Create a Wizard to Manipulate Case

Click & Retrieve Source CODE!

by Chris Barlow

Use the power built into the Word 97 object model to handle mixed-type treatments.

Editor's Note: The goal of this redefined column is to introduce programmers either to basic concepts of Windows development with Visual Basic, or to new capabilities or approaches.

Chris Barlow is president and CEO of SunOpTech, a developer of manufacturing decision-support applications, including the ObjectBank and the ObjectJob Systems. He and Ken Henderson hold U.S. Patent #5,550,976 related to software for decentralized distributed asynchronous object-oriented systems. Chris holds degrees from Harvard Business School and Dartmouth College, where he worked with Drs. Kemeny and Kurtz on the Basic language. Reach Chris on the Internet at ChrisB@SunOpTech.com or through SunOpTech's World Wide Web server at www.SunOpTech.com.

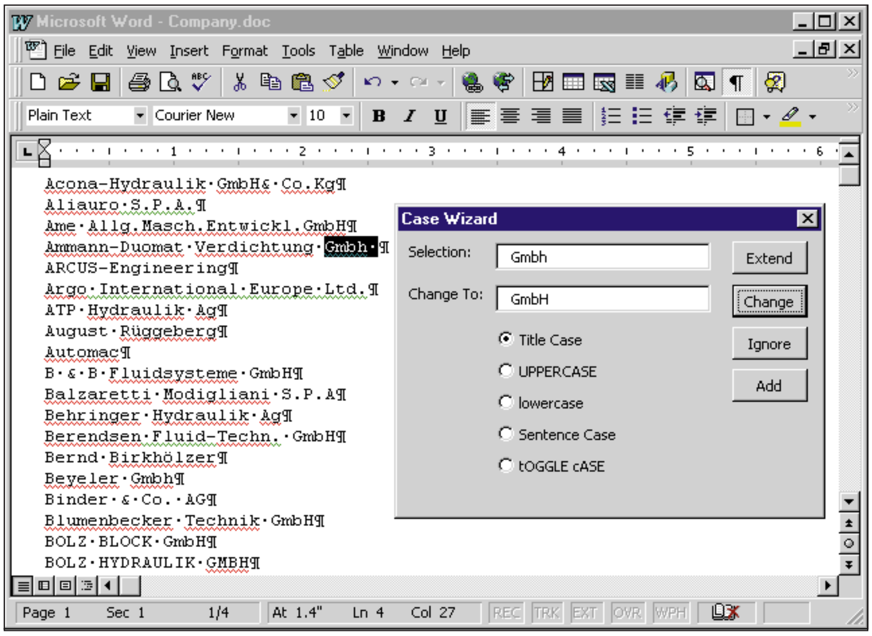
Even if you're just getting started with Visual Basic, you easily can write useful tools to help you and your users be productive. By expanding your knowledge of Visual Basic with knowledge of the powerful object models of Microsoft Office, you can create some cool tools with surprisingly little code.

The goal of this column is to teach general VB programming techniques. As you follow each column and master these techniques, you will learn the Visual Basic statements, file access, database access, and form design that make up a complete Visual Basic application. You will also gain exposure to several different ActiveX servers and learn how to use their object model's properties and methods. This month, you will learn how to use some of the power of the Microsoft Word object model to add a useful function to Word.

The Word team did such a thorough job that it is hard to think of any function that needs to be added. But yesterday, I came across a special function that I needed. At SunOpTech, we develop Visual Basic applications for customers in both the United States and Europe. Like most systems, our ObjectOrder order-management system has a master location file containing customer and vendor information. When converting to the ObjectOrder system, you can import this data from the legacy system that is being replaced.

Often, legacy systems store organization names and addresses in all uppercase characters, regardless of whether the company treats its name and business information in such a manner. This treatment is both inaccurate and unnecessary, thanks to the power of modern Windows systems.

Word and Visual Basic contain functions that change the case of words. Visual Basic


Microsoft Word - Company.doc
File Edit View Insert Format Tools Table Window Help
Plain Text
Courier New
10
Case Wizard
Extend
Change
Ignore
Add
Page 1 Sec 1 1/4 At 1.4" Ln 4 Col 27
REC TRK EXT OVR WPH

```
Acona-Hydraulik·GmbH&·Co.Kg¶
Aliauro·S.P.A.¶
Ame·Allg.Masch.Entwickl.GmbH¶
Ammann-Duomat·Verdichtung·Gmbh·¶
ARCUS-Engineering¶
Argo·International·Europe·Ltd.¶
ATP·Hydraulik·Ag¶
August·Rüggeberg¶
Automac¶
B·&·B·Fluidsysteme·GmbH¶
Balzaretti·Modigliani·S.P.A¶
Behringer·Hydraulik·Ag¶
Berendsen·Fluid-Techn.·GmbH¶
Bernd·Birkhölzer¶
Beyeler·Gmbh¶
Binder·&·Co.·AG¶
Blumenbecker·Technik·GmbH¶
BOLZ·BLOCK·GmbH¶
BOLZ·HYDRAULIK·GMBH¶
```

Selection: Gmbh

Change To: GmbH

[✓ Title Case] UPPERCASE lowercase Sentence Case tOGGLE cASE

FIGURE 1 ***The Case Wizard Form.*** *The new Microsoft Forms engine lets you design forms with any ActiveX control. Notice that the caption of each option button shows the user how the selection will be formatted.*

has two functions, UCase and LCase, that change a text string to all uppercase or all lowercase. In addition to these case conversions, Word provides three case functions. You can convert to a more conventional "title" case, in which the first letter of each word is capitalized. You can convert to "Sentence" case, where only the first word in the sentence has an initial capital letter. Finally, you can simply toggle the case of each character in the selection.

TAKE ADVANTAGE OF THE MACRO RECORDER

You can use the macro recorder to see the Word object model's properties and methods used by the built-in Change Case form. The first step is to turn on the macro recorder, which then records your keystrokes, translates them into VBA, and saves them for you to review, edit, or run later.

The easiest way to turn on the Word macro recorder is to click on the panel labeled REC on the status bar at the bottom of the screen. The Record Macro dialog that appears lets you enter a name and description for the macro, assign it to a toolbar or keyboard shortcut, and decide whether to store it in a template or in this document. Enter TestChangeCase for the macro name and click on the OK button.

After you do so, the image of a small tape recorder appears next to your mouse pointer. You will also see a small toolbar with Stop and Pause buttons. Now click on the Change Case menu item on the Format menu to show Word's Change Case form. Click on the Title Case button, and click on the OK button. Then press the Stop button to stop recording the macro, and press Alt-F11 to jump to the Visual Basic editor. Here's the code for the procedure Sub TestChangeCase():

```
Sub TestChangeCase()
Selection.Range.Case = wdTitleWord
End Sub
```

See how easily you can learn VBA from the macro recorder? By looking at the VBA code written by the macro recorder, you can learn the important parts of the object model and understand how to use them in your own procedures. Notice that the line begins with the Selection object. In Word, if you do not select text, the Selection object returns the text after the current cursor position.

VBA5

```
Option Explicit

Private Sub UpdateChange()
Dim wrd
If optTitle Then
  Selection.Range.Case = _
     wdTitleWord
ElseIf optUpper Then
  Selection.Range.Case = _
     wdUpperCase
ElseIf optLower Then
  Selection.Range.Case = _
     wdLowerCase
ElseIf optSentence Then
  Selection.Range.Case = _
     wdTitleSentence
ElseIf optToggle Then
  Selection.Range.Case =  _
     wdToggleCase
End If
txtChange = Selection.Range
End Sub
Function SetCase(cs)
If cs = wdTitleWord Then
  optTitle = True
ElseIf cs = wdUpperCase Then
  optUpper = True
ElseIf cs = wdLowerCase Then
  optLower = True
ElseIf cs = wdTitleSentence Then
  optSentence = True
Else
  optTitle = False
  optUpper = False
  optLower = False
  optSentence = False
  optToggle = False
End If
End Function

Sub NextWord()
Do
  Selection.MoveRight _
     Unit:=wdWord, Count:=1
  Selection.MoveRight _
     Unit:=wdWord, Count:=1, _
     Extend:=wdExtend
Loop While Selection = "." Or _
  Selection = "," Or _
  Selection = vbCr
SetCase (Selection.Range.Case)
txtSel = Selection
butChange.Enabled = False
UpdateChange
End Sub

Private Sub butAdd_Click()
Selection = txtChange
AddWord txtChange
NextWord
End Sub

Private Sub butChange_Click()
Selection = txtChange
NextWord
End Sub
```

CONTINUED ON PAGE 100.

LISTING 1 ***CaseWizard Code.*** *This source is embedded in the Word document file. You can move this form to your Normal template to make it available to all documents, or you can add it to a special template you can open when the feature is required.*

Most of the properties and methods used in the Word object model have preassigned constants. As you type code, you can use the Ctrl-Shift-J keyboard shortcut to display the available constants for an object. VBA uses the special Word constant, wdTitleWord, to change the case of the selected Range. To prove your macro can change the case of the selection, press Alt-F8 to display the Macro's dialog and run your TestChangeCase macro. You should see the case of the selection change to Title Case.

An important part of the Word object model is the Range object, which represents a contiguous area of a Word document defined by the starting- and ending-character positions. You define a Range object any time you want to manipulate a portion of a document—insert text, copy text, convert text to a table, or change the format, such as the case. The code created by the recorder actually uses the Range method of the Selection object to create a Range object. It then changes the Case property of the newly created Range object.

You can create a Range object independent of the current selection. Try typing this code in the code module:

```
Sub TestBold()
dim MyRange as Range
set MyRange = ActiveDocument._
   Range(Start:=0, End:=10)
MyRange.Bold = True
set MyRange = ActiveDocument._
   Paragraphs(2).Range
MyRange.Bold = True
End Sub
```

The first line defines a Range object called MyRange. The second and third lines use the Range method of the ActiveDocument object to set your MyRange object to the first 10 characters of the active document, and to set the Bold property of that range to True. The next two lines use the Range method of the Paragraphs collection to set the second paragraph in the active document to bold.

CREATE THE CASEWIZARD

These case-formatting routines are useful for most documents. Unfortunately these standard functions don't work well with organization names. Many organizations use mixed-case text to convey their business identities, like "SunOpTech," or perhaps just a nonstandard case treatment, as in "INSO Corporation." These differences cause more trouble if you go international. In Germany, for example, many company names end in "GmbH" rather than "Inc."

With Visual Basic, you can write a tool to handle these special cases. In doing so you'll have a chance to learn some of the Word object model. Unlike Excel 97, which has changed little from Excel 95's object model, Word 97's object model is new. Fortunately, while the object model has hundreds of properties and methods, you will use only a small number of these objects in most of your VBA procedures. The built-in macro recorder, debugger, and object browser will help you learn about these objects. For more information, see my Getting Started with VBA column, "Program Word with VBA 5.0" [*VBPJ* December 1996].

The CaseWizard form should operate in a manner similar to the operation of the Spelling form in Word. You want to be able to start with the selected word and then click on an Ignore button to leave the case as is and move to the next word, or type the change to the case and click on a Change button. It would be nice to add to

CONTINUED FROM PAGE 98.

```
Private Sub butExtend_Click()
Selection.MoveRight Unit:=wdWord, _
  Count:=1, _
  Extend:=wdExtend
txtSel = Selection
UpdateChange
End Sub

Private Sub butIgnore_Click()
NextWord
End Sub

Private Sub optLower_Click()
UpdateChange
End Sub

Private Sub optSentence_Click()
UpdateChange
End Sub

Private Sub optTitle_Click()
UpdateChange
End Sub

Private Sub optToggle_Click()
UpdateChange
End Sub

Private Sub optUpper_Click()
UpdateChange
End Sub

Private Sub txtChange_Change()
If txtChange <> txtSel Then _
  butChange.Enabled = True
End Sub

Private Sub UserForm_Activate()
NextWord
End Sub
```

a file a special case setting for a word using an Add button in the same way you can add a special spelling for a word with the Spelling form. For the code to create The CaseWizard, go to the Registered Level of The Development Exchange (for more details, see the Code Online box at the end of this column).

Open a new Word document and move to the VBA development environment by pressing Alt-F11. Click on the UserForm menu item on the Insert menu to add a form to your Visual Basic project. Then add TextBox, Label, and OptionButton controls (see Figure 1). Add four CommandButton controls with the captions Extend, Change, Ignore, and Add. Select each control and change the Name property to "but" and the caption.

Change the Enabled property of the Change button to False so that your code can enable it when a change has been made. Double-click on the Change text box to expose the code window, and enter this code to enable the Change button if the text in the Change text box does not match the text in the Selection text box:

```
Private Sub txtChange_Change()
If txtChange <> txtSel Then _
   butChange.Enabled = True
End Sub
```

Now write a procedure called NextWord that moves the selection one word to the right (you can use the MoveRight method of the Selection object). I added a Do loop to skip "words" that are actually a period, comma, or carriage return. The procedure then calls the SetCase procedure to set the OptionButton controls on the form to the case of the new word, and to put this new word into the Selection text box. Then the procedure disables the Change button and updates the Change text box:

```
Sub NextWord()
Do
   Selection.MoveRight Unit:=wdWord, _
      Count:=1
   Selection.MoveRight Unit:=wdWord, _
      Count:=1, Extend:=wdExtend
Loop While Selection = "." Or _
   Selection = "," Or Selection = vbCr
SetCase (Selection.Range.Case)
txtSel = Selection
butChange.Enabled = False
UpdateChange
End Sub
```

Now double-click on the Change button and enter the code to change the Selection to the text in the Change text box. Call the NextWord procedure:

```
Private Sub butChange_Click()
Selection = txtChange
NextWord
End Sub
```

The Ignore button code is even easier—you simply call the NextWord procedure:

```
Private Sub butIgnore_Click()
NextWord
End Sub
```

You also call the NextWord procedure in the UserForm_Activate procedure to initialize the form with the first word. Even with these few lines of code, you can test your form. In a module, add a procedure called CaseWizardShow to show the CaseWizard form. From the Word document, select the Macro menu item from the Tools menu and run this macro. You should see your form appear, and the Ignore and Change buttons should function.

Review the entire CaseWizard procedure (see Listing 1) to enable the proper option button within the NextWord procedure. For an expanded version of the code that enables the Add button to save special case formats from a word to a file, go to the Premier Level of The Development Exchange (for details, see the Code Online box). ☒

Code Online

You can find all the code published in this issue of VBPJ *on The Development Exchange (DevX) at http://www.windx.com. All the listings and associated files essential to the articles are available for free to Registered members of DevX, in one ZIP file. This ZIP file is also posted in the Magazine Library of the* VBPJ *Forum on CompuServe. DevX Premier Club members ($20 for six months) can get each article's listings in a separate file, as well as additional code and utilities for selected articles, plus archives of all code ever published in* VBPJ *and* Microsoft Interactive Developer *magazines.*

Create a Wizard to Manipulate Case

Locator+ Codes

Listings ZIP file plus CaseWizard.doc (free Registered Level): VBPJ0297

✪ *Listings for this article plus a Word document that helps you prepare an ASCII file of saved case formats for words (subscriber Premier Level): GS0297P*