Style Enhancer Development Report

Christian Gagné

May 31, 2013

Notes

Disclaimer: Please bear with me; many of my notes and comments are very basic and newbie-level, because I am learning the language as I write the extension. I progress slowly, but I gain knowledge and a more fine-grained comprehension day by day.

Status of this document

This document is the development environment for the *Style Enhancer* extension. It also acts as the development report and testing environment for the extension's modules and functions.

The extension consists of a single StarBasic library which is contained in the document's appropriate subdirectory.

After a subroutine has been written, if it does not yield any interesting result and appears to be a dead-end, it is cut from the module and pasted at the end of its entry in the documentation. The subroutines that do work are to be found in the module. The reasons why I do this is that, as I said above, I am a complete programming newbie and, among other things, I get confused easily if a module is too large. I keep my failed subroutines in the documentation to retrace my steps.

The Plan

This extension is to provide the following functionality:

> Searching for occurrences of character styles and replacing or bookmarking them

A dialog box is provided. It displays a list of characters styles defined in the document (all character styles, both application-provided and cutom ones). The user can select one or more character styles and has the option to:

- replace them all with another character style chosen from the list of

available ones;
- boormark them all: the bookmarks created are given as a prefix the name of the style and also contain the string of the portion to which the style is applied.

The replacement or bookmarking can be scoped: the dialog provides the option to replace only the character styles of text portions that are in certain paragraphs (according to paragraph style) or certain sections (according to section name).

> Automatically modifying multiple properties of multiples styles, including setting properties to defaults or the parent style's values (setting inheritance)

Another dialog.

> Grouping styles in simulated "namespaces" by prefixing them and normalizing naming schemes

Yet another dialog.

Ideas and observations

2013-03-07: My current impression is that iteration over text elements is the big success story and that search descriptors are of limited use for what I want to do. I perceive that search descriptors will be useful somehow in limited contexts (for example, single sections or single paragraphs).

Index entries can be created for certain paragraph styles, but not for strings that have a certain character style.

The first significant subroutine on which I am working should add all text portions with a certain character style to a list and export this list of occurrences to a plain text file. Then, anything can be done with this text file (process it with UNIX text tools, search the strings in the document to find all occurrences and apply other styles, specify the occurrences as index entries etc.) In fact, the file could be in the SDI format, which is actually a CSV file intented specifically for index entries in Writer documents.

2013-03-07: In fact, perhaps the extension's user interface itself could be in a document instead of being an unwieldy modal dialog box. All significant output, event messages and the like would be written to the document in appropriate sec-

tions (or written to separate plain text or ODT files to which sections of the UI document would link). At the top of the document, there would be the application's menu, and the tasks' results would be appended to the document, in order, with a timestamp, each in their own section. In this way, the UI document would simultaneously act as a log, but if the actual output is kept in separate linked files this would be even more convenient. If this is done, the UI document would ask the user which open text document should be operated on. The selected document would then become the "ThisComponent", and once the task is complete, the application would switch back to the UI document to write the log. Another benefit: this means that the extension's UI could be *instantiated* for each particular document set on which related tasks must be performed. There would then be a different UI/log document for each document set, each with its own data/configuration directory… In order for all of this to happen, I have to make sure that OO and LibO's plain text and I/O support are robust enough. For example, I am aware of the fact that many problems persist with writing to and reading from SDI concordance files.

Script idea: "If span with *any* character style contains *only* whitespace, un-apply character style but keep said whitespace."

Character style OrgCode has a character color that is different from the parent style's: reset it to the parent's value with a script (`getPropertyState` is your friend).

2013-03-11 20:09:20: The next thing I must try is a function that prints all of a given style's properties with the state of their inheritance: is their value set specifically in this style or inherited from a parent? This function will be a stepping-stone to many others.

Library: StyleEnhancer

Module: styleenhancer

Here is the header of the module on 2013-05-31:

```
' Visual guide for sizing the IDE window,
' so that it does not exceed 80 columns:
'     ->'     ->'     ->'     ->'     ->'     ->'     ->'     ->'     ->'   80->

' Everything gets thrown into this module for now. Appropriate separation
' will comme later.

' 2-space indents are used because some statements can get quite long.
```

```
Option Explicit
```

Subroutine: applyCharacterStyletoNextWord

This subroutine uses a `TextCursor`.

This subroutine currently works and the range of text to which it is applied could be changed by getting a `TextRange` value from another function.

```
sub applyCharacterStyletoNextWord()

  ' Variable declarations
  dim theCurrentDocument As Object, currentDocText As Object
  dim styleCursor As Object

  ' Variable assignments
  theCurrentDocument = ThisComponent
  currentDocText = theCurrentDocument.Text

  ' Task
  styleCursor = currentDocText.createTextCursor
  styleCursor.gotoNextParagraph(false)
  styleCursor.gotoNextWord(false)
  styleCursor.gotoEndOfWord(true)
  styleCursor.CharStyleName = "Emphasis"

end sub
```

Subroutine: findCharStyleinDocument

This subroutine is currently a bunch of ridiculous nonsense. I have yet to understand how to find the occurrences of a *character style* in a document.

Hypothesis: It would seem that a `searchDescriptor` can only search the same stuff that can be searched in the GUI, or properties defined by the `awt` module. The problem is that this module is supposed to be used for GUI development using an "Abstract Window Toolkit" that gives access to OpenOffice's "Visual Component Library". In other words, these properties are part of the frame or the controller, not the model. The only "model" style properties that can be found with a `searchDescriptor` are those of paragraphs.

2013-03-05 16:19:25

Right now, this subroutine only searches for the attribute `CharStyleName` without a value, which returns 0 occurrences. What do you mean zero occurrences? I am positive that there *is* something in my document to which a `CharStyleName` is applied. Update: same thing with `CharStyleNames` instead of `CharStyleName`.

Next attempt: iterate over paragraphs (create enumerations), find text ranges (text portions?) that have the desired character style.

```
sub findCharStyleinDocument()

' Declarations:
  dim theCurrentDocument As Object
  dim mySearchOperation As Object, foundStuff As Variant
  dim appliedCharStyle(0) As New com.sun.star.beans.PropertyValue

' Assignments:
  ' Document object
  theCurrentDocument = ThisComponent

  ' This attribute allows us to check whether content has an applied style
  appliedCharStyle(0).Name = "CharStyleNames"

  ' Search operation
  mySearchOperation = theCurrentDocument.createSearchDescriptor
  mySearchOperation.SearchAttributes = appliedCharStyle()
  mySearchOperation.ValueSearch = False

  foundStuff = theCurrentDocument.findAll(mySearchOperation)

' Print Job:
print "Occurrences: " & foundStuff.Count

End Sub
```

Subroutine: printCharacterStylesofParagraph

This subroutine uses a `TextCursor`.

The property `CharStyleNames` is supposed to yield the names of all character styles applied to the text range in the text cursor, but it currently does not work (Basic gives an error about an unspecified property). How are we to get a list of all the different character styles applied in a paragraph and the strings (text ranges) to which they are applied? When a paragraph has *only one character style* applied to *all of it*, the use of the property `CharStyleName` yields the name of this style, but if there are many styles or if a single is applied only to a subset of the paragraph, the subroutine returns an empty string and the dialog box is empty. This looks very dumb.

```
sub printCharacterStylesofParagraph()

  ' Variable declarations
  dim theCurrentDocument As Object, currentDocText As Object
  dim styleCursor As Object
```

```
  ' Variable assignments
  theCurrentDocument = ThisComponent
  currentDocText = theCurrentDocument.Text

  ' Task
  styleCursor = currentDocText.createTextCursor
  styleCursor.gotoNextParagraph(false)
  styleCursor.gotoEndOfParagraph(true)
  print styleCursor.CharStyleName

end sub
```

Subroutine: ReplaceCharStyleInEnumParPortions

2013-03-06 15:14:11

Success! This is the first thing that actually works as expected, and will be the basis for all further work. From this subroutine, I will create one that prints a list of the styled portions with their corresponding styles, one which replaces styled portions only in paragraphs of a certain style (see next subroutine) and another one which replaces styled portions only in paragraphs of a certain style that contain a certain string. Eventually, I will get the styles to operate with from a dialog box, but that is not for now.

Subroutine: ReplaceCharStyleInStyledEnumParPortions

2013-03-06 15:14:11

This subroutine also works as expected.

Subroutine: ReplaceParaStyleOnlyIfNextParaStyleIsSuch

2013-03-12

14:36:27: I don't yet know whether this can work in the way I expect it to, but what I want to do is replace the paragraph style of all paragraphs with style "A" *only if* the paragraph style of the next paragraph is "B".

15:04:02: Not working right now with

```
TextElementEnum.nextElement.ParaStyleName = ""
```

This has exergue

This also XVIII has exergue

But this has regular text body

This again has exergue

but this has CIERL-Corps-de-texte

The previous anonymous section is a test area for the subroutine.

```
sub ReplaceParaStyleOnlyIfNextParaStyleIsSuch()

dim TheCurrentDoc As Object
dim TextElementEnum As Object
dim TextPortionEnum As Object
dim TextElement As Object
dim TextPortion As Object

TheCurrentDoc = ThisComponent
TextElementEnum = TheCurrentDoc.Text.createEnumeration

while TextElementEnum.hasMoreElements
  TextElement = TextElementEnum.nextElement

  if TextElement.supportsService("com.sun.star.text.Paragraph") then

    if TextElement.ParaStyleName = "CIERL-Exergue" and _
    TextElementEnum.nextElement.ParaStyleName = "CIERL-Exergue" then
      print "There are two adjacent paragraphs with the same style."
'      TextElement.ParaStyleName = "CIERL-Corps-de-texte"

    end if
  end if
wend

end sub
```

Subroutine: ReplaceCharStyleofStringInStyledEnumParPortions

2013-03-12

16:04:25: This works as expected. However, text portions are really only delimited by character properties, so I cannot find a text portion by specifying only a part of the portion's string; I have to provide the string of the whole portion for it to be recognized.

```
sub ReplaceCharStyleofStringInStyledEnumParPortions()

dim TheCurrentDoc As Object
dim TextElementEnum As Object
dim TextPortionEnum As Object
dim TextElement As Object
dim TextPortion As Object

TheCurrentDoc = ThisComponent
TextElementEnum = TheCurrentDoc.Text.createEnumeration

while TextElementEnum.hasMoreElements
  TextElement = TextElementEnum.nextElement
```

```
  if TextElement.supportsService("com.sun.star.text.Paragraph") then

    if TextElement.ParaStyleName = "CIERL-Exergue" then
      TextPortionEnum = TextElement.createEnumeration

      while TextPortionEnum.hasMoreElements
        TextPortion = TextPortionEnum.nextElement

        if TextPortion.String = "This has exergue" then
          TextPortion.CharStyleName = "Bold"
        end if
      wend
    end if
  end if
wend

end sub
```

Subroutine: InsertBookmarksAtCharStyles

2013-05-31

This subroutine uses the same iteration pattern, but creates a bookmark instance, sets its name and inserts it, for each relevant text portion.