

VBA Foundations, Part 10

A Tutorial in VBA for Beginners—The Tenth of a Twelve Part Series

Richard L. Binning / rbinning@attbi.com

All right, I see that you are back for more. Just like a kid in a candy store, can't get enough? Okay, reach in with both hands and let's see what we come up with. In this issue we will take a look at an old friend, the Object Model. "Wait!" You say. "We haven't worked with the Object Model yet!" I just smile and shake my head. "Actually, we have been working with object models all along!" Everytime we are typing in the editor and we see the "Intellisense" helper displayed (See Figure 1.), it has displayed because it was reading ahead and using the Object model to assist you in writing your code.

In previous issues I mentioned the "dot operator" or the period symbol, which is also used to access the object model properties or methods of the object in question. "What about me?" Get it? "Me" as in "me.width" or "me.textbox1.text". In the "Me" examples we used an alias, "Me", to represent the current object and accessed the object model through the "Me" alias with the assistance of the dot operator. It's all rushing back in now isn't it? Now let's start unwrapping that candy. In this issue we will discuss some new terms and concepts, watch the Object Browser in action, and look at special control structures for use with object models.

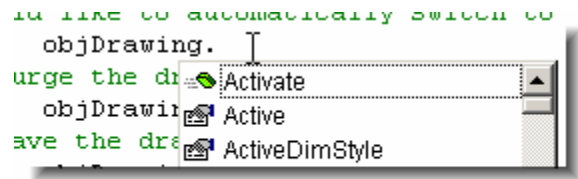



Figure 1.

The object model is what makes VBA customization possible. A good object model is key; because it is the means through which we will interact with AutoCAD. Therefore, the object model must be extendable, easy to use, and a solid representation of the application. Applications expose themselves to VBA through this mechanism known as the Object Model. This is actually part of COM, the acronym also known as the Component Object Model. Internet.com defines COM as the technology that "enables programmers to develop objects that can be accessed by any COM-compliant application." Let's explore this a little deeper by defining exactly what an "object" is. We have already used objects in our previous explorations. The form, aka UserForm1, was an object and the controls, such as the command button or the combo box that we placed on the form, were also objects. If we had to define an object then we might describe it in two parts: Firstly, An "object" is something which is able to save information or data. Secondly, an object offers more than a single behavior or activity that can either examine or affect or change the information or data it can save. Can you figure out what information the form had? What activity or behavior did it have? To find the answer to both of these questions, we can use the object browser to learn about the form. If you don't already have AutoCAD™ loaded please start AutoCAD. When it is fully loaded, type the following command at the command line: "vbanew". Now hold down the "alt" key and press and release the "F11" key to open the VBA editor. Next press the "F2" key or  select the "Object Browser" toolbar button from the menu as shown here. Take a look at the

Object Browser dialog box as shown in figure 2 to the right. Locate the top combo box and click it to see the object models available. Did you notice that MSForms is not listed? Take a look at the toolbar, did you see that the controls toolbox is grayed out and unavailable? There are a couple of ways out of this mess so pay attention as we load the Microsoft Forms 2.0 Object Model. The easiest way is to simply move the mouse to the "insert" pulldown, click it, and select "UserForm". Hey, not so fast, let's do it the hard way so we know how. Move your mouse to the "tools" menu and click the "references" item in the pull-down list. When the "References – ACADProject" dialog box is displayed, browse through the selections in the "Available References" list box displayed. Can't find it? It isn't there unless you load it or chose the easy way out in the last step and inserted a UserForm. Often you will wish to access an object model

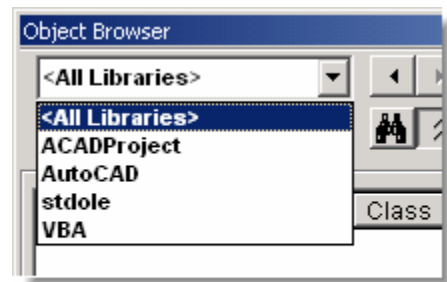


Figure 2.

VBA Foundations, Part 10

A Tutorial in VBA for Beginners—The Tenth of a Twelve Part Series

Richard L. Binning / rbinning@attbi.com

provided by others, or maybe you've been reading ahead and wish to load the ActiveX dll you created yourself. We can load any ActiveX dynamic link library (dll), or Microsoft Office Object Library (olb), or binary typelibrary (Tlb) by simply finding the correct library and selecting it from the references dialog box. Lets take the long road and find the Microsoft Forms 2.0 Object Model and get it loaded. The file is named "FM20.dll" and it will be located in your system folder location. If you are running Windows NT, 2000, or XP you should find the file in your C:\Winnt\System32 folder. If you aren't sure where it is located take the easy route and insert a UserForm. Once you have loaded it you should notice that it is now available in the "Available References" drop down list as shown in Figure 3. In fact, you can't unload it now that it has been loaded, go ahead and try if you don't believe me. While you are exploring this dialog box, take this opportunity to notice that the path and file name of the currently selected reference is displayed in the frame immediately below the "Available References" drop down list in the "References" dialog box. This feature can come in handy when a reference file goes

missing. At least you can find out the name of the file and perform a search for it. This can also assist you when installing routines on other computers and a resource file is missing, you will now know the filename and where it should be registered for the routine to work in the new location.

Lets explore our newly loaded object model by selecting it in the Object Browser's top combo box as shown in Figure 4. If we look back at our simplistic definition of what an object is, we are reminded that an object has information or data that can be saved, and this object also has actions or behaviors that can affect this data or information. As shown in Figure 4 select the "MSForms" object library from the combo box located at the top left of the object browser. Using the scroll bar located to the left of the "Classes" list on the left side of the Object browser dialog box, scroll down and select the "UserForm" class. Take a look at the icons used in the "members of UserForm" list as displayed in the pane on the right in Figure 4. The "Properties" icon looks like a piece of paper being gripped by a white gloved hand. The "Method" icon looks like a green flying brick. Both the "Properties" and "Method" icons have a unique twin brother

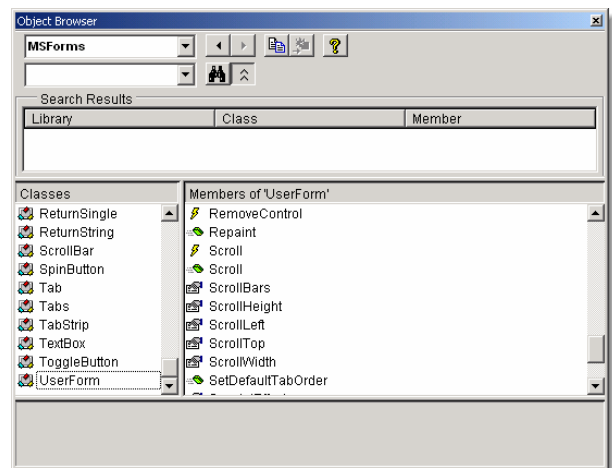


Figure 4.

VBA Foundations, Part 10

A Tutorial in VBA for Beginners—The Tenth of a Twelve Part Series

Richard L. Binning / rbinning@attbi.com

known as the “default”. These icons look identical to their brothers except they have a floating blue ball shown above them. We’ll talk about these defaults later and use the shortcuts they provide. The “Event” icon is a lightning bolt. What do these icons tell us? The “Properties” icons alert us to the data or information tied to this object. The “Methods” icons point us to the actions or activities that the object uses to save or store the properties mentioned earlier. So that leaves the “Events”, what does this icon alert us to? It provides built-in activities that occur to your object, for example the UserForm. These “Events” are at the heart of VBA and are precisely what makes this language so useful and interesting. Because of these events, we are now able to react to things the system or the user does. This is where interactivity takes place and is what elevates VBA above the “procedural” languages.

Lets think about objects as real world entities for a little bit. Next time you drive to work, think about what makes your car an object? What information or properties might your car “object” possess? Typical properties of your car “object” might include the color, the number of wheels, the number of doors, the speed it travels, etc. If you had to describe the typical activities or behaviors you might include driving, turning, starting, stopping, etc. Starting to sink in a little? Good! Now take this a step further, can you define any other vehicles in a similar manner? Cars and trucks are quite similar when abstracted and broken down into the parts, pieces, and activities that make an object an object. Although you might prefer to practice before driving a certain vehicle you are unfamiliar with such as a Truck if you always drive a car or a Van if you are used to driving a pick up truck. But, if push came to shove you could most likely jump behind the wheel of that unfamiliar vehicle and drive it quite successfully. How is it possible for you to do this? And How can I apply this to programming in VBA? You simply think of your familiar object or vehicle and using what you know you begin applying it to this “somewhat different” object. What happens when you have questions? That’s when you break out the Object Browser and take it for a spin, looking for those similar properties, methods, and events and explore those new properties, methods, and events you discover along the way. Does this work for everything in VBA you ask? Nearly everything. What other common objects in our everyday environment might we break down and abstract as an object? Wall switches come to mind. Do you need to know the intricate workings of how a wall switch works? Nope, you just need to toggle the switch to the opposite side to turn something either on or off. What if the switch was mounted sideways? This results in the same type of activity just in a different direction. What if there is more than one switch? What if the switch is really a push button? Are you starting to see the picture? This VBA thingee is not so incomprehensible after all.

So now we are starting to understand the cryptic term “object” a little better. This brings us back to COM, the component object model is a collection of “objects” and “groups of objects” that allow VBA to interact with applications, the operating system, and you the user. Now just like the car example, once you begin to work with an object model and start to understand it, you will find that you also understand other object models as a result. Although each car, truck, van, etc. is unique in some way, once you find out where the switches, dials, and gauges are you can operate any vehicle in much the same manner as your own car. Trust me when I say, that the COM concept is the same. Once you learn what switches to look for, you’ll be driving virtually any object model almost as soon as you sit in the drivers seat.

Microsoft describes using an object model as exploring the *content* and *functionality* of the object model. The *content* of the object model is the *objects* or *properties* it exposes to you. The *functionality* of the object model is how you and the operating system and your application *interact* with the *objects* or *properties*. Lets see how this is represented in AutoCAD’s object model. What might the content of the object model be? Thankfully, we don’t have to look very far or hard to fully understand this based on our in-depth knowledge of AutoCAD itself. Lets list a few of these and see where it takes us. Probably the most basic content would be the application itself. Think about what you can do in simple terms with AutoCAD. You can open and close it. What does the

VBA Foundations, Part 10

A Tutorial in VBA for Beginners—The Tenth of a Twelve Part Series

Richard L. Binning / rbinning@attbi.com

application work with? “Drawings you say?” “Good Answer!” What can you do with a drawing? You can open and close it, plot it, save it, save it as some other name, and purge it. These functions that we are so familiar with are also activities or methods of the object model. What might the content or information associated with our object model consist of? Perhaps the caption you see at the top of the AutoCAD window, or the drawings name itself, what about the version? Yes these and more are the properties or information states available to us when we begin exploring the object model.

Now that we have a better understanding of what an object is and how objects are tied into the VBA environment lets learn about how it is organized and how we can make use of it. Objects in the object model are provided in a hierarchical fashion. If we think of an object model in the shape of a triangle we might place the most basic (also the most powerful) object at the top and start working our way down. In our example we would place the application object, representing AutoCAD itself, at the top of the triangle. In order to work with anything inside of AutoCAD, it must exist first. Because VBA is built into AutoCAD, this step is taken for us. The “ThisDrawing” class, also known as the “ActiveDocument”, is provided for us automatically in every project. It is tied into AutoCAD itself.

Lets begin our exploration. Return to the VBA environment and double click on the “ThisDrawing” class located in the project browser window. This should activate the code window so that we can get busy. Move your mouse to the “Insert” menu and choose “procedure” from the pull down menu. Enter a name for your procedure as shown in Figure 5 and select the “OK” button after making sure your options are selected as shown in Figure 5. This action will insert a new procedure into the “ThisDrawing” class’ code window so that we can start exploring the application object.

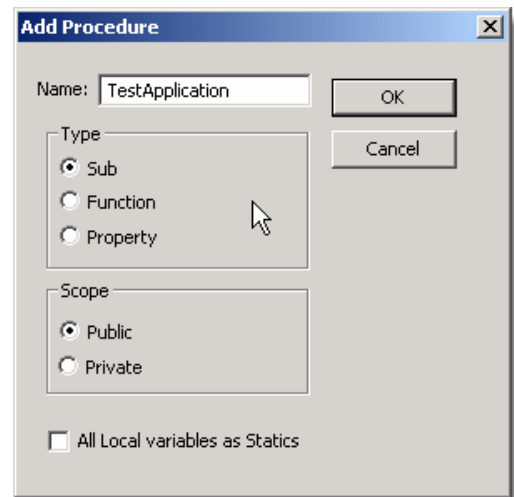


Figure 5.

So what does this application object contain? I am glad you asked! In the code window on the blank line inside our newly created procedure begin typing the word application. When you have typed the entire word place the dot operator at the end of the word. Did you notice that this item is apparently already declared for you since the “Intellisense” jumped right into action? (See Figure 6 for an example) See I told you that the “ThisDrawing” class had a lot of built in abilities. What you see in the “Intellisense” window is a real time listing of the object model pertaining to this particular class, “Application”. In the example you will notice that I mentioned a lot of these properties previously, such as Documents, Caption, FullName, etc. The Documents property is a collection that contains the drawings that are currently open inside of AutoCAD. The caption property is what you see at the top of the AutoCAD Application window. Have some fun, lets change the value of the AutoCAD caption to see the object model at work.

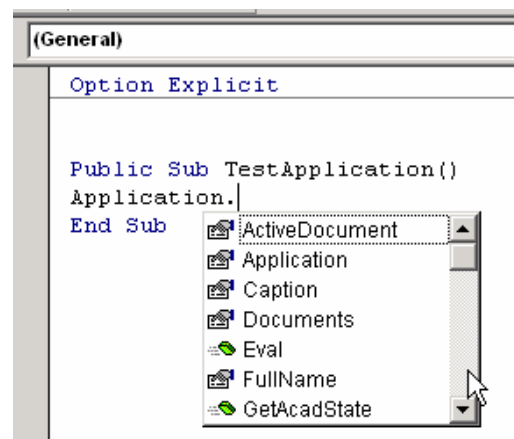


Figure 6.

VBA Foundations, Part 10

A Tutorial in VBA for Beginners—The Tenth of a Twelve Part Series

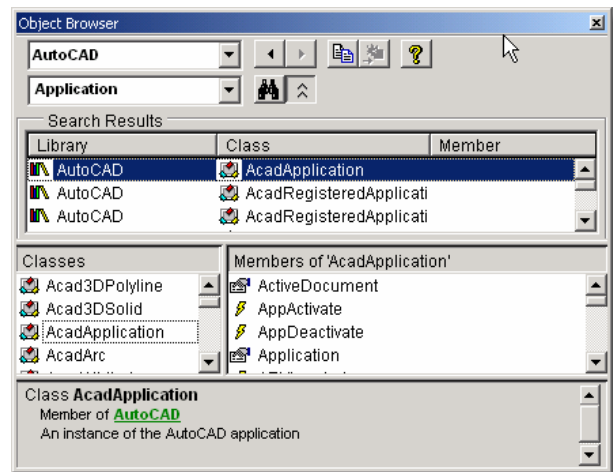
Richard L. Binning / rbinning@attbi.com

Since we have already started typing in the code window, why don't we finish out the sentence so that it matches what is shown below.

```
Public Sub TestApplication()  
    MsgBox Application.Caption  
End Sub
```

Alright, now let's run it. Do you remember how? If not then it's time to review some of the back issues. Alright I'll give you a hint. (Hint: F5) What happened? How might we make use of this? Pretty powerful stuff that VBA isn't it? If you're looking for this object in the object browser, make use of the search function by selecting

"AutoCAD" in the top-most combo box and typing in "Application" in the lower combo box. Now click the button that looks like a pair of binoculars. Your Object Browser window should look like the one shown in (Figure 7.). Another hint is probably in order here, most of the properties and objects we happily refer to while drawing in AutoCAD have a prefix in the object model of "acad". So if the application is "AcadApplication" what are the chances that other objects also have this same prefix? Chances are very good indeed. Take this opportunity to look at the definition of the "Class AcadApplication" as shown at the bottom of Figure 7. Now using the scroll bars located to the right of the "Members of 'AcadApplication'" column scroll down until we find our "Caption" property. What does the definition tell you about this property? Doesn't look like we can change the window caption does it? Actually we can, but you'll have to wait for a more advanced series of Articles. Go ahead and look for more properties, methods, and events.



(FIGURE 7.)

Okay back to our "Pyramid" like explanation of the AutoCAD object model. Two objects, "ActiveDocument" and "Application", taken care of what is next you ask? Well if we continue down the pyramid, you may notice that it is getting wider the farther down we go. If we look inside of the drawing itself, an "AcadDocument" object (notice the prefix?) which is part of the "Documents" collection, we realize that we will find other objects, such as layers ("AcadLayer" part of "AcadLayers" collection), blocks ("AcadBlock" part of "AcadBlocks" collection), dictionaries ("AcadDictionary" part of "AcadDictionaries" collection), etc. It looks like any item or function we can think of in AutoCAD is also either an object itself (usually with the prefix "Acad"), a group of objects (usually the same name as the single object with an "s" added to the end signifying that it is a built-in collection), or an activity (aka method) to be performed on an object, or finally an event that is triggered by either the user or the system itself. This is where we want to be in our understanding, we want to start seeing the relationships between what we know and what we wish to learn. With a mature object model at our disposal and our meager understanding of this new VBA language we should even now be able to find a method (activity), a property (content or information state), or an object or group of objects to accomplish almost anything we could accomplish manually inside of AutoCAD as long as we think it through well enough.

VBA Foundations, Part 10

A Tutorial in VBA for Beginners—The Tenth of a Twelve Part Series

Richard L. Binning / rbinning@attbi.com

There is one aspect of object models that we haven't really explored yet. This aspect is the concept of events. An event is anything that can be said to have happened. Objects can react to things that have happened. This is the heart of automation and what differentiates VBA from some other programming languages. Events are at the heart of VBA and the object model and AutoCAD has enabled many objects with the ability to react to events. In fact with a beginners understanding of events, one can create new events for objects to react against and with. The application object has events, so do the documents (drawings), and many more sub objects found in AutoCAD.

Lets look at a simple event. Switch over to the AutoCAD VBA editor and activate your "ThisDrawing" class code window. Using the "Object" combo box select "AcadDocument" from the list. Now lets look at the Methods and Events combo box immediately to the right. Select the "BeginRightClick" event from the list provided. This action should create an Event sub in your code window automatically. Can you figure out how this event will be triggered? Okay, lets test it. Type the following code as shown below:

```
Private Sub AcadDocument_BeginRightClick(ByVal PickPoint As Variant)
    TestApplication
End Sub
```

Now test this code by switching over to AutoCAD's window. How is it triggered? That's right! Every time you right click you mouse in the drawing area of the Application window, our previous sub is run which accesses the Application object and displays the words shown in AutoCAD's Main Window caption. That could get annoying, but it is a good demonstration of the kind of interaction that is possible with very little effort. If you really want to explore these concepts further, load up the following file and play with some of the examples contained within: Go to the directory where AutoCAD is installed, and find and load "example_events.DVB". It is located under the "Samples" folder in the "VBA" folder. Another good example file to explore is the "example_code.DVB" located in the same folder.

In summary, the objects, groups of objects, properties, and activities available in an application are divided among the objects in the object model. Basically, the objects in the object model's hierarchy represent all the properties or functionality provided by the application that is exposed to Visual Basic for Applications. Individually, these objects provide access to very specific properties and/or functionality. To discover or change a property or state of an object, you get or set one of the object's characteristics. To perform an action on or with an object, you use one of the object's methods. Some objects also provide events that are triggered by a reaction to some discrete happening or action, so we can write code that will run in response.

As always, use the on-line help to further explore these concepts as necessary. Download the longer version of this article to get more in-depth treatment and "hands-on" examples of these concepts. If you are really stumped, please send in your questions to the VBA guild or the email address at the top of this article. See you on the guilds or in the next issue of AUGIWorld™ magazine coming soon to a mailbox near you.