatted by paragraph styles, so each heading can be treated as a paragraph. This way, the function can do its job in three steps. First, it identifies paragraphs in the document as objects. Second, for each paragraph object, the function locates the portion formatted with a heading paragraph style, and third, wraps the text portion into the specified tags (Listing 2). This function also requires three parameters: *StyleName*, *StartTag*, and *EndTag*. *StyleName* specifies the paragraph style. In this case, the values that can be assigned to the variable are *Heading 1*, *Heading 2*, *Heading 3*, and so on. Finally, I need one more function to format hyperlinks in the document (Listing 3). As you can see, this function is almost identical to the one in Listing 2. The only difference is that it looks for hyperlinks instead of heading formatting. Also note that the function doesn't require any arguments.

With all functions in place, you can work on the macro. This part is probably the easiest. All you have to do is call the functions and provide them with the appropriate parameters. For example, the statement that converts headings formatted with the *Heading 1* style should be:

```
MarkupHeadingsFunc(➘
   "Heading 1", "====== ", " ======")
```

and the statement to convert text fragments in bold should be:

```
MarkupTextFunc("CharWeight", ➘
   com.sun.star.awt.FontWeight.BOLD, ➘
   "**&**")
```

So the entire macro that processes and converts the current Writer document looks like Listing 4.

Improving the Macro

Although this simple macro does the job, you can improve it in a number of ways. For example, you might want to tweak it so that it saves the converted document as a plain text file. To do this, you need to make several modifications to the original macro. To begin, you have to define a new variable at the beginning of the macro:

```
Dim Args(0) As New com.sun.star.beans.PropertyValue
```

Also, you need to add a code block to check that the document has been saved (i.e., it has a location on the hard disk). This can be done in just three lines of code:

```
If ThisDoc.hasLocation=False Then
  MsgBox ("You have to save the document first!", 64, "Attention") :End
End If
```

Listing 2: Converting Headings

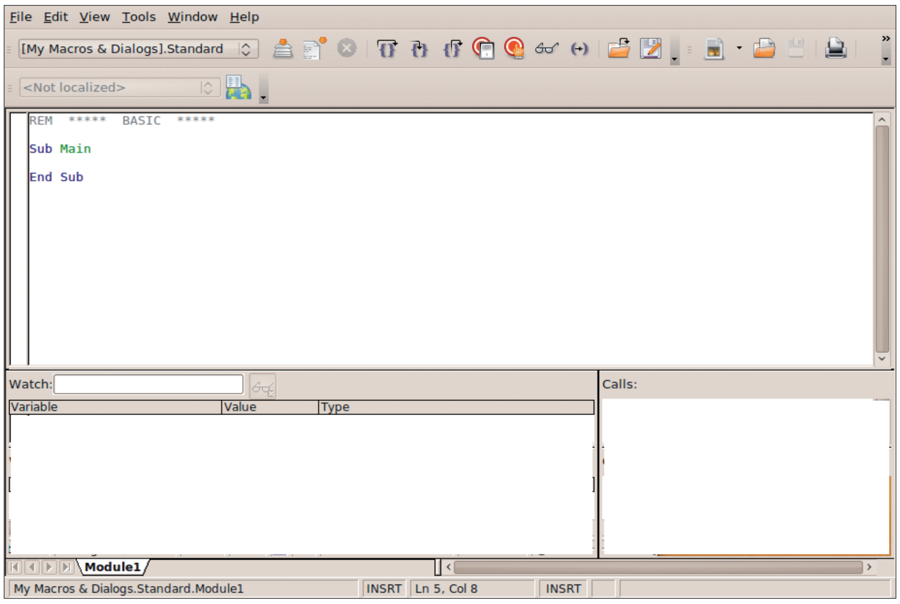
```
Function MarkupHeadingsFunc (StyleName, StartTag, EndTag)
ThisDoc=ThisComponent
ThisText=ThisDoc.Text
ParaEnum=ThisText.createEnumeration
While ParaEnum.hasmoreElements
 Para=ParaEnum.nextElement
  PortionEnum=Para.createEnumeration
   While PortionEnum.hasMoreElements
    Portion=PortionEnum.nextElement
     If Portion.paraStyleName = StyleName Then
       Portion.String = StartTag + Portion.String + EndTag
      End If
   Wend
Wend
End Function
```

Listing 3: Converting Hyperlinks

```
Function MarkupURLFunc()
ThisDoc=ThisComponent
ThisText=ThisDoc.Text
ParaEnum=ThisText.createEnumeration
While ParaEnum.hasmoreElements
 Para=ParaEnum.nextElement
  PortionEnum=Para.createEnumeration
   While PortionEnum.hasMoreElements
    Portion=PortionEnum.nextElement
     If Portion.HyperlinkURL <> "" Then
       Portion.String = "[[" + Portion.HyperlinkURL +"|" +Portion.String + "]]"
      End Jf
   Wend
Wend
End Function
```

Listing 4: Converting the Writer Document

```
Sub Markup
ThisDoc=ThisComponent
MarkupHeadingsFunc("Heading 1", "====== ", " ======")
MarkupHeadingsFunc("Heading 2", "===== ", " =====")
MarkupHeadingsFunc("Heading 3", "==== ", " ====")
MarkupHeadingsFunc("Heading 4", "=== ", " ===")
MarkupHeadingsFunc("Heading 5", "== ", " ==")
MarkupTextFunc("CharWeight", com.sun.star.awt.FontWeight.BOLD, "**&**")
MarkupTextFunc("CharPosture", com.sun.star.awt.FontSlant.ITALIC, "//&//")
MarkupTextFunc("CharUnderline", com.sun.star.awt.FontUnderline.SINGLE, "__&__")
MarkupURLFunc
End Sub
```


File Edit View Tools Window Help
[My Macros & Dialogs].Standard
<Not localized>
Watch:
Variable
Value
Type
Calls:
Module1
My Macros & Dialogs.Standard.Module1
INSRT
Ln 5, Col 8
INSRT

```
REM  *****  BASIC  *****

Sub Main

End Sub
```

Figure 1: To open the macro editor, click on Tools | Macros | Organize Macros | OpenOffice.org Basic, then choose the sample macro (Module1 in this case), and click Edit.

Next, the macro has to initiate the built-in Tools library with the following steps:

```
If (Not GlobalScope.BasicLibraries.isLibraryLoaded("Tools")) Then
  GlobalScope.BasicLibraries.LoadLibrary("Tools")
End If
```

The macro then uses the *DirectoryNameOutOfPath* routine from the Tools library to obtain the directory where the current document is stored:

```
DocURL=ThisDoc.getURL()
DocDir=DirectoryNameOutOfPath(DocURL, "/")
```

Additionally, the macro has to pull the name of the document from the document's path:

```
FileName=Left(Dir(DocURL, 0), Len(Dir(DocURL, 0))-4)
```

Finally, the macro uses the obtained information to save the document as a plaintext file at the same location:

```
Args(0).Name="FilterName"
Args(0).Value="Text"
TextFilePath=ConvertToURL(↩
   DocDir & "/" & FileName & ".txt")
ThisDoc.StoreToURL(↩
  TextFilePath, Args())
```

That's it. Listing 5 is the entire macro.

Adding FTP Upload

The great thing about DokuWiki is that it stores all pages as plaintext *.txt* files. This means that you can upload converted documents without any additional tweaking. Of course, you can do it manually with an FTP client, or you can write another macro that would do this for you. In this way, you can convert, save, and upload any Writer document in one fell swoop. The best part is that writing a macro that does the upload is rather easy because you can reuse most of the code from the conversion macro, as you can see in the FTPUpload macro in Listing 6.

The only difference here is the *FTPPath* = "*ftp://username:password@ftp.server.com/*" statement that specifies the FTP connection string. Obviously, you have to replace the placeholders with an actual username, password, and FTP address for this macro to work properly. If you prefer not to hard-wire the connection string into the macro, you

Listing 5: The Markup Macro

```
Sub Markup
Dim Args(0) As New com.sun.star.beans.PropertyValue
ThisDoc=ThisComponent

 If ThisDoc.hasLocation=False Then
 MsgBox ("You have to save the document first!", 64,
         "Attention") :End
End If

MarkupHeadingsFunc("Heading 1", "====== ", " ======")
MarkupHeadingsFunc("Heading 2", "===== ", " =====")
MarkupHeadingsFunc("Heading 3", "==== ", " ====")
MarkupHeadingsFunc("Heading 4", "=== ", " ===")
MarkupHeadingsFunc("Heading 5", "== ", " ==")
MarkupTextFunc("CharWeight", com.sun.star.awt.FontWeight.
               BOLD, "**&**")
MarkupTextFunc("CharPosture", com.sun.star.awt.FontSlant.
               ITALIC, "//&//")
MarkupTextFunc("CharUnderline", com.sun.star.awt.
               FontUnderline.SINGLE, "__&__")
MarkupURLFunc

If (Not GlobalScope.BasicLibraries.isLibraryLoaded(
    "Tools")) Then
  GlobalScope.BasicLibraries.LoadLibrary("Tools")
End If

DocURL=ThisDoc.getURL()
DocDir=DirectoryNameOutOfPath(DocURL, "/")
FileName=Left(Dir(DocURL, 0), Len(Dir(DocURL, 0))-4)
Args(0).Name="FilterName"
Args(0).Value="Text"
TextFilePath=ConvertToURL(DocDir & "/" & FileName & ".txt")
ThisDoc.StoreToURL(TextFilePath, Args())
End Sub
```

can replace it with an input box that prompts the user to enter the connection string:

```
FTPPath=InputBox("FTP Address", "Input required", ⮐
   "ftp://user:password@192.168.1.7/pub/")
```

Listing 7 shows how to save the connection string in a text file and tweak the macro to read the string (replace *path/to/connectftp.txt* with the filepath): Then, modify the Markup macro to ask the user whether to upload the converted file:

```
Answer=MsgBox("Upload file?",36, "Upload")
If Answer=6 Then
FTPUpload
End If
```

Pressing the *Yes* button triggers the FTPUpload macro.

Final Word

With some tweaking, this macro can work with any markup, for example. The described macros provide basic functionality, so you might want to add support for lists and tables. ■

Listing 6: The FTPUpload Macro

```
Sub FTPUpload
Dim Args(0) As New com.sun.star.beans.PropertyValue

ThisDoc=ThisComponent
If (Not GlobalScope.BasicLibraries.isLibraryLoaded("Tools")) Then
GlobalScope.BasicLibraries.LoadLibrary("Tools")
End If

DocURL=ThisDoc.getURL()
DocDir=DirectoryNameOutOfPath(DocURL, "/")
FileName=Left(Dir(DocURL, 0), Len(Dir(DocURL, 0))-4)

FTPPath="ftp://username:password@ftp.server.com/"
SaveFTP = FTPPath & FileName & ".txt"
Args(0).Name="FilterName"
Args(0).Value="Text"
ThisDoc.storeToURL(SaveFTP, Args())
End Sub
```

Listing 7: Reading a Connection String from a File

```
FTPPath=ConvertToURL ("path/to/connectftp.txt")
f1=FreeFile()
Open FTPPath For Input As #f1
Do While not Eof(f1)
Line Input #f1, FTPString
SaveFTP = FTPString & FileName & ".txt"
Args(0).Name="FilterName"
Args(0).Value="Text"
ThisDoc.storeToURL(SaveFTP, Args())
Loop
```