Introduction to VBA Programming with ArcObjects

GeoTREE Center
University of Northern Iowa
Geography
July 18, 2007

Workshop Outline

- ArcObjects/VBA overview (9:15-9:45)
- Customizing ArcMap interface (9:45 – 10:30)
- Visual Basic for Applications (VBA) environment (10:30-11:00)
- Morning break (11:00-11:15)
- VBA programming concepts (11:15-12:15)
- Lunch (12:15-12:45)
- ArcObjects overview (12:45-1:30)
- Using ArcObjects
  - Using ArcObjects 1: Map Display (1:45 – 2:45)
  - Afternoon Break (2:45 – 3:00)
  - Using ArcObjects II: Selecting, Geoprocessing (3:00 – 4:00)

ArcObjects/VBA Overview

Warning

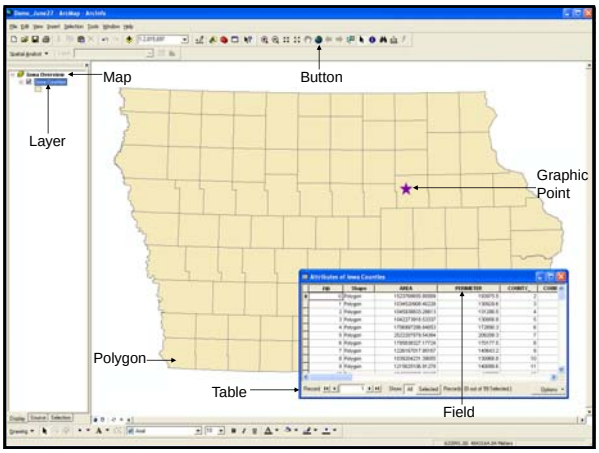
- Developing ArcGIS functionality and understanding ArcObjects is complicated
  - This workshop is a basic introduction to help you develop ArcGIS customizations

ArcObjects/VBA Overview

- ArcGIS provides a large amount of functionality
- However users often want to harness that functionality in different ways than is possible out of the box
  - Develop customizations to carry out work-flow tasks
  - Develop customized spatial modeling operations
  - Combine multiple steps into a single customized tool

ArcObjects

- Set of components or building blocks on which the scaleable ArcGIS framework is built
- Developed by ESRI using C++ as classes
- Basically everything you see and interact with in any ArcGIS application is an ArcObject
  - Maps
  - Layers
  - Points
  - Tables
  - Fields
  - Rasters
  - Buttons



ArcObjects

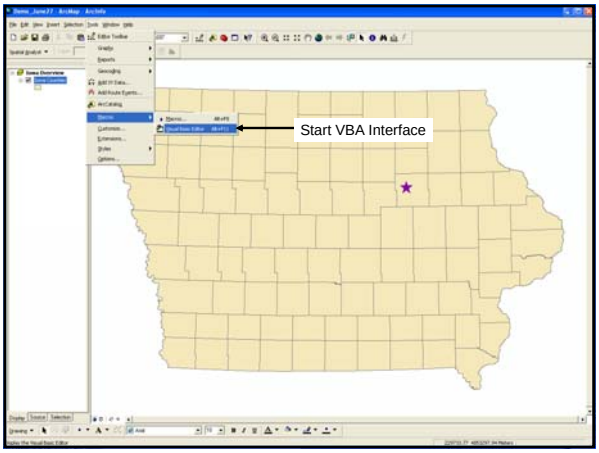
- There are a huge number of ArcObjects
- Accessible through various programming/development enviroments
  - Focus today on VBA
- Almost impossible to get to know all ArcObjects
- A strong background using the applications (ArcMap, ArcCatalog, etc.) important
- Learn how to navigate to get to proper ArcObject

Visual Basic for Applications (VBA)

- VBA is a development environment that is provided with ArcGIS (also with Microsoft Word, Excel, Powerpoint, etc.) with which you can access ArcObjects
- It is a simplified version of Visual Basic
- For customizing applications

Other Development Environments

- Visual Basic
- C#
- C++
- Delphi
- others



Scripting vs. development environment (ArcObjects)

- Scripting for geoprocessing in ArcGIS
  - Python, VBScript, etc.
- Scripting calls on ArcObjects to do processing
  - Scripting calls upon one main ArcObject
    - Geoprocessing ArcObject
- Development environments (VBA, VB, C# etc.)
  - Allow access to all ArcObjects
  - Developing customized interfaces
  - Distributable customizations

Scripting vs. Arcobjects (VBA,etc.)

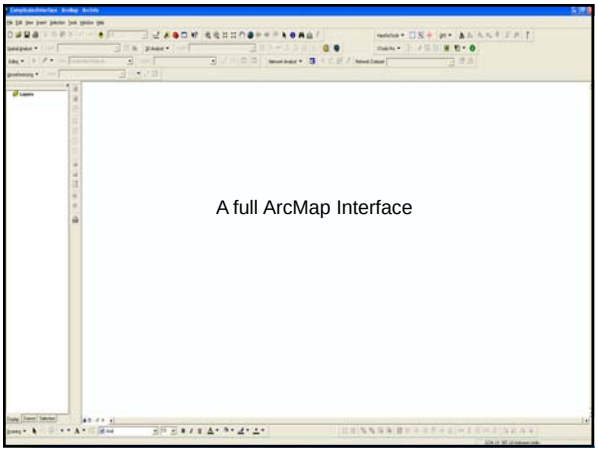
- Scripting with Python
  - Clip feature class with another feature class

- ArcObjects with VBA
  - Clip feature class with another feature class
  - Add clipped layer to map
  - Symbolize the new layer
  - Create a map layout with the new layer
  - Print the map

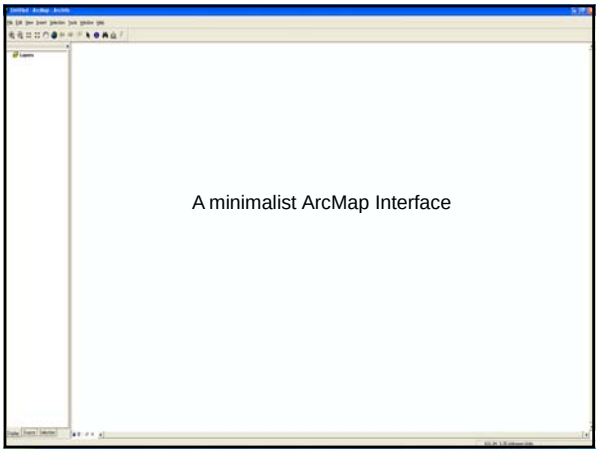
Customizing ArcMap Interface

Customizing ArcMap Interface

- You can control the look and feel of the ArcMap interface
  - Add/remove existing controls
  - Create new controls
  - Can associate VBA code to newly created tools, buttons and menu items



A full ArcMap Interface



A minimalist ArcMap Interface

Control Types in ArcMap

- Toolbars
  - Buttons, tools

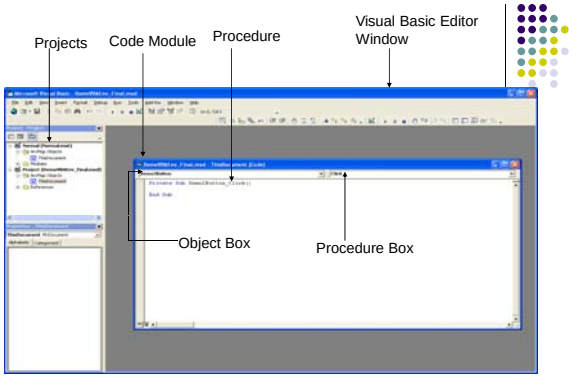
  - Buttons make something happen immediately
  - Tools work by clicking on the control and then clicking in the map display
- Menus
  - Menu items (really are just buttons)
- Combo boxes (e.g. Map scale)
  - Provide user dropdown choice
- Editbox (rarely used)
  - To enter and report information

Customization Demonstration and Exercise

VBA Development Environment

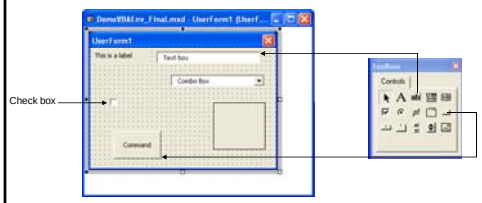
VBA Development Environment

- Accessed through ArcMap or ArcCatalog
- Tools for developing code and user interfaces
  - i.e. modules for code and forms for user interfaces
- A sophisticated program in itself that takes time to learn
- Lots of functionality and tools to help you more efficiently write code



Forms

• User can design custom interfaces using the form designer



VBA Environment

- You can store customized controls and codes in either an .mxd project, in the Normal.mxt or in one of your own template.
- If you save these customizations in the Normal.mxt they will be available every time you open ArcMap (*only on your computer*).
- Today we are only going to work saving customizations into a specific .mxd.

VBA Development Environment Demonstration and Exercise

VBA Programming Concepts

Comments

- For your sake and others it is important to put comments in your code
- Comment enough so when you return to the code later it will make it much easier for you to understand
- Comments begin with ‘ and are shown in red

```
‘Get the number of layers that are in the map
intLayerCnt = pMap.LayerCount
```

Intellisense

- VBA has functionality to finish code for you
- While typing a variable, method, procedure, etc., clicking Ctrl+Spacebar will finish code
- An example for a variable named strStateName
  - Type strSt and click Cntrl+Spacebar and VBA will finish strStateName for you
- Very useful to guard against typos

Variables

- Variables are used to store values
- It is encouraged practice to ‘declare’ variables
  - Dim intMyNumber as Integer
- This tells the program what kind of data type the variable is and provides clarity
- One programming convention has the variable name prefaced by an abbreviation for what the data type is

Data types

- Numbers
  - Integer – whole numbers from -32768-32767
  - Long – large whole numbers
  - Double – all numbers (very large or with decimals)
- Strings – text
- Boolean – true or false
- Dates – hold dates
- Variant – a generic data type that can hold any data type

Basic data types and abbreviations

- Integer – int
  - intTemp = 32
- Long – lng
  - lngLength = 45000
- Double – dbl
  - dblArea = 1254.56
- String – str
  - strStreet = “Clay Street”
- Boolean – bln
  - blnCancel = True
- Date – dat
- Variant - var

Setting variables

- You set the variables in an assignment statement

Ex. 1

```
Dim lngX as Long
lngX = 120000
```

Ex. 2

```
Dim dblAnnualTax as Double
Dim dblParcelValue as Double
dblParcelValue = 100000
dblAnnualTax = 0.05 * dblParcelValue
```

Conditional logic

- It is common to have to account for different conditions in programming
- Use conditional logic
- Most common is If Then

```
If intTempF <= 32 then
    msgBox “It might snow”
Else
    msgBox “It might rain”
End if
```

Looping

- A program often needs to loop through a collection of objects
- First way to do it is with a For….Next

```
For intNum = 1 to 10
    msgBox “The number is “ & intNum
Next I
    ArcMap Example
For i = 0 to pMap.LayerCount - 1
    msgBox “The layer name is “ & pMap.Layer(i).Name
Next i
```

Looping

- Second way to do it is with a Do….Until or Do…While

```
Do While intCnt < 50
    msgBox intCnt
    intCnt = intCnt + 1
Loop
    ArcMap Example
Do Until pRow Is Nothing
    dblArea = pRow.Value(2)
    Set pRow = pCursor.NextRow
Loop
```

Procedures

- Procedures hold blocks of code that carry out specific functions
- We have seen event procedures
  - E.g. MyButton_Click
- Two types
  - Sub procedures
  - Functions

Sub procedures

- Can be a control event procedure or a stand-alone procedure that is called from somewhere else
- Should be named logically
  - E.g. ReturnMapRasters
- When you create a control in ArcMap or Form then each sub procedure linked with an event is automatically named
  - 'ZoomOut_Click' (ArcMap button)
  - 'cmdGetMapName_Click' (form command button)

Call sub example

```
Public Sub ShowName()
    Dim strName as string
    Call GetName(strName)
    msgBox "The name is " & strName
End Sub

Public Sub GetName(strName as string)
    strName = InputBox("Enter Name", "Name")
End Sub
```

Functions

- Functions take input, process the input, and return output
- Many built-in functions in VBA
  - Int(2.6) returns 2
  - Len("Long") – returns 4 (i.e. length of string)
- Functions have a single output that can be string or numeric
- You can define your own functions

Function example

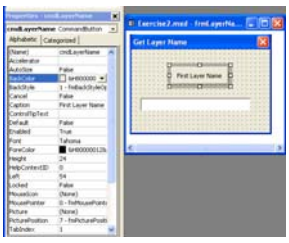
```
Public Sub ReportTax()
    ……..
    dblPropVal = 80000
    dblTotalTax = CalculateTax(dblPropVal)
    msgBox "The tax due is " & dblTotalTax
End Sub

Public Function CalculateTax(dblPropVal) as Double
    CalculateTax = 0.075 * dblPropVal
End Function
```

Objects

- Mentioned ArcObjects before
- There are other objects
  - Forms and all controls are objects
  - Other objects such as Collections and Arrays
- Each of these objects has properties, events and methods
  - Property is a characteristic or attribute
  - Events are user actions that happen to an object
  - Methods are things an object can do

Properties

- Properites are characteristics of an object
- Can set properties in form designer window



Properties

- Can also set form and command properties through code
- Use the 'object.property' syntax
  - cmdLayerName.Caption = "First Layer Name"
  - cmdLayerName.Enabled = False
  - txtLayerName.Text = ""
  - frmLayerName.Width = 200

Events

- Forms and controls have a number of potential events they react to
  - cmdLayerName.Click
  - txtLayerName.Change
  - frmLayerName.Initialize

Methods

- Things that an object can do
  - frmLayerName.Hide
  - cmdLayerName.Move 25, 50
  - cboLayerName.AddItem "Iowa Counties"
- Other VBA objects have methods
  - E.g. Collection objects are lists to which can hold different variables
    - rasterColl.Add pRaster
    - intRasterCnt = rasterColl.Count

Other Tips

- To get help put your cursor on a method or property and click F1

Overview of ArcObjects

Object Oriented Programming

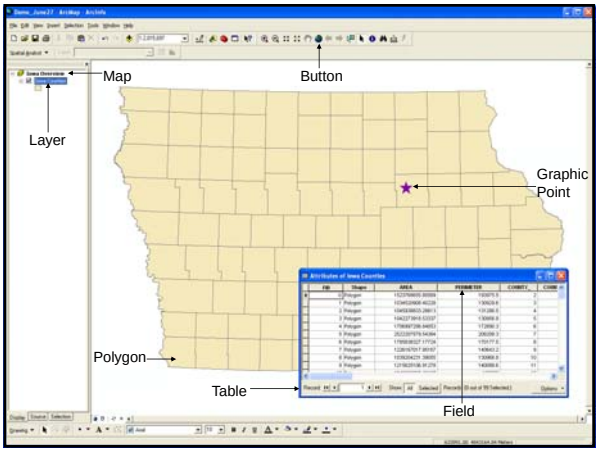
- OOP is centered around objects
- OOP programs have objects that hold data, have properties, respond to methods, and raise events
- E.g. a professor's program to calculate grades
  - A student object might hold name, midterm grade, etc.
  - A SemGrade method might calculate semester grade

Object Oriented Programming

- Two tiers of OOP
  - Low-level is creating and using objects (properties and methods) from *existing* classes (client)
  - Upper-tier of creating the classes themselves (server) and writing code for properties and methods
- We will mainly look at the client side today
  - i.e. We are going to make use of existing objects (VBA and ArcObjects)

ArcObjects

- Set of components or building blocks on which the scaleable ArcGIS framework is built
- ArcObjects come from classes designed by ESRI programmers
- Basically everything you see and interact within any ArcGIS application is an ArcObject
  - Maps
  - Layers
  - Points
  - Tables
  - Fields
  - Rasters

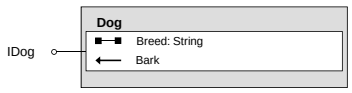


Programming Interfaces

- In order to work with ArcObjects you need to learn how to access objects with interfaces
- An interface is a logical grouping of properties and methods for a class
- Interfaces start with the letter I and variables are prefaced with p

  Dim pMap as IMap
- Can have multiple interfaces on a single class
- All ArcObjects classes have interfaces

Hypothetical Dog Class Example



- A single interface (IDog) on a dog class which has a single property (Breed) and method (Bark)

Dog class example (cont.)

- To declare and use and IDog object from the dog class

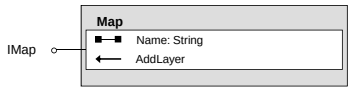
  Dim pDog as IDog

  Set pDog = New Dog

  pDog.Breed = “Poodle”

  msgBox “The dog is a “ & pDog.Breed

ArcObjects example

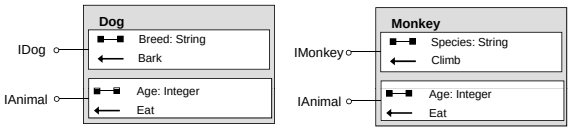


- The Map class has an interface named IMap and through that interface you can get/set name of the map and you can add a layer.

Multiple Interfaces

- As mentioned before, can be multiple interfaces on the same class
- In order to access properties and methods from multiple interfaces you might set up two variables that are equal to the same object
  - This is called a QueryInterface or QI
- Following is a hypothetical example
  - Will see ArcMap related examples as we go on

Hypothetical Dog Class Example (Two interfaces)



- Two interfaces with different properties and methods on the Dog object
- The IAnimal is an interface on different classes

Dog class example (cont.)

- To declare and use and IDog object from the dog class

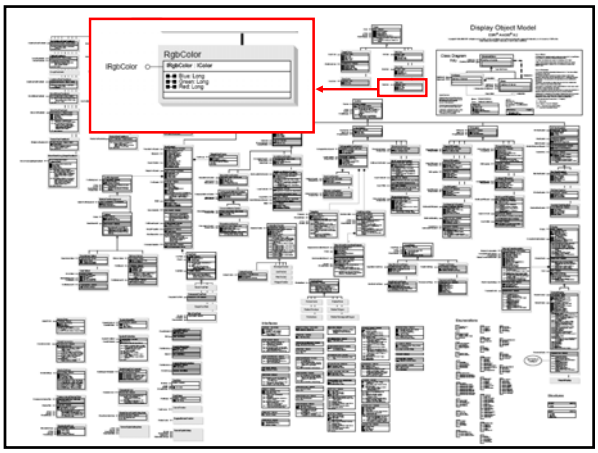
```
Dim pDog as IDog
Dim pAnimal as IAnimal
Set pDog = New Dog
Set pAnimal = pDog      'QueryInterface
pDog.Breed = "Poodle"
pAnimal.Age = 10
msgBox "The dog is a " pDog.Breed & " and she is "
   pAnimal.Age & " years old."
```

Object Model Diagrams

- There are many (thousands) of ArcObjects classes
- In order to get to a given object you might have to navigate through many others
  - MxDocument - Map – Layer
- There are a set of diagrams (pdf files) which provide a graphical representation of these objects, interfaces, methods, properties, and relationships
  - Object Model Diagrams

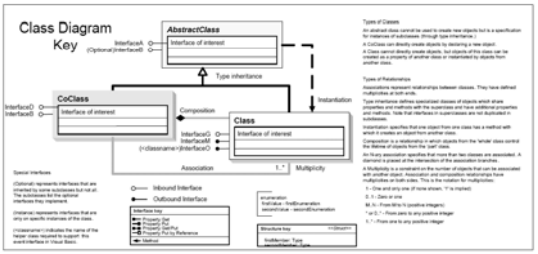
OMD's

- Designed with Unified Modeling Language
- Very detailed
- Have to learn how to read like a roadmap
- Can be complicated and daunting
- Learn how to read and only get to what you need
- OMD's organized by categories (i.e. Geometry, Geodatabase, ArcCatalog, Spatial Analyst)



OMD Key

- There is a key on every OMD explaining classes and relationships



Symbols

- Get/Put (read/write)
- Get (read)
- Put (write)
- Put by reference (use Set ..)
- Method
- Interface

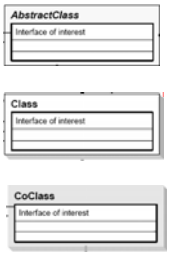
OMD's online

- Go to
  - http://edndoc.esri.com/arcobjects/9.2/welcome.htm
  - Panel on right of window click on ArcObjects Library Reference at bottom
  - Choose the category you think that you think your ArcObject might be found and click on it
  - Click on the ….Object Model Diagram and it will open up the OMD

Class Types

- There are different kinds of classes
  - Abstract classes – no objects created from these
  - Classes (regular) – made or gotten from other classes
  - Coclasses – can create objects from coclasses
    - E.g. our Dog class earlier could create a new Dog object.
    - Can also get objects of coclasses from other objects that return them
      - E.g. a new Map object is returned from another class with the .FocusMap method

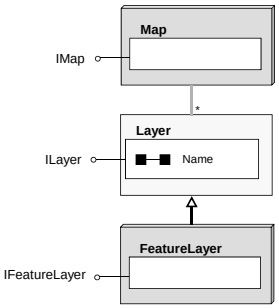
Class Types Symbology



Class Relationships

- Associations
- Instantiation – one class has a method that creates new object from another class
- Inheritance – a class uses as an interface from a more general class
- Composition – objects in one class ('whole class') control lifetime of another class ('part class')

Association and inheritance



Layer(s) associated with Map

FeatureLayer(s) use interfaces of Layer (inheritance)

Code Examples

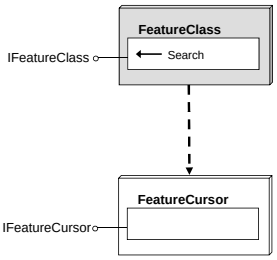
- Association

```
Dim pMap as IMap
Dim pLayer as ILayer
…… 'pMap set here
Set pLayer = pMap.Layer(0)
```

- Inheritance

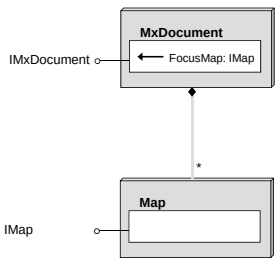
```
Dim pFeatureLayer as ILayer
Set pFeatureLayer = New FeatureLayer
pLayer.Name = "Iowa Counties"
```

Instantiation (create)



A FeatureCursor object is created from a FeatureClass object using the Search method

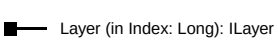
Composition (create)



An MxDocument can have multiple maps. If you delete the MxDocument then the Maps are deleted

Miscellaneous OMD stuff

A balloon like this indicates you need to go to another OMD



This syntax indicates the Layer property requires a variable of type Long and returns an ILayer object

Code Example

- Instantiation

```
…… 'pFeatureClass would be set in here
Dim pFeatureCursor as IFeatureCursor
Set pFeatureCursor = pFeatureClass.Search(Nothing, True)
```

Two special objects

- To use VBA for ArcMap a map document must be already be open
- Two objects are already in use at this point
  - Application object
    - Called Application
  - MxDocument object
    - Called ThisDocument

MxDocument

- With an .mxd open saved as ThisDoc.mxd

Application.Caption

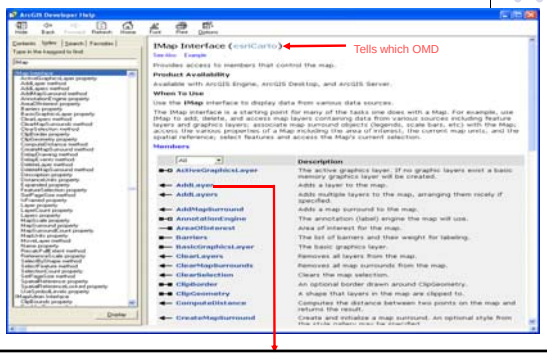
MsgBox ThisDocument.Title

ArcObjects and VBA Help

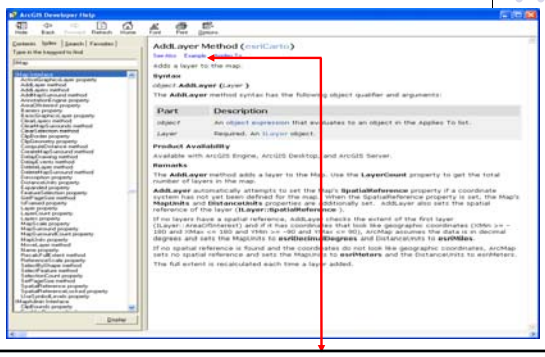
- In the VBA editor to get Visual Basic Help go to Help – Microsoft Visual Basic Help

- To get help for ArcObjects click F1 on and interface in the code module windows
  - E.g. put your mouse on 'IMap' and click F1

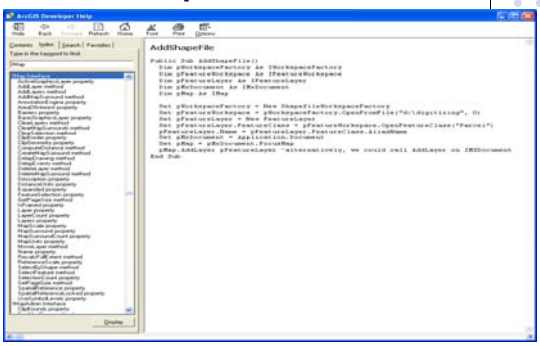
Example ArcObjects Help



Method help



Code example

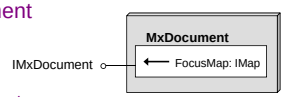


Using ArcObjects: Map Display, Layers, Feature Classes, and Tables

Practical Examples – Get Map

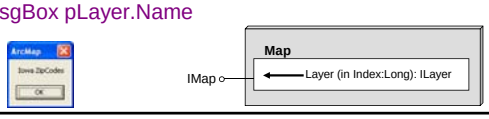
- Get the MxDocument and the active map or data frame
- The following code is probably going to be used in most programs you write

```
Dim pMxDoc as IMxDocument
Dim pMap as IMap
'get the map
Set pMxDoc = ThisDocument
Set pMap = pMxDoc.FocusMap
```



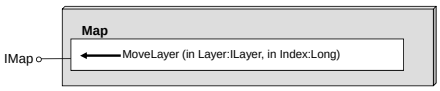
Get layer from Map

```
Dim pMxDoc as IMxDocument
Dim pMap as IMap
Dim pLayer as ILayer
'get the map
Set pMxDoc = ThisDocument
Set pMap = pMxDoc.FocusMap
'get the first layer in the map
Set pLayer = pMap.Layer(0)
msgBox pLayer.Name
```



Find and move a layer

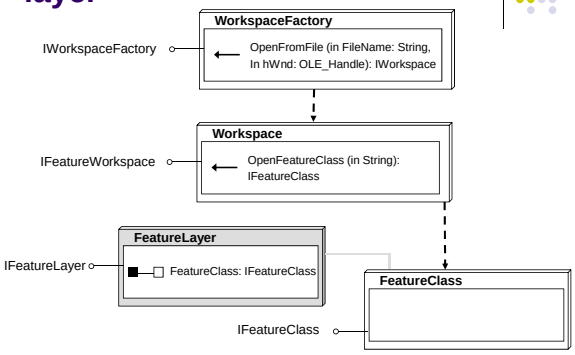
```
……
'get layer count, use that to get bottom layer,
   'move that layer to the top
intLayerCnt = pMap.LayerCount
Set pMoveLayer = pMap.Layer(intLayerCnt – 1)
pMap.MoveLayer pMoveLayer, 0
```



Create and add a new feature layer

```
……
Dim pSFWSFact as IWorkspaceFactory
Dim pFeatWS as IFeatureWorkspace
Dim pFeatureClass as IFeatureClass
Dim pFeatureLayer as IFeatureLayer
'set the workspace
Set pSFWSFact as new ShapefileWorkspaceFactory
Set pFeatWS = pSFWSFact.OpenFromFile("D:\VBAWshop", 0)
'open feature class
Set pFeatureClass = pFeatWS.Open("IowaRivers.shp")
'create layer, set its feature class, set name, and add to map
Set pFeatureLayer = new FeatureLayer
Set pFeatureLayer.FeatureClass = pFeatureClass
pFeatureLayer.Name = "Iowa Rivers"
pMap.AddLayer pFeatureLayer
```

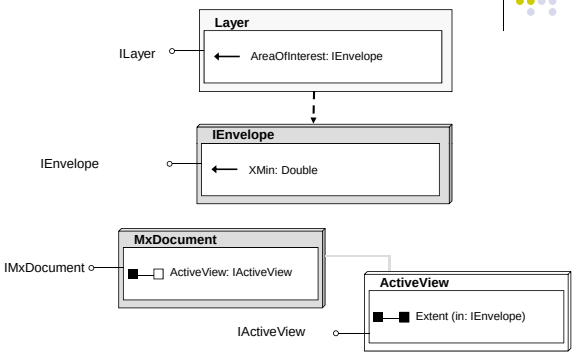
Create and add a new feature layer



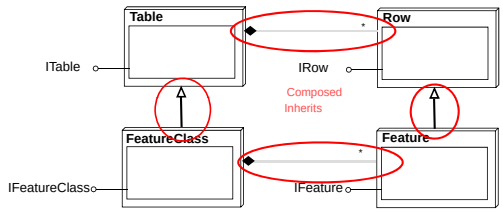
Zoom to Extent of Layer

```
……..
Dim pExtent as IEnvelope
'get the layer and it's extent
Set pZoomLayer = pMap.Layer(0)
Set pExtent = pZoomLayer.AreaOfInterest
'zoom to the extent and refresh the view
pMxDoc.ActiveView.Extent = pExtent
pMxDoc.ActiveView.Refresh
```

Zoom to layer extent



Tables and feature classes



Tables

- Open from an IWorkspace object
  - Use an AccessWorkspaceFactory to open a personal geodatabase table
  - Use a ShapefileWorkspaceFactory to open a .dbf table
  - ExcelWorkspaceFactory to open an .xls table
  - …others???

Open Table (.dbf)

```
'declarations
Dim pSFWSFact As IWorkspaceFactory
Dim pTableWS As IFeatureWorkspace
Dim pOpenTable As ITable
Dim intRowCnt As Integer

'set the workspace
Set pSFWSFact = New ShapefileWorkspaceFactory
Set pTableWS = pSFWSFact.OpenFromFile("D:\temp\julytrash\VBAwshop", 0)

'open the table
Set pOpenTable = pTableWS.OpenTable("IowaCounty_Population.dbf")

'get the number of rows and report
intRowCnt = pOpenTable.RowCount(Nothing)
MsgBox intRowCnt
```

Open Table (Personal Geodatabase table)

```
'declarations
Dim pAccessFact As IWorkspaceFactory
Dim pTableWS As IFeatureWorkspace
Dim pOpenTable As ITable
Dim intRowCnt As Integer

'set the workspace
Set pAccessFact = New AccessWorkspaceFactory
Set pTableWS = pAccessFact.OpenFromFile("D:\VBAWshop\Iowa.mdb", 0)

'open the table
Set pOpenTable = pTableWS.OpenTable("IowaCountyPopulation")

'get the
intRowCnt = pOpenTable.RowCount(Nothing)
MsgBox intRowCnt
```

Using ArcObjects II: Cursors, Selection Sets, Geoprocessing

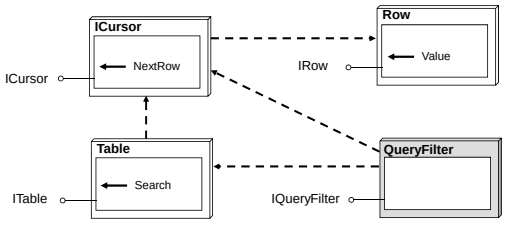
Cursors

- Cursors are used to retrieve a set of records
- You can step through a cursor row by row in a forward direction
- These are not the selected records you might see through an attribute query
- Very useful for getting and setting values in records row by row

Table cursor example

```
…….
Dim pCursor as ICursor
Dim pQF as IQueryFilter
Dim pRow as IRow
Dim intFldPos as integer
'get the field position of the County name field
intFldPos = pOpenTable.FindField("COUNTY")
 'set up the cursor using a query filter
Set pQF = New QueryFilter
pQF.WhereClause = "[TOT_POP] > 100000"
Set pCursor = pOpenTable.Search(pQF, True)
Set pRow = pCursor.NextRow
'loop through and list counties with Pop > 100000
Do Until pRow Is Nothing
    MsgBox pRow.Value(intFldPos)
    Set pRow = pCursor.NextRow
Loop
```

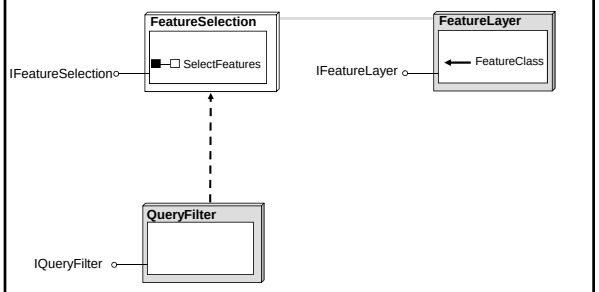
Tables and feature classes



Selection Set

```
'select the counties that have a population > 100000
  'Declarations
  Dim pCountyFLayer As IFeatureLayer
  Dim pCountyQF As IQueryFilter
  Dim pCountyFSel As IFeatureSelection
……..
  'get the layer and its feature class
  Set pCountyFLayer = pMap.Layer(0)

  'set up the query filter
  Set pCountyQF = New QueryFilter
  pCountyQF.WhereClause = "Tot_Pop > 100000"

  'set feature selection to the layer and refresh the map
  Set pCountyFSel = pCountyFLayer  'QI
  pCountyFSel.SelectFeatures pCountyQF, esriSelectionResultNew, False
  pMxDoc.ActiveView.Refresh
```

Tables and feature classes



Spatial Processing

- There is no central location for accessing spatial processing objects
- IBasicGeoprocessor
  - Clip, Dissolve, Intersect, Merge, Union, etc.
- ITopologicalOperator
  - Buffer, Clip, Cut, Simplify, etc.
- ITopologicalOperator2
  - ConstructBuffers, ClipToDomain

Buffer Example

```
'This sub should buffer the first feature layer in map with graphics
  'Declarations
  ..........
  Dim pTopoOperator As ITopologicalOperator
  Dim pFeatureCursor As IFeatureCursor
  Dim pFeature As IFeature
  Dim pElement As IElement
  Dim pGraphicsContainer As IGraphicsContainer
  ..........
  'set the graphics container
  Set pGraphicsContainer = pMap
  'get the layer and feature class
  ..........
  'set up feature cursor, loop through and buffer each feature, add graphic to map
  Set pFeatureCursor = pBufferFC.Search(Nothing, True)
  Set pFeature = pFeatureCursor.NextFeature
  Do Until pFeature Is Nothing
     Set pTopoOperator = pFeature.Shape
     Set pElement = New PolygonElement
     pElement.Geometry = pTopoOperator.Buffer(2500)
     pGraphicsContainer.AddElement pElement, 0
     Set pFeature = pFeatureCursor.NextFeature
  Loop
  'refresh the view
  pMxDoc.ActiveView.Refresh
```