April, 2004

Advisor Answers

Accessing Word form data

VFP 8/7/6

Q: I have Word documents that contain textboxes, checkboxes, and so forth. I'd like to read the data entered by the users into FoxPro tables. Is there a way to do this?

A: Textboxes, checkboxes, and dropdowns in a document are managed using the document's FormFields collection. You can examine their contents and values and manipulate them by working with that collection.

Before I show you how to access the data in Word's controls, let me provide some background. Word allows you to create forms that let users enter data without modifying fixed items. These forms are analogous both to VFP's forms and to paper forms.

To create a form in Word, you use the Forms toolbar (View | Toolbars | Forms). It includes the three controls available (textbox, checkbox and dropdown). You add a control to a form by clicking that control on the toolbar—it's placed at the current insertion point. Once you add a control, you can set its properties by double-clicking on it or by clicking the Form Field Options button on the Forms toolbar. The properties vary with the control, and include things such as maximum text length for a textbox, the list of items to include in a dropdown, and whether a checkbox is initially checked.

Once you've finished designing a Word form, you change it from design mode to data entry mode by protecting the document (or the sections of the document that include controls). You do so by clicking the Protect Form button on the Forms toolbar or by choosing Tools | Protect Document from the menu. In the latter case, you then need to specify that you're protecting the document for Forms and, if the document has multiple sections, indicate which sections are to be protected. When you use the menu item, you can also attach a password to the document to keep other people from changing it back to design mode.

Once a document has been protected, anyone who opens it can't change the regular text it contains; all he or she can do is use the

controls. (However, unless a password is attached, any user can unprotect the document—that is, change it to design mode—and modify the regular text.)

To read the data from Word's controls via Automation, you need to unprotect the document first. The Document object's Unprotect method handles this task. The method accepts the document's password as an optional parameter:

```
oDocument.Unprotect("My Password")
```

Once you unprotect the document, you can access all the data in the controls using the FormFields collection. Like most collections, FormFields has a Count property to indicate how many items are in the collection and you can address the individual FormField items using the FormFields(nItemNumber) notation.

Retrieving the data is a little different for each type of control. The FormField object has a Type property that lets you determine the type of control. For textboxes, you can retrieve the user's entry using the Range.Text property:

```
oFormField.Range.Text
```

For checkboxes, the Checkbox.Value property returns .T. or .F.:

```
oFormField.Checkbox.Value
```

For dropdowns, Dropdown.Value returns a number indicating the position of the chosen item in the list:

```
oFormField.Dropdown.Value
```

You may not want to depend on the order of the fields in the document to indicate which is which. Fortunately, Word allows you to assign a bookmark to each control, and to retrieve the bookmark's name using the Name property of the FormField. You can use the Name property to ensure you store the data to the right field in your table.

This code (FormFields.PRG on this month's Professional Resource CD) opens a Word form (WordFormFilled.DOC on the PRD), and retrieves the data stored there. It puts each value into a variable with the same name as the corresponding control on the Word form. When it's done, it displays a messagebox showing the names and values of all the variables created.

```
#DEFINE wdFieldFormTextInput  70
```

```
#DEFINE wdFieldFormCheckBox  71
#DEFINE wdFieldFormDropDown  83

#DEFINE CRLF CHR(13) + CHR(10)

LOCAL oWord, oDoc, oFormField, nFieldCount, nField
LOCAL cFieldName, nFieldType, cOutput

oWord = CREATEOBJECT("Word.Application")
*** Don't forget to add the path to the filename
oDoc = oWord.Documents.Open("WordFormFilled.DOC")


WITH oDoc
  .Unprotect()
  nFieldCount = .FormFields.Count

  cOutput = ""
  FOR nField = 1 TO nFieldCount
    WITH .FormFields[nField]
      cFieldName = .Name
      nFieldType = .Type

      DO CASE
      CASE nFieldType = wdFieldFormTextInput
        STORE .Range.Text TO (cFieldName)
      CASE nFieldType = wdFieldFormCheckBox
        STORE .Checkbox.Value TO (cFieldName)
      CASE nFieldType = wdFieldFormDropDown
        STORE .Dropdown.Value TO (cFieldName)
      ENDCASE
    ENDWITH
    cOutput = cOutput + CRLF + cFieldName + " = " + ;
              TRANSFORM(EVALUATE(cFieldName))
  ENDFOR

  * Clean up
  .Close(0)
ENDWITH
oWord.Quit()

MESSAGEBOX(cOutput)

RETURN
```

–Tamar