Macros for Word Processors

Unit 4 (Part 2)

"Along with ‘Antimatter,’ and ‘Dark Matter,’ we’ve recently discovered the existence of ‘Doesn’t Matter,’ which appears to have no effect on the universe whatsoever."
VBA Features

- **Controlling your Programs**
  - VBA’s If statement is remarkably flexible, with several formats to choose from. All these forms involve three basic parts:
    - A condition test that is evaluated to yield a value of True or False
    - A then part, which supplies one or more statements that execute only if the result of the condition test is True
    - An else part, which supplies one or more statements that execute only if the result of the condition test is False

```vba
If Selection.Type = wdSelectionNormal Then
    ActiveDocument.Indexes.MarksEntry Range:=Selection.Range,
    Entry:="index"
Else
    MsgBox ("Please select something to mark.")
End If
```
VBA Features

- **Controlling your Programs**
  - VBA supports a special type of If structure, using the ElseIf keyword. The ElseIf form is a shorthand notation that allows you to simplify If structures that follow this form:

    ```vba
    If expression Then
        statements
    Else
        If expression Then
            statements
        Else
            If expression Then
                statements
            ElseIf expression Then
                statements
            ElseIf expression Then
                statements
            End If
        End If
    End If
    ```

**Example: Daily Advices**
VBA Features

- **Controlling your Programs**
  - For/Next loops allow you to set up a basic looping structure in which a series of statements execute repeatedly, with the value of a counter variable increased by one (or more) each time until the counter variable reaches a certain value.

    For counter-variable = start To end [Step increment]
    statements...
    Next [counter-variable]

  - While/Wend loops provide a more sophisticated form of looping, in which the loop continues as long as a specified condition remains True. The general form is

    While condition
    statements
    Wend
VBA Features

- **Controlling your Programs**

  The Select Case statement is designed for just this type of situation. It lets you test an expression for various values, executing different statements depending on the result. Its general form is

  ```vba
  Select Case expression
    Case case-condition
      statements
    [ Case case-condition
      statements ]
    [ Case Else
      statements ]
  End Select
  ```
Word’s Object Model

What?

- Word’s object model is the programming interface that lets you manipulate Word documents from VBA.
- The object model consists of a variety of different object types, such as documents, paragraphs, styles, tables, and so on.
- Each of these objects has its own properties and methods you can use from VBA. And many of these objects are contained in special collections.
- The most difficult aspect of writing Word macros is dealing with this object model, in part because Word’s object model is huge.
- Fortunately, you don’t have to know about all the objects, methods, and properties that make up the complete object model to start writing useful VBA macros.
Word’s Object Model

- **Major Objects in the WOM (1)**
  - **Application**: The Microsoft Word application itself. This object is considered to be the start of the Word object model. Use the Application object to access other top-level objects. For example, use the ActiveDocument property to access the Document object for the current document.
  - **Bookmark**: Used to access bookmarks. You can access the Bookmarks collection via a Document object.
  - **Cell**: Used to access cells in a table. You can access the Cells collection via a Row or Column object. You can also access an individual cell by using the Cell method of a Table object.
  - **Column (also Row)**: Represents a column in a table. You can access the Columns collection via a Table object.
  - **Diagram (Shapes)**: Represents a diagram. Diagrams are stored along with other shapes in a Shapes collection. You can access the Shapes collection via a Document object.
Word’s Object Model

- **Major Objects in the WOM (2)**
  - **Document**: Represents a single document. The Documents collection (available via the Application object) contains all open documents.
  - **Find**: Provides the function of the Find command on the Home tab.
  - **Font**: Represents a font. Accessed via the Font property of a Selection, Range, or Style object.
  - **HeaderFooter**: Represents a header or footer for a section of a document. The Header Footers collection is accessed via the Headers or Footers property of a Section object.
  - **List**: Represents a list in a document. The Lists collection is accessed via a Document object.
  - **Options**: Represents the settings in the Word Options dialog box.
Word’s Object Model

- **Major Objects in the WOM (3)**
  - **Page**: Represents the pages in a document. The Pages collection is accessed via a Pane object, not a Document object, as you may expect.
  - **PageSetup**: Represents the page setup, including margins and columns. Accessed via a Document or Section object.
  - **Paragraph**: Represents a paragraph. The Paragraphs collection is accessed via a Document, Selection, or Range object.
  - **ParagraphFormat**: Represents the format for a Paragraph or Style object.
  - **Replacement**: Provides the function of the Replace command on the Home tab.
  - **Section**: Represents a document section. The Sections collection is accessed via a Document object.
Word’s Object Model

- **Major Objects in the WOM (4)**
  - **Selection**: Represents the current selection and is accessed via a Selection or Range object.
  - **Shape**: Represents a shape. The Shapes collection is accessed via a Document object.
  - **Style**: Represents a style. The Styles collection is accessed via a Document object.
  - **Table**: Represents a table. The Tables collection is accessed via a Document, Range, or Selection object.
  - **Template**: Represents a template. All available templates (normal, attached, and global) are available in the Templates collection, which you can access via the Application object.
    - Use the AttachedTemplate property of a Document object to access the template attached to a document.
Word’s Object Model

- **Major Objects in the WOM (5)**
  - **TextColumn**: Represents a column. The TextColumns collection is accessed via a PageSetu object.
  - **Variable**: Represents a document variable. The Variables collection is accessed via a Document object.
  - **View**: Represents the view settings, such as whether paragraph marks or field codes are visible. Accessed via a Window or Pane object.
  - **Window**: Represents an open window. The Application. Windows collection has all open windows. The Windows collection for a Document object has just those windows that are open for the document.
Word’s Object Model

- **The Application Object**
  - The Application object is the starting point for Word’s object model. It represents the Word application itself. Here are a few of its more interesting properties:
    - Documents, Options, Selection, Templates, Windows…
    - UserAddress, UserInitials, UserName, Version.
  - The Application object also has several methods that can be useful; in particular:
    - **CleanString**: Removes nonprintable characters from a text string
    - **GoBack**: Moves the insertion point back to previous locations
    - **GoForward**: Moves forward to previous editing locations
    - **OnTime**: Starts a timer that runs a macro at a specified time
    - **Quit**: Quits Microsoft Word
    - **Repeat**: Repeats the most recent editing action
    - **Run**: Runs another macro
Word’s Object Model

- **Working with Documents**
  - You use the Document object to access a document in Word. The Document object has many useful properties and methods. Here are a few of the more interesting properties:
    - Name, Path, ReadOnly, Saved, Windws, AttachedTemplate
  - Here are some of the methods of the Document object:
    - **Activate**: Makes the document the active document
    - **Close**: Closes the document
    - **PrintOut**: Prints all or part of the document
    - **Save**: Saves the document and prompts the user for a name if the document has never been saved
    - **SaveAs**: Saves the file with a specified name and path
Word’s Object Model

- **Working with Documents**
  - **Accessing Documents**: You can access all documents that are open via the Documents collection.
    
    ```vba
    Documents(“Document1”).Save
    ```
  
  - **Creating Documents**: You can create a new document by using the Add method of the Documents collection. The Add method returns the document that was created.
    
    ```vba
    Dim d As Document
    Set d = Documents.Add
    ```
  
  - **Opening Documents**: You might have noticed that the Document object has no Open method. However, the Documents collection does
    
    ```vba
    Dim report As Document
    Set report = Documents.Open(“QuarterlyReport.doc”)
Word’s Object Model

- **Selection and Range Objects**

  - One of the most confusing aspects of Word’s object model is that two distinct objects refer to portions of a document: Selection and Range.

  - The Selection and Range objects are similar, with many overlapping features. However, selections can do some things that ranges can’t and vice versa.

  - A Selection object refers to a portion of a document that is selected. The selection can be made by the user before running the macro or the macro itself can select text.

  - Like a selection, a range is a portion of a document. However, a range isn’t selected, so it isn’t highlighted in the document window. Ranges allow you to work on document text without drawing attention to the text.

  - A document can have only one selection at a time

  - However, you can create as many ranges for a document as you want
Word’s Object Model

- **The Selection Object**
  - The Selection object refers to the portion of the document that’s selected.
  - If you’re writing a macro that manipulates the text selected by the user, you most likely need to use the Selection object:
    ```vba
    Selection.Font.Bold = True
    MsgBox Selection.Text
    ```
  - A user may make many different types of selections (Selection.Type):
    - `wdNoSelection`
    - `wdSelectionBlock`
    - `wbSelectionColumn / Row`
    - `wdSelectionNormal`
    - `wdSelectionShape`
Word’s Object Model

- **The Range Object**
  - A Range object identifies a portion of a document. The range can be as short as a single character or as long as the entire document.
  - Many objects with the Word object model have a Range property you can use to create a range that represents a portion of the document.
  - Suppose that you want to access the first paragraph of the active document as a range. To do so, you can use this bit of code:
    ```vba
    Dim r As Range
    Set r = ActiveDocument.Paragraphs(1).Range
    MsgBox r.Text
    ```
  - Object with Range: Bookmark, Cell, Comment, Footnote, Hyperlink, List, Row, Selection, Section, Table, etc.
Word’s Object Model

Moving Selections and Ranges

- The Selection and Range objects both have Start and End properties that indicate the character position of the start of the section or range and the position of the end of the section or range:

  Selection.End = Selection.End + 1

- The Selection and Range objects sport several methods that let you move the start or end of the selection or range:
  - **EndOf, StartOf**: Moves the start or end of the selection or range to the start or end of a specified unit
  - **Expand**: Expands the selection to include the next unit.
  - **Move**: Collapses the selection or range and then moves the selection or range the specified number of units.
  - **MoveStart, MoveEnd**: Moves the start or end of the selection or range the specified number of units.
  - **Next, Previous**: Moves the selection to the next or previous specified unit.
Word’s Object Model

- **Moving Selections and Ranges**
  - This table lists the possible unit values, which are used in several other methods as well.

<table>
<thead>
<tr>
<th>Constant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wdCharacter</td>
<td>Character</td>
</tr>
<tr>
<td>wdWord</td>
<td>Word</td>
</tr>
<tr>
<td>wdSentence</td>
<td>Sentence</td>
</tr>
<tr>
<td>wdParagraph</td>
<td>Paragraph</td>
</tr>
<tr>
<td>wdSection</td>
<td>Section</td>
</tr>
<tr>
<td>wdStory</td>
<td>Story</td>
</tr>
<tr>
<td>wdCell</td>
<td>Table cell</td>
</tr>
<tr>
<td>wdColumn</td>
<td>Table column</td>
</tr>
<tr>
<td>wdRow</td>
<td>Table row</td>
</tr>
<tr>
<td>wdTable</td>
<td>Table</td>
</tr>
</tbody>
</table>

I’m boring!! An example please!
Word’s Object Model

- **Working with Text**
  - **Accessing Text**: The Document, Selection, and Range objects have several properties you can use to access the contents of a selection or range:
    - **Text**: Returns a string that contains the text marked by the selection or range.
    - **Characters**: Returns a collection of Range objects, each representing one character of the selection or range.
    - **Words**: Returns a collection of Range objects, each representing one word of the selection or range. As with characters, no separate object exists for words. Instead, each word is represented by a Range object.
    - **Sentences**: A collection of Range objects, each representing one sentence of the selection or range.
    - **Paragraphs**: A collection of Paragraphs, each representing one paragraph of the selection or range.
    - **Sections**: Returns a collection of Section objects, representing the sections in the selection or range.
Word’s Object Model

Working with Text

Inserting Text: The Selection and Range objects offer the following methods for inserting text into your document:

- InsertAfter: Inserts the specified text after the selection or range. The selection or range expands to include the new text.
- InsertBefore: Inserts the specified text before the selection or range.
- InsertParagraph: Replaces the selection with an empty paragraph.
- InsertParagraphAfter: Inserts an empty paragraph after the selection or range.
- InsertParagraphBefore: Inserts an empty paragraph before the selection or range.
Word’s Object Model

- Working with Text
  - Deleting Text: Just using Delete method of a Range object
  - Copying, cutting, and pasting: Both the Selection object and the Range object support the standard copy, cut, and paste operations via the Clipboard:
    - **Copy**: Copies the selection or range to the Clipboard.
    - **Cut**: Cuts the selection or range to the Clipboard.
    - **Paste**: Pastes the contents of the Clipboard into the selection or range.
      - Use the **Collapse** method to collapse the selection or range if you don’t want the Clipboard contents to replace the contents of the selection or range.
Word’s Object Model

**Working with Text**

- **Formatting Text**: All options for formatting text are available through objects you can access as properties of the Selection or Range objects.
  - **Borders**: A collection of Border objects that define the borders for the selection or range.
  - **Font**: A Font object that lets you set character formatting. For more information, see the later section “Using the Font object.”
  - **ParagraphFormat**: A ParagraphFormat object that controls paragraph formatting, such as line spacing, alignment, and indentation.
  - **Style**: Sets the name of the style for the selection or paragraph
    ```
    Selection.Style = "Normal"
    ```
  - **TabStops**: A collection of TabStop objects. This collection is available only from a Paragraph object.
**Word’s Object Model**

- **Using the Font object**
  - The Font object gives you access to all the character formatting options available via the Font dialog box

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bold</td>
<td>Applies bold formatting</td>
</tr>
<tr>
<td>Color</td>
<td>Sets the font color</td>
</tr>
<tr>
<td>DoubleStrikethrough</td>
<td>Applies double-strikethrough formatting</td>
</tr>
<tr>
<td>Emboss</td>
<td>Embosses the text</td>
</tr>
<tr>
<td>Engrave</td>
<td>Engraves the text</td>
</tr>
<tr>
<td>Hidden</td>
<td>Hides the text</td>
</tr>
<tr>
<td>Italic</td>
<td>Applies italic formatting</td>
</tr>
<tr>
<td>Kerning</td>
<td>Sets the smallest point size at which Word applies kerning</td>
</tr>
<tr>
<td>Name</td>
<td>Sets the font name</td>
</tr>
<tr>
<td>Outline</td>
<td>Applies outline formatting</td>
</tr>
</tbody>
</table>

(continued)
Word’s Object Model

- **Using the Font object**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>Raises or lowers the text by the number of points specified</td>
</tr>
<tr>
<td>Shading</td>
<td>Sets shading for the text</td>
</tr>
<tr>
<td>Shadow</td>
<td>Applies a shadow effect</td>
</tr>
<tr>
<td>SmallCaps</td>
<td>Applies small cap formatting</td>
</tr>
<tr>
<td>Strikethrough</td>
<td>Applies strikethrough formatting</td>
</tr>
<tr>
<td>Subscript</td>
<td>Applies subscript formatting</td>
</tr>
<tr>
<td>Superscript</td>
<td>Applies superscript formatting</td>
</tr>
<tr>
<td>Underline</td>
<td>Underlines the text</td>
</tr>
<tr>
<td>UnderlineColor</td>
<td>Sets the color for the underline</td>
</tr>
</tbody>
</table>
Word’s Object Model

- Using the **ParagraphFormat** object
  - The ParagraphFormat object gives you access to all the paragraph formatting options available via the Paragraph dialog box

<table>
<thead>
<tr>
<th>Property</th>
<th>What It Specifies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment</td>
<td>The alignment (most common values: <code>wdAlignParagraphLeft</code>, <code>wdAlignParagraphCenter</code>, <code>wdAlignParagraphRight</code>, and <code>wdAlignParagraphJustify</code>)</td>
</tr>
<tr>
<td>Borders</td>
<td>Text borders</td>
</tr>
<tr>
<td>FirstLineIndent</td>
<td>Indentation of first line, in points</td>
</tr>
<tr>
<td>KeepTogether</td>
<td>Whether paragraph remains on one page</td>
</tr>
<tr>
<td>KeepWithNext</td>
<td>Whether paragraph is on same page as next paragraph</td>
</tr>
<tr>
<td>LeftIndent</td>
<td>Left indent in points</td>
</tr>
<tr>
<td>LineSpacing</td>
<td>Line spacing in points</td>
</tr>
</tbody>
</table>
### Word’s Object Model

- **Using the ParagraphFormat object**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LineSpacingRule</td>
<td>Type of line spacing (options: wdLineSpace1pt5, wdLineSpaceAtLeast, wdLineSpaceDouble, wdLineSpaceExactly, wdLineSpaceMultiple, and wdLineSpaceSingle)</td>
</tr>
<tr>
<td>PageBreakBefore</td>
<td>Whether paragraph begins new page</td>
</tr>
<tr>
<td>RightIndent</td>
<td>Right indent in points</td>
</tr>
<tr>
<td>Shading</td>
<td>Paragraph shading</td>
</tr>
<tr>
<td>SpaceAfter</td>
<td>Spacing after paragraph, in points</td>
</tr>
<tr>
<td>SpaceAfterAuto</td>
<td>Whether Word automatically sets space that appears after paragraph</td>
</tr>
<tr>
<td>SpaceBefore</td>
<td>Spacing before paragraph, in points</td>
</tr>
<tr>
<td>SpaceBeforeAuto</td>
<td>Whether Word automatically sets the space that appears before paragraph</td>
</tr>
<tr>
<td>Style</td>
<td>Name of style for paragraph</td>
</tr>
<tr>
<td>TabStops</td>
<td>TabStops collection for paragraph</td>
</tr>
</tbody>
</table>
Unit 4 Bibliography

- John Paul Mueller. VBA For Dummies. 2007