A Brief Tour of ArcObjects and ArcGIS

intro to arcobjects

types of classes used in arcobjects

reading the object model diagram

vba editor for arcgis

maps

types of data layers

tables and fields

Matt Marsik
Department of Geography
March 25, 2008
Why ArcObjects?

ArcObjects is one development platform for ArcGIS Desktop.

ArcObjects is the development platform for the ArcGIS family of applications such as ArcMap, ArcCatalog, and ArcScene.

ArcObjects is a framework that lets you create domain-specific components from other components.

Learning ArcObjects allows you to pick apart existing scripts to understand process flow and create new scripts.
Classes and objects

There are 3 types of classes shown in the UML diagrams—abstract classes, co classes, and classes.

A coclass represents objects that you can directly create using the object declaration syntax (the New keyword). In VBA, this is written as

```vba
Set pPoint = new Point
```

A class cannot be created with New, but objects can be created as a property of another class or by functions from another class.

```vba
Set pMxDoc = ThisDocument
```

An abstract class cannot be used to create new objects, but it is a specification for subclasses.

eg: “line” could be an abstract class for “primary line” and “secondary line”
Type Inheritance

- *Type inheritance* defines specialized classes that share properties and methods with a super class and have additional properties and methods.

This diagram shows that a primary line (creatable class) and secondary line (creatable class) are types of a line (abstract class).
Instantiation

• *Instantiation* specifies that one object from one class has a method with which it creates an object from another class.

• A common way to get at additional properties and methods

A pole object might have a method to create a transformer object.
Interpreting object model diagrams

ArcObjects object model diagrams are based on the UML notation, an industry-diagramming standard for object-oriented analysis and design.

The development environment, Visual Basic or other, lists all of the many classes and members but does not show the structure of those classes.

These diagrams enhance your understanding of the ArcObjects components.
Object model diagrams: The Key

**AbstractClass**
- Interface: Inherited interface
- Properties
- Methods

**CoClass**
- Interface: Inherited interface
- Properties
- Methods

**Class**
- Interface: Inherited interface
- Properties
- Methods

**Types of Classes**
- **Abstract class**: cannot be used to create new objects, it is a specification for instances of subclasses (through type inheritance.)
- **CoClass**: can directly create objects by declaring a new object.
- **Class**: cannot directly create objects, but objects of a class can be created as a property of another class or instantiated by objects from another class.

**Types of Relationships**
- **Association**: represent relationships between classes. They have defined multiplicities at both ends.
- **Type inheritance**: defines specialized classes of objects that share properties and methods with the superclass and have additional properties and methods. Note that interfaces in superclasses are not duplicated in subclasses.
- **Composition**: specifies that one object from one class has a method with which it creates an object from another class.
- **N-ary association**: specifies that more than two classes are associated. A diamond is placed at the intersection of the association branches.

**Special Interfaces**
- **Property Get**: represents interfaces that are inherited by some subclasses but not all. The subclasses list the optional interfaces they implement.
- **Property Put**: represents interfaces that are only on specific instances of the class.
- **Property Get/Put**: (Optional) represents interfaces with a method that performs both get and put operations.
- **Property Put by Reference**: represents interfaces that perform put operations using a reference.
- **Function**: represents interfaces with a method that performs a specific action.
- **Event function**: indicates the name of the helper class required to support this event interface in Visual Basic.

**Multiplicity**
- 1 - One and only one (if none shown, one is implied)
- 0..1 - Zero or one
- M..N - From M to N (positive integers)
- * or 0..* - From zero to any positive integer
- 1..* - From one to any positive integer
Our ArcObjects Testbed
Getting at the VBA editor for ArcMap
VBA Editor Revealed and the Autocode Completion helper

Autocode Completion

```vba
Private Sub showLayers()
    dim pMxDoc as IMxDoc
End Sub
```

The Autocode Completion feature in the VBA Editor helps in automatically completing code as you type. It suggests relevant code snippets based on the context and the code you have already written.
Getting At Layers – Iterating Through an Enumeration of Layers

' setup Class interface (pointer) variables
Dim pMxDoc As IMxDocument
Dim pMap As IMap
Dim pLayers As TNumLayer
Dim pLayer As ILayer

Private Sub showLayers()
    ' pass the Class to object variables
    Set pMxDoc = ThisDocument
    Set pMap = pMxDoc.FocusMap
    Set pLayers = pMap.Layers

    ' display number of layers in map
    Debug.Print pMap.LayerCount

    ' since we're using a enumeration of layers need to start at the top
    pLayers.Reset

    ' get first layer in enumeration
    Set pLayer = pLayers.Next

    ' iterate through layer enumeration
    Do While Not pLayer Is Nothing
        ' print layer name
        Debug.Print pLayer.Name

        ' get next layer in enumeration
        Set pLayer = pLayers.Next
    Loop
End Sub
Enumerators – IEnumLayer example

• Enumerators are similar to collections but do not have as many properties and methods

• Does not have properties Count or Item

• Next returns next ILayer

• Reset moves to the top of enumeration

' setup Class interface (pointer) variables
Dim pMxDoc As IMxDocument
Dim pMap As IMap
Dim pLayers As IEnumLayer
Dim player As ILayer

Private Sub showLayers()
    ' pass the Class to object variables
    Set pMxDoc = ThisDocument
    Set pMap = pMxDoc.FocusMap
    Set pLayers = pMap.Layers

    ' display number of layers in map
    Debug.Print pMap.LayerCount

    ' since we're using a enumeration of layer
    pLayers.Reset

    ' get first layer in enumeration
    Set player = pLayers.Next

    ' iterate through layer enumeration
    Do While Not player Is Nothing
        ' print layer name
        Debug.Print player.Name
        player = pLayers.Next
    Loop

    End Sub
Getting At Layers – Instantiations of ILayer

Private Sub getTopLayer()
  ' pass the Class to object variables
  Set pMxDoc = ThisDocument
  Set pMap = pMxDoc.FocusMap

  ' get top layer
  Set pLayer = pMap.Layer(0)
  Debug.Print pLayer.Name

  ' get bottom layer
  Set pLayer = pMap.Layer(pMap.LayerCount - 1)
  Debug.Print pLayer.Name
End Sub

Private Sub getSelectedLayer()
  ' pass the Class to object variables
  Set pMxDoc = ThisDocument
  Set pMap = pMxDoc.FocusMap

  ' get selected layer
  Set pLayer = pMxDoc.SelectedLayer

  ' test for a selected layer
  If pLayer Is Nothing Then Exit Sub

  Debug.Print pLayer.Name
End Sub
Collections – IMap and ILayer examples

• Collection of maps or layers
• Each collection item is numbered
• Reference items by index number (not by name)

Properties
• Count layers in collection
• Get a layer with an index number

Methods

Private Sub getTopLayer()
    ' pass the Class to object variables
    Set pMxDoc = ThisDocument
    Set pMap = pMxDoc.RootMap

    ' get top layer
    Set player = pMap.Layer(0)
    Debug.Print player.Name

    ' get bottom layer
    Set player = pMap.Layer(pMap.LayerCount - 1)
    Debug.Print player.Name
End Sub
Collections – ILayer and selected layers

Private Sub getSelectedLayer()
' pass the Class to object variables
Set pMxDoc = ThisDocument
Set pMap = pMxDoc.FocusMap

' get selected layer
Set pLayer = pMxDoc.SelectedLayer

' test for a selected layer
If pLayer Is Nothing Then Exit Sub

Debug.Print pLayer.Name
End Sub
Collections – IFeatureLayer and IFields

• Feature classes have attribute tables
• Need to get table from feature class
• Tables contain a Fields collection
• Fields collection is composed of fields
• Use IFields and IField to get properties

Private Sub getFeatureLayer(pLayer As ILayer)
' setup Class interface (pointer) variables
Dim pFeatureLayer As IFeatureLayer
Dim pFeatureClass As IFeatureClass
Dim pField As IField
Dim pFields As IFields

' instantiate feature layer from generic layer class
Set pFeatureLayer = pLayer
' instantiate feature class from the feature layer class
Set pFeatureClass = pFeatureLayer.FeatureClass
' instantiate fields class from feature class interface
Set pFields = pFeatureClass.Fields
For i = 0 To pFields.FieldCount - 1
' instantiate single field from the fields collection
    Set pField = pFields.Field(i)
' print some field properties
    Debug.Print pField.Name
    Debug.Print pField.Length
    Debug.Print pField.Type
    Debug.Print pField.Precision
    Debug.Print pField.Scale
Next
End Sub

IField Interface (esriGeoDatabase)

Example

Provides access to members that return information about the field.

Product Availability
Available with ArcGIS Engine, ArcGIS Desktop, and ArcGIS Server.

Description
The field object represents a column in a table. A field has many properties, the most obvious ones being its name and its datatype. The esriFieldType enumeration lists the possible datatypes.

Members
IFields: What’s Next?

• Instantiate more interfaces:
  • ITable
  • ICursor
  • IRow
  • IQueryFilter

Why? To create selections, edit table cells and summarize fields.

```vbscript
For i = 0 To pPtFields.FieldCount - 1
    Set pTField = pPtFields.Field(i)
    strFldName = pTField.Name
    If Left(strFldName, 1) = “Y” Then 'Test for the fields that start with Y
        If strFldName = “Y1959” Then
            lngDateFldCnt = lngDateFldCnt + 1
        Dim pPolyTo1 As ITable
        Dim pPolyCursor As ICursor
        Dim pPolyRow As IRow
        Dim strBasinFldVal As String
        Set pPolyTo1 = pFLyrPoly.FeatureClass
        Set pPolyCursor = pPolyTo1.Search(Nothing, False)
        Set pPolyRow = pPolyCursor.NextRow
        lngPolyFeatCnt = pFClsBasin.FeatureCount(Nothing)
        'iterate thru rows in subbasin pFLyrPoly
        lngPolyCnt = 0
        Do While Not pPolyRow Is Nothing
            lngPolyCnt = lngPolyCnt + 1 'count subbasin polys
            strBasinFldVal = pPolyRow.value(pPolyRow.Fields.FindField("DESCRIPTIO")) 'make DESCRIPTIO field dynamic
            'create header info and print to file when necessary
            If lngPolyCnt = 1 Then
                strFileHdr = strFileHdr & ","
            Else
                strFileHdr = strFileHdr & strBasinFldVal
            End If
            'create selection from pFLyrPoly based on subbasin name
            Dim pQF As IQueryFilter
            Set pQF = New QueryFilter
            pQF.WhereClause = "DESCRIPTIO = " & strBasinFldVal & "" 'dynamic
            'create featcursor based on query
            Set pFeatCrsr = pFLyrPoly.FeatureClass.Search(pQF, False)
            Set pGeomCol = New GeometryBag
            'create pFeatCrsr and increment for each spatial feature used in the spatial selection
            Set pFeature = pFeatCrsr.NextFeature
            Do While Not pFeature Is Nothing
                pGeomCol.AddGeometry pFeature.Shape
                Set pFeature = pFeatCrsr.NextFeature
                Loop
            Do While Not pFeature Is Nothing
                pGeomCol.AddGeometry pFeature.Shape
                Set pFeature = pFeatCrsr.NextFeature
                Loop
```
IFeatureLayer: What’s Next?

• Instantiate more interfaces:
  • IWorkspaceName
  • IFeatureClassName
  • ITable

• Why? Perform spatial operations between 2 or more vector layers

IBasicGeoprocessor Interface (esriCarto)

See Also

Provides access to members that control the basic geoprocessor.

Product Availability

Available with ArcGIS Engine, ArcGIS Desktop, and ArcGIS Server.

When To Use

IBasicGeoprocessor provides access to the methods and properties of the BasicGeoprocessor object. The functionality of the BasicGeoprocessor object had been superseded by the new Geoprocessor framework of ArcGIS starting at version 9.0. Developers should utilize the new framework whenever possible.

Members

<table>
<thead>
<tr>
<th>All</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CancelTracker</td>
<td>The cancel tracker.</td>
</tr>
<tr>
<td>Clip</td>
<td>Clips features.</td>
</tr>
<tr>
<td>Dissolve</td>
<td>Dissolves features.</td>
</tr>
<tr>
<td>Intersect</td>
<td>Intersects features.</td>
</tr>
<tr>
<td>Merge</td>
<td>Merges features.</td>
</tr>
<tr>
<td>SpatialReference</td>
<td>The output spatial reference.</td>
</tr>
<tr>
<td>Union</td>
<td>Creates a union of features.</td>
</tr>
</tbody>
</table>

CoClasses that implement IBasicGeoprocessor

<table>
<thead>
<tr>
<th>CoClasses and Classes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BasicGeoprocessor</td>
<td>A basic geoprocessor.</td>
</tr>
</tbody>
</table>
Getting At Raster Layers – Instantiation of IRasterLayer

Private Sub passILayer()
  ' pass the Class to object variables
  Set pMxDoc = ThisDocument
  Set pMap = pMxDoc.FocusMap

  ' get selected layer
  Set pLayer = pMxDoc.SelectedLayer

  ' test for a selected layer
  If pLayer Is Nothing Then Exit Sub

  ' test for a selected raster layer
  If Not pLayer Is IRasterLayer Then
    Debug.Print "Layer is not a raster! Select a raster layer in the TOC."
    Exit Sub
  End If

  ' send ILayer to RasterLayer sub
  Call getRasterLayer(pLayer)
End Sub

Private Sub getRasterLayer(pLayer As ILayer)
  Dim pRasterLayer As IRasterLayer

  ' instantiate the IRasterLayer interface
  Set pRasterLayer = pLayer

  ' print out some basic info now we have the raster
  Debug.Print pRasterLayer.Name
  Debug.Print pRasterLayer.BandCount
  Debug.Print pRasterLayer.RowCount
  Debug.Print pRasterLayer.FilePath
End Sub
IRasterLayer Interface (esriCarto)

See Also

Provides access to members that create or modify a raster layer.

Product Availability

Available with ArcGIS Engine, ArcGIS Desktop, and ArcGIS Server.

Members

- AreaOfInterest
- BandCount
- Cached
- ColumnCount
- CreateFromDataset
- CreateFromFilePath
- CreateFromRaster
- DataFrameExtent
- DisplayResolutionFactor
- Draw
- FilePath
- MaximumScale
- MinimumScale
- Name
- PrimaryField
- PyramidPresent
- Raster
- Renderer
- RowCount
- ShowResolution
- ShowTips
- SpatialReference
- SupportedDrawPhases
- TipText
- Valid
- Visible
- VisibleExtent

Private Sub getRasterLayer(pLayer As ILayer)
    Dim pRasterLayer As IRasterLayer
    Set pRasterLayer = pLayer

    ' instantiate the IRasterLayer interface
    Set pRasterLayer = pLayer

    ' print out some basic info now we have the raster
    Debug.Print pRasterLayer.Name
    Debug.Print pRasterLayer.BandCount
    Debug.Print pRasterLayer.ColumnCount
    Debug.Print pRasterLayer.RowCount
    Debug.Print pRasterLayer.FilePath
IRasterLayer – What’s Next?

• Instantiate more interfaces:
  • IRaster
  • IRasterDataset
  • IRasterWorkspace
  • IRasterBands
  • IRasterProps

  • Why? To get at more raster properties and to create and edit rasters programatically.
What’s next?

Anything’s possible!

ArcGIS Developer Help
C:\Program Files\ArcGIS\DeveloperKit\Help\VB\VB6_Desktop.chm

Object Model Diagrams
C:\Program Files\ArcGIS\DeveloperKit\Diagrams

Check out existing VBA scripts for ArcGIS
http://arcscripts.esri.com/

Look on Discussion forums for more info/help
http://support.esri.com/index.cfm?fa=forums.gateway

Integrating Python and ArcObjects
http://www.codeplex.com/IronPython