Understanding 'Me' (no flowers, no bees)

Posted on September 5, 2018September 6, 2018 by Rubberduck VBA

You may have read that **Me** was a *keyword*, or that it was some kind of "*special object* that's built into Excel"; or, you might have inferred that it's some kind of hidden instance/module-level variable that's only there in class/form/document modules: that's pretty much how I *was* understanding **Me**, until I saw what the language specifications (https://msdn.microsoft.com/en-us/library/ee156851.aspx) say about it (emphasis mine):

> Within the <procedure-body> of a procedure declaration that is defined within a <class-module-code-section> the declared type of the reserved name **Me** is the named class defined by the enclosing class module and the data value of **Me** is an object reference to ***the object that is the target object of the currently active invocation of the function***.

So **Me** is a *reserved name*… and it only exists in *procedure scope;* the type being the class it's used in makes it easy for *IntelliSense* to know what the members are, but its *value* is ultimately provided by the caller – from section 5.3.1.5 (https://msdn.microsoft.com/en-us/library/ee199631.aspx), "Parameter lists":

> Each procedure that is a method has an **implicit ByVal parameter** called the *current object* that corresponds to the target object of an invocation of the method. The current object acts as an anonymous local variable with procedure extent and whose declared type is the class name of the class module containing the method declaration. […]

In other words when you do this:

```
1  Dim foo As Class1
2  Set foo = New Class1
3  foo.DoSomething 42
```

What really happens under the hood is something like this:

```
1  Dim foo As Class1
2  Set foo = New Class1
3  Class1.DoSomething foo, 42
```

So every parameterless method you ever wrote like this:

```
1  Public Sub DoSomething()
2  End Sub
```

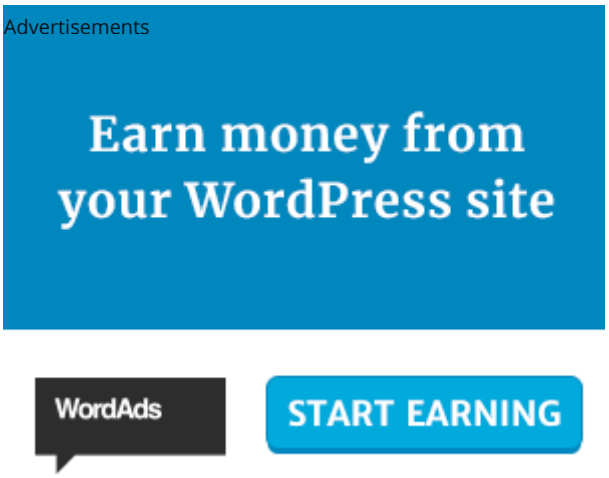
Is understood by VBA as this (assuming that method is in `Class1`):

```
1  Public Sub DoSomething(ByVal Me As Class1)
2  End Sub
```

…which, interestingly, is pretty much the same mechanics as the `this` pointer in C++.

So `Me` isn't a magic keyword, and doesn't have anything whatsoever to do with Excel (or whatever your VBA host application is) – `Me` is simply a *reserved name* that allows us to refer to this hidden *current object* pointer inside a procedure scope, and that *current object* is whichever instance of the *current class* the calling code is working with.

Advertisements



REPORT THIS AD

REPORT THIS AD

Posted in *tutorials*, *vba* *Tagged* *tutorial*, *vba*

Published by Rubberduck VBA

I'm Mathieu Guindon (Microsoft MVP Office Apps & Services, 2018), you may have known me as "Mat's Mug" on Stack Overflow and Code Review Stack Exchange. I manage the Rubberduck open-source project, whose goal is to bring the Visual Basic Editor (VBE) - VBA's IDE - into the 21st century, by providing features modern IDE's provide. *View all posts by Rubberduck VBA*

2 thoughts on “Understanding ‘Me’ (no flowers, no bees)”

1. **jonpeltier** *September 5, 2018* *Reply*

   You need to escape the less-than and greater-than characters in the quoted text blocks.

   Rubberduck VBA *September 6, 2018* *Reply*

   Thanks! Fixed!