

# Developing with ArcObjects APIs --Based on ArcGIS 10

周岳昆

Esri中国（北京）培训中心

[zyk13032@gmail.com](mailto:zyk13032@gmail.com)

# Presentation Outline

- ✓ Introduction of ArcObjects and .Net SDK
- Customize and Extending ArcObjects
- Access Data and Display Layer
- Feature Creation and Editing
- All Below Is Geometry
- Query Selection and Analysis
- Display and Export Map
- Deploying the Application (Engine)
  - Extending 1: ArcGIS Raster APIs
  - Extending 2: Spatial Network APIs

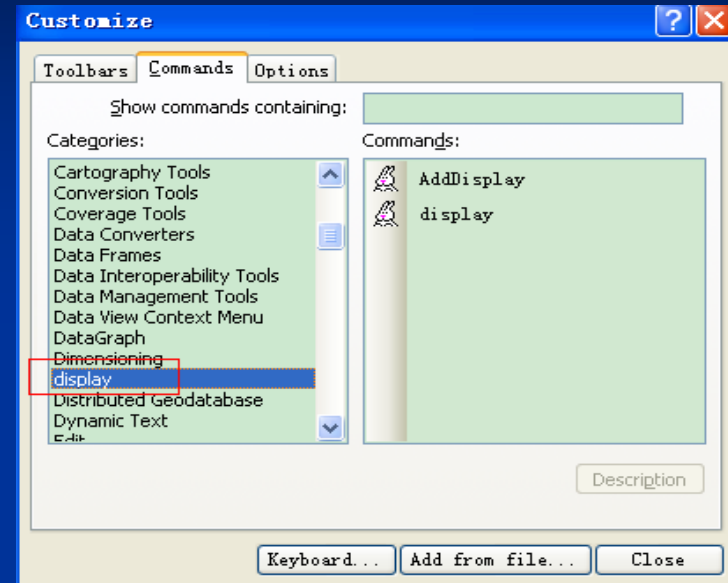
# What Is About ArcObjects

- ArcObjects is a library of COM (Component Object Model) that make up the foundation of ArcGIS.
- ArcObjects is different from ArcGIS Engine.
  - Engine Runtime is a part of ArcObjects library
  - ArcObjects can consume as Desktop, Engine or Server, while Engine Runtime only support Engine.
  - When setup ArcGIS application (e.g. ArcMap), ArcObjects is installed ;while Engine is a standalone application using Runtime and SDK (e.g. .NET)

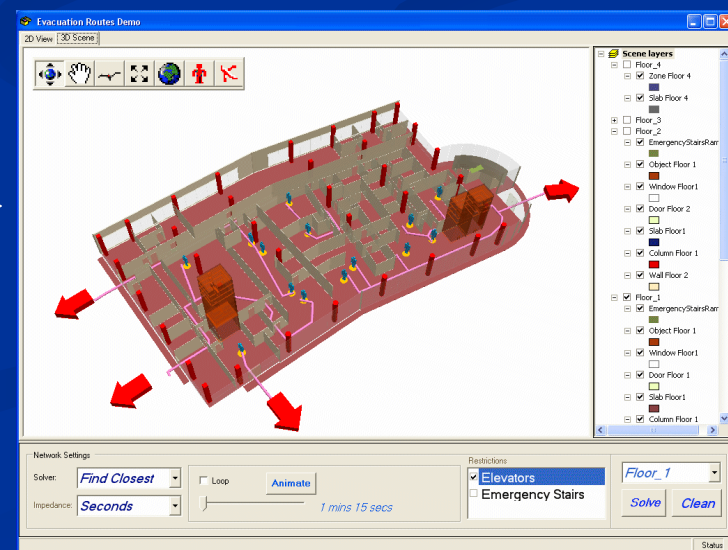
# What Can ArcObjects Do



Extending  
ArcObjects



Standalone  
Application



# Determine Which Application You Need

Don't Forget Add-In

- To customize the ArcGIS Desktop applications  
([ArcObjects Extending](#))
- To build standalone mapping applications  
([ArcGIS Engine](#))
- To develop Web applications  
(ArcGIS Server)

# A complete ArcGIS

## Applications

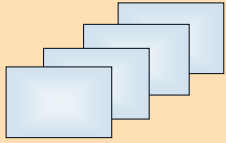
**ArcGIS Desktop**



---

Desktop Developer Kit  
.NET


**ArcGIS Engine**



---

Engine Developer Kit  
.NET  
C++ Java

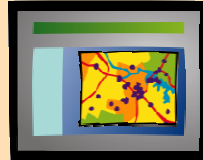
**ArcGIS Explorer**



---

API  
.NET

**Web Application**



---

API  
.NET Java  
JavaScript Flex

**ArcGIS Mobile**



---

SDK  
.NET

**ArcObjects**

## Services



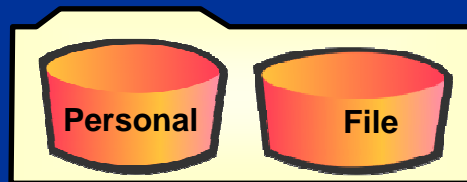
## ArcGIS Server



OGC SOAP  
KML REST

## ArcGIS Online

## Data (Geodatabase)



**ArcSDE**

# ArcObjects Core Library

- esriCarto: *Layer, Map, FeatureSelection, Annotation, Element*
- esriDisplay: *Symbol, Color, Display, ScreenDisplay*
- esriGeoDatabase: *Workspace, FeatureClass, Dataset, Feature, Row, Field, Cursor, FeatureCursor*
- esriGeometry: *Point, Polyline, Polygon, SpatialReference*
- esriOutput: *Export, IPrinter*
- esriSystem: *Array, Set*
- esriControls: *MapControl, ToolbarControl, TOCControl, LicenseControl e.g. ( only Supported by Engine )*
- esriSystemUI: *ITool, ICommand ( Unique UI library in Engine)*
- ADF.BaseClasses: *BaseTool, BaseCommand*

# More Libraries

- `esriDataSourceFile`
- `esriDataSourceGDB`
- `esriDataSourceRaster`
- `esriGeoProcessing`



# What Is New In ArcGIS 10

- A single SDK for ArcObjects, which combined Engine, Desktop and Server.
- Customize ArcGIS Desktop using more Add-Ins other than VBA.
- Map automation using python
- Some libraries have enhanced
  - esriCarto add MosaicLayer class to show mosaic dataset.
  - esriDataSourceRaster add MosaicRasterDataset class
  - esriGeoDatabase add IQueryDef2 interface to support postfix clause such as Order by or Group by.
  - esriDisplay add IStyleGalleryItem2 interface to access symbol or element by Tag property

# When Need Help

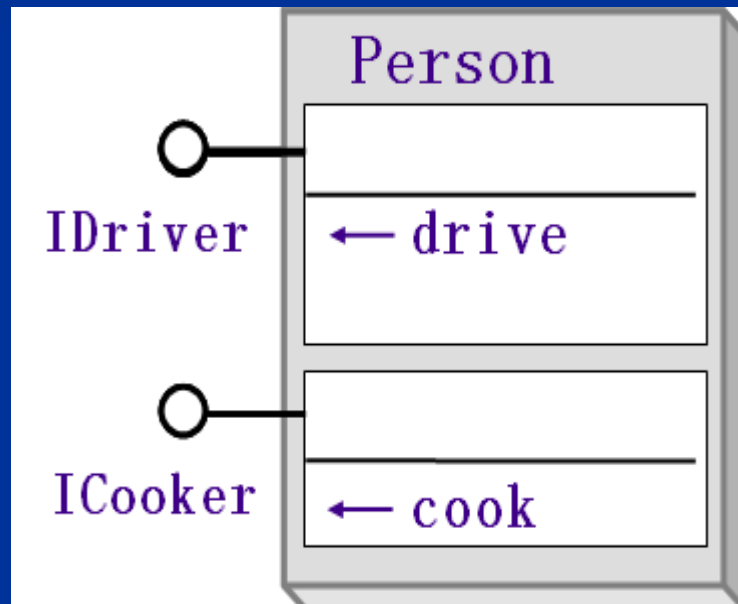
- Developer Help Local
  - Documentation and Samples
- Object Model Diagrams
- Esri Developer Network, Go to: [edn.esri.com](http://edn.esri.com),  
[resource.esri.com](http://resource.esri.com), [support.esri.com](http://support.esri.com)
  - Documentation Library
  - Technical Articles and White Papers
  - Data Model
  - Samples and Solutions
- Forums
  - [www.gisall.com](http://www.gisall.com) (Esri)

## Also Need Some COM

- COM is a binary protocol, .dll or .exe  
See at <directory>/ArcGIS/com/ and <directory>/ArcGIS/Bin
- All communication happens through interfaces, and when registered never changed.
  - In-Bound Interface: Method and Attribute
  - Out-Bound Interface: Events
- Polymorphism
- Query Interface (QI)  
What do you think what Query Interface is ?

# QI Once More, It's more Important

- One Dramatic Example is Enough



```
IDriver p1=new Person();
```

```
p1.drive();
```

```
//p1 and p2 are titles of the same person
```

```
ICooker p2=p1; // QI
```

```
p2.cook();
```

Think Why ?

# Wrapper In .Net SDK

- MapControl: Map object
- PageLayoutControl: PageLayout object
- ToolbarControl: Container for commands, tools, and menus
- TOCControl: Displays layer information for an associated control
- LicenseControl: Performs license initialization
- Other Controls...



# Get Ready?

- ArcGIS Engine Runtime is better. ArcObjects SDK is appropriate.
- Microsoft .Net Framework 3.5 at least and Visual Studio 2008 or above version.

# Presentation Outline

- Introduction of ArcObjects and .Net SDK
- ✓ *Customize and Extending ArcObjects*
- Access Data and Display Layer
- Feature Creation and Editing
- All Below Is Geometry
- Query Selection and Analysis
- Display and Export Map
- Deploying the Application (Engine)
  - Extending 1: ArcGIS Raster APIs
  - Extending 2: Spatial Network APIs

# ArcGIS Functions Are All Commands

- Commands are organized into categories
- COM should be registered in correct categories
  - ArcMap Extending: AxCommands
  - ArcGIS Engine: CommandControls
- Inherit from BaseClass to override functionality
  - BaseCommand
  - BaseTool
  - BaseToolbar
  - BaseMenu



# OnCreate Method

- Called when command is initialized
- Provides a *hook* referring to object that created it
  - *hook* is MxApplication, when ArcMap is referred
  - *hook is* AxMapControl, when MapControl is referred
- IHookHelper Interface (only Engine)
  - Store *hook*
  - Access the Map, PageLayout and ActiveView by IHookHelper Attributes

# Samples: Create Command

```
public class AddLayer : BaseCommand {  
    private IHookHelper m_hookHelper ;  
    private IMapControl4 m_mapControl ;  
    public AddLayer () { // code some base property here }  
    public override void OnCreate(object hook) {  
        if (hook == null) return;  
        if (m_hookHelper == null) m_hookHelper = new HookHelper ();  
        m_hookHelper.Hook = hook;  
        if ( hook is MapControl ) m_mapControl = hook as IMapControl4; }  
    public override void OnClick() {  
        // code the function when click the command }  
}
```

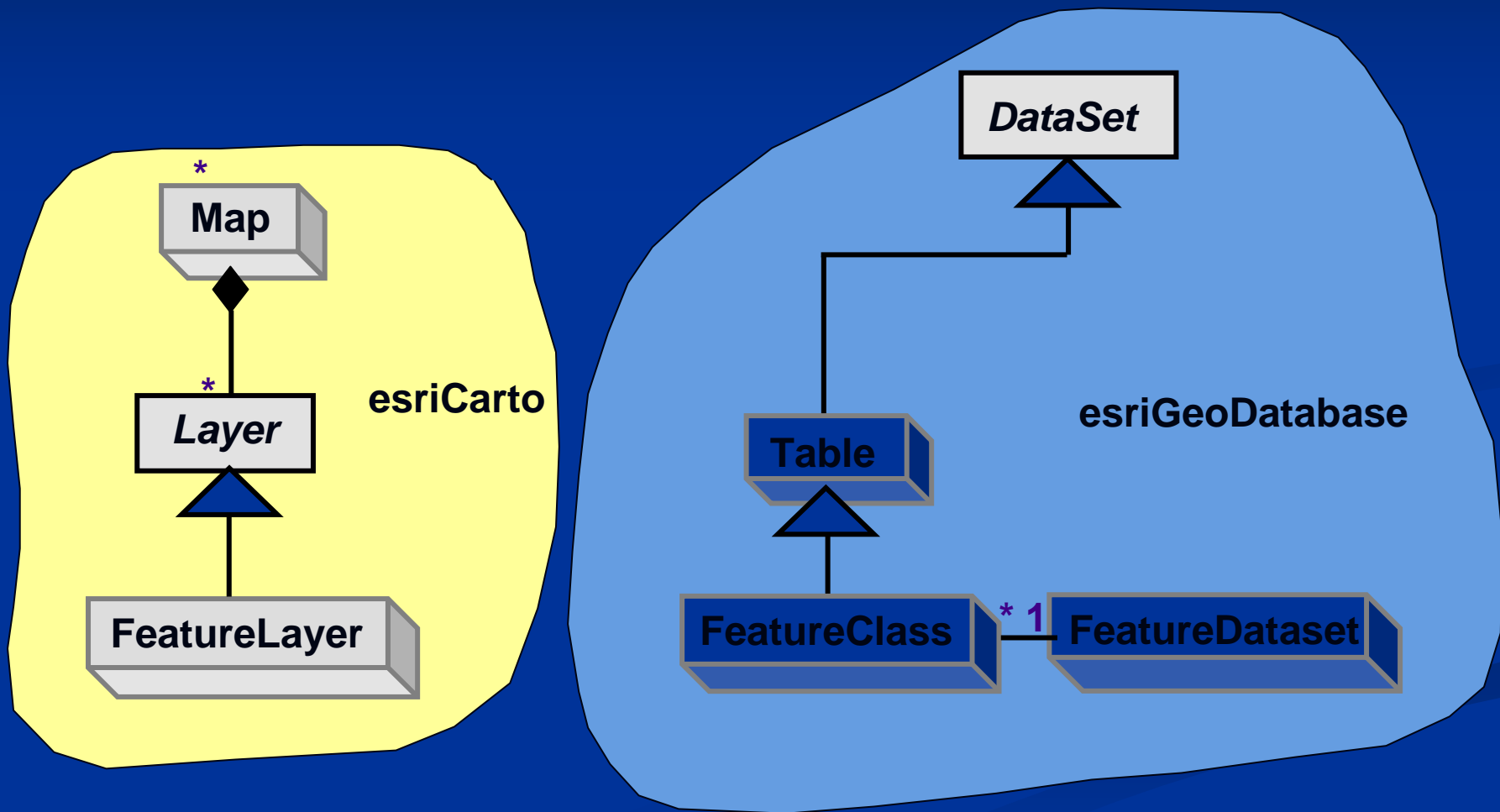
# Add-In: More Convenient Customizing

- Add-In provides a simple and light-weight approach to most common customizations, which use to add custom functionality to ArcGIS Desktop.
- Convenient to custom buttons, tools, menus, palettes, dockable windows, combo boxes, multi-items, application and editor extensions.

# Presentation Outline

- Introduction of ArcObjects and .Net SDK
- Customize and Extending ArcObjects
- ✓ *Access Data and Display Layer*
- Feature Creation and Editing
- All Below Is Geometry
- Query Selection and Analysis
- Display and Export Map
- Deploying the Application (Engine)
  - Extending 1: ArcGIS Raster APIs
  - Extending 2: Spatial Network APIs

# Display Layer Overview



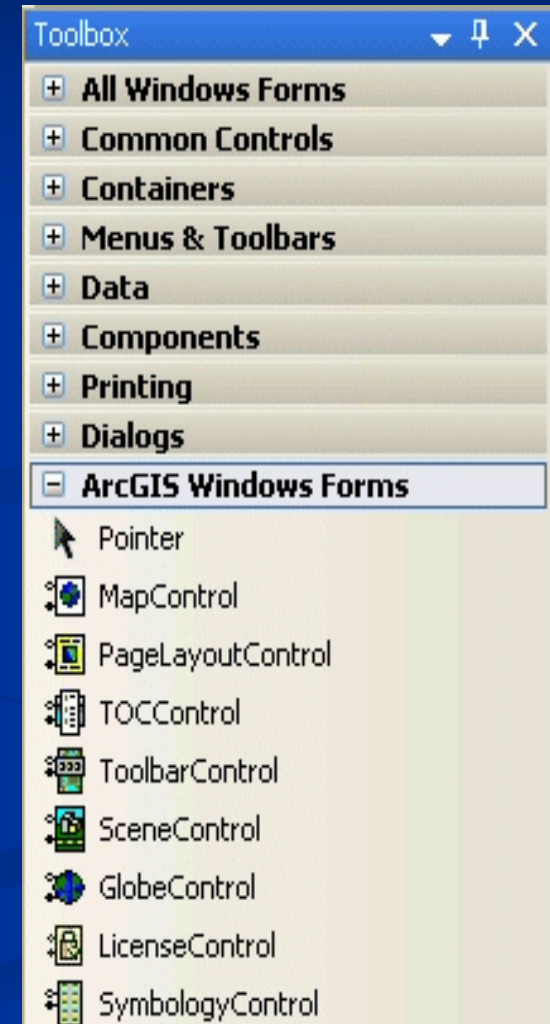
# Review Controls in Engine SDK

- Create a MapControlApplication
- Automation to develop
  - Add map document to MapControl
  - Add items to ToolbarControl
  - Remember: SetBuddyControl

AxMapControl and  
IMapControl



What is the different?



# Wrapper In .Net SDK

- MapControl: Map object
- PageLayoutControl: PageLayout object
- ToolbarControl: Container for commands, tools, and menus
- TOCControl: Displays layer information for an associated control
- LicenseControl: Performs license initialization
- Other Controls...



# IMapControl4 Interface

- Access to members that control the MapControl
  - AddLayer: add layer into map
  - CheckMxFile: return Boolean whether Mx file is exist
  - ReadMxMaps: return Array Mx maps get
  - LoadMxFile: load specific map into MapControl
  - Map: map contained by MapControl
- Remember: One MapControl contains one Map only, other than Maps.
- Review the help of IMapControl4:
  - Map, ActiveView, CustomProperty
  - FlashShape, ToMapPoint



# Work With Map's Layer

- `IMap.get_Layer(item)`
- `IEnumLayer::Next` returns `ILayer`

```
UID uid=new UIDClass()
```

```
uid.value="{40A9E885-5533-11d0-98BE-00805F7CED21}"
```

```
// FeatureLayer, second parameter shows recursive is true
```

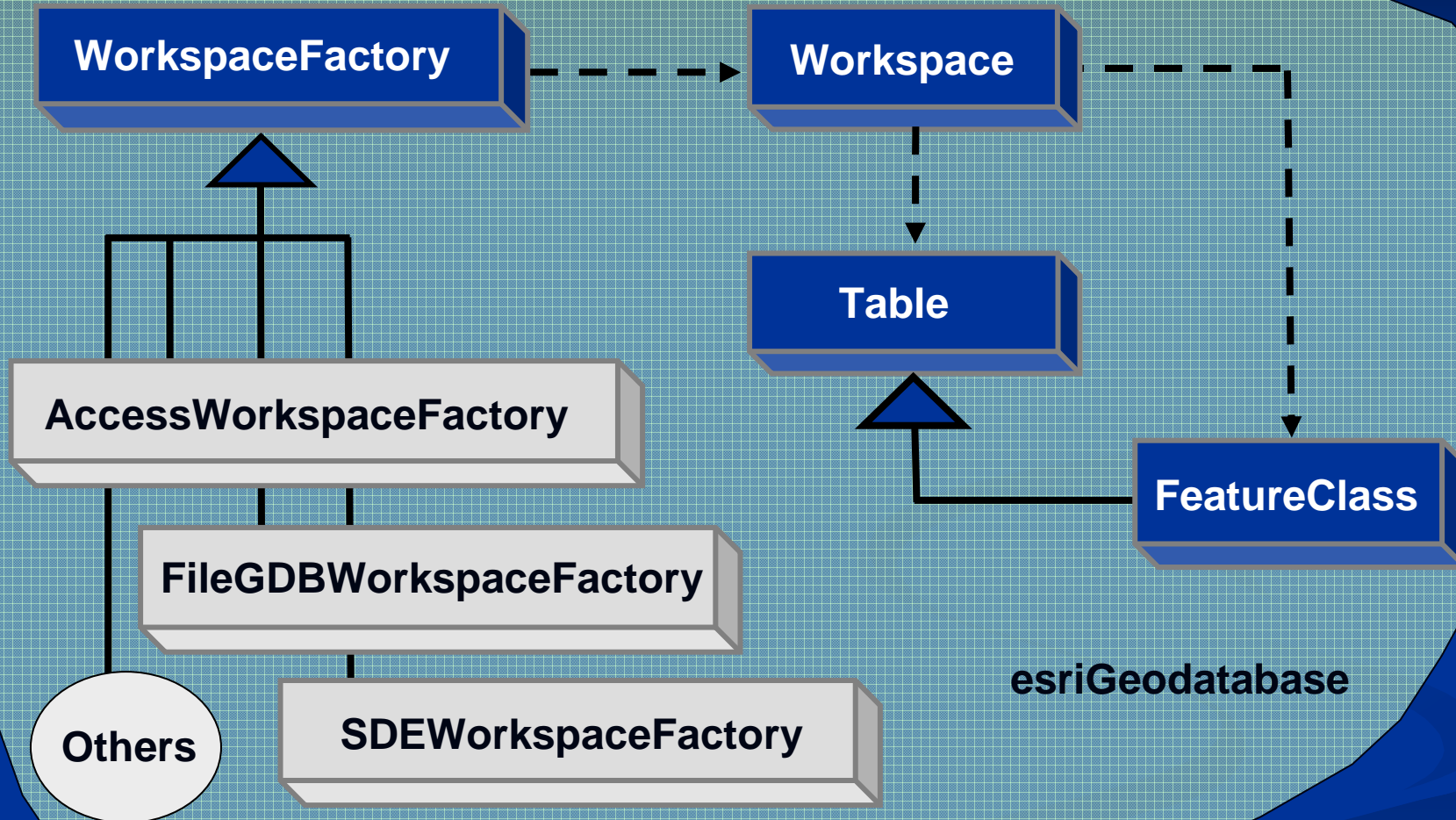
```
IEnumLayer layers=map.get_Layers(uid, true)
```

```
ILayer layer=layers.Next()
```

# IFeatureLayer Interface

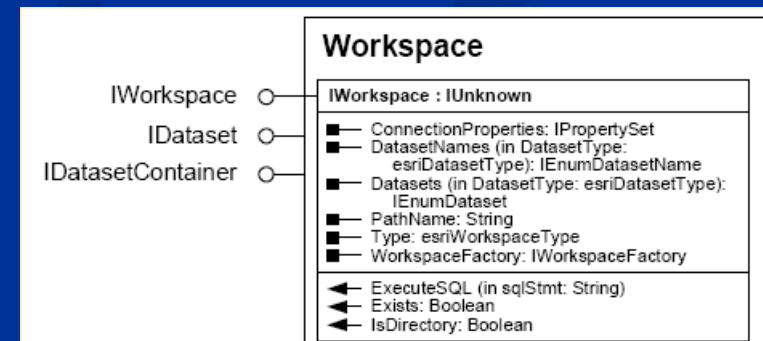
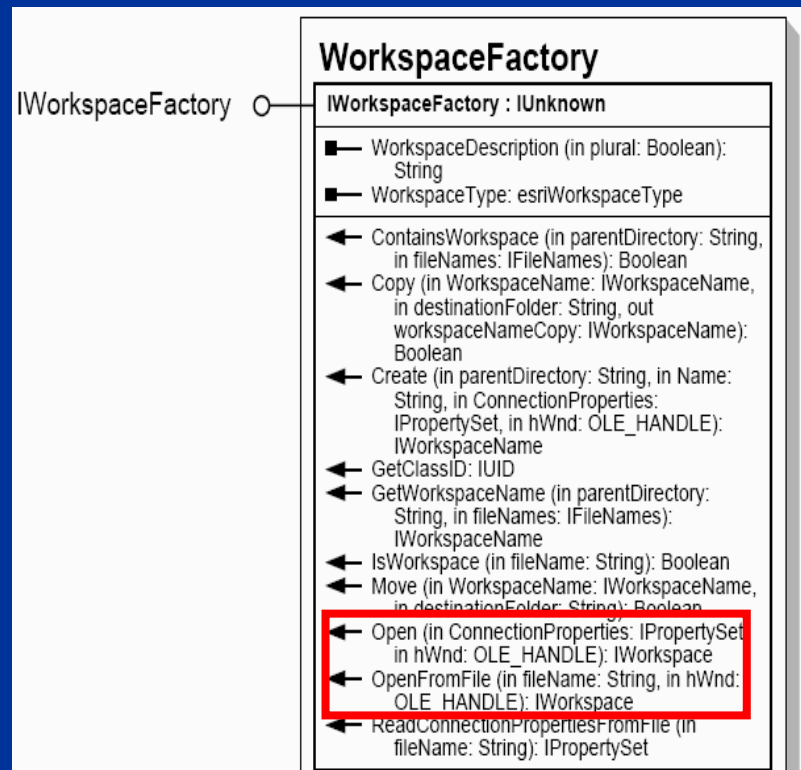
- This interface has properties that specify the feature class of the layer, which inherit ILayer.
  - FeatureClass: the layer's featureclass
  - SpatialReference: spatial reference in layer (Read)
  - Search: return a featureCursor based upon filter
    - Recycling true: cursor recycle at once point to next
    - Recycling false: cursor recycle until cursor is destroyed
- If want to access Render and Labeling properties, using IGeoFeatureLayer instead.

# Access Data Objects Overview



# Accessing Workspace

- IWorkspaceFactory to return Workspace object
  - OpenFromFile: Accesses an existing folder on disk
  - Open: Connects to existing database (e.g., ArcSDE)



# Accessing Feature Workspace

- QI to Feature Workspace
  - CreateFeatureClass: create new feature class
  - CreateTable: create new table
  - OpenFeatureClass: open an existing feature class
  - OpenTable: open an existing table

```
IFeatureWorkspace fws=workspace; // QI
```

```
IFeatureClass fclass=fws.openFeatureClass("Lake");
```

# Add New FeatureLayer to Map

- Create new FeatureLayer
- Access Feature Class to FeatureLayer
- Add FeatureLayer to map

```
IFeatureLayer flayer=new FeatureLayer();  
IFeatureWorkspace fws=workspace; // QI  
IFeatureClass fclass=fws.openFeatureClass("Lake");  
flayer.FeatureClass=fclass;  
map.AddLayer( flayer);
```

# Exercise 01: ArcGIS Control Application

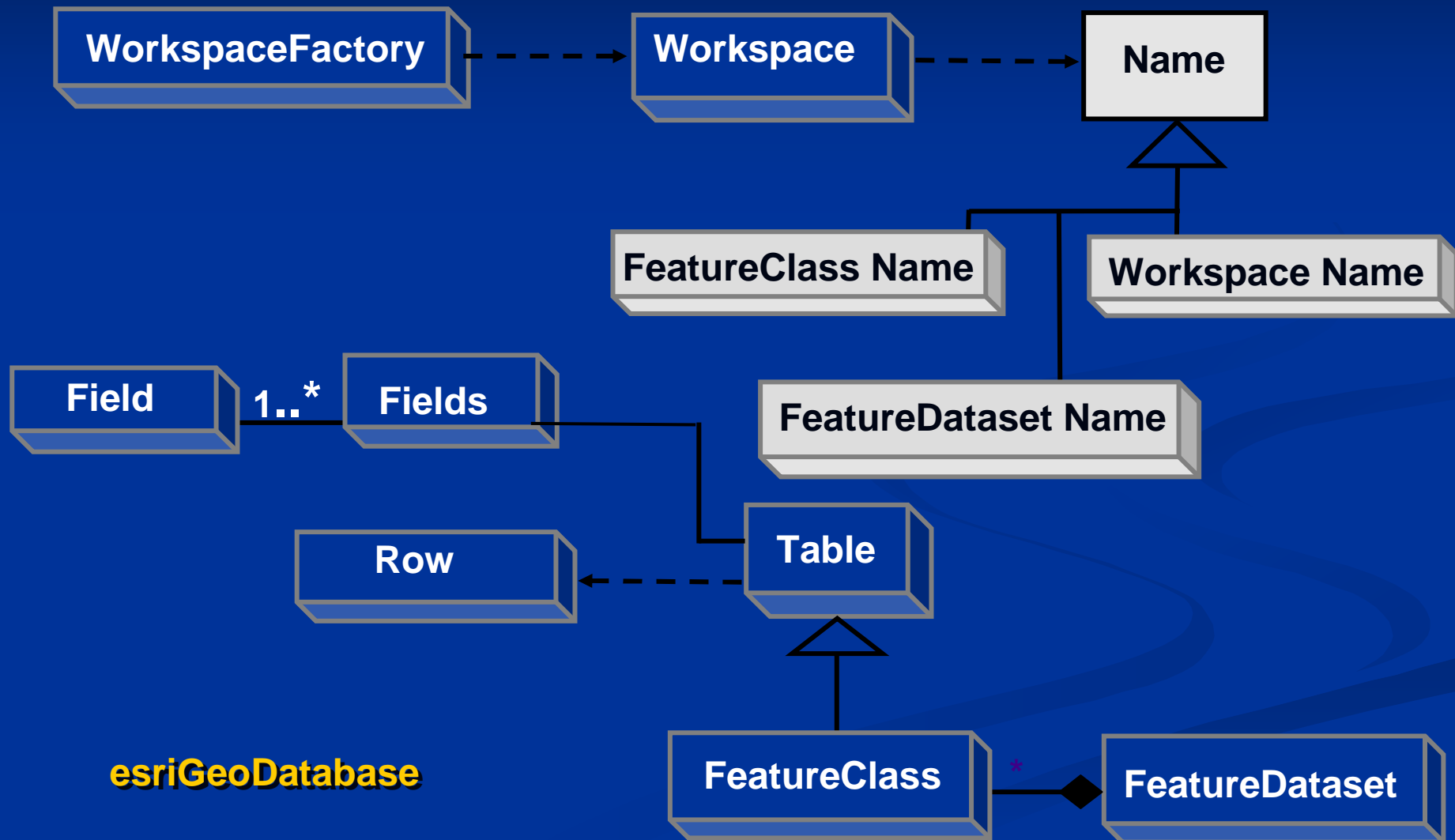
- Create MapControl Application
- Using IMapControl Interface
- Create Tab Control: Map and PageLayout
- Set Buddy Control
- Add Tab Control Events

# Presentation Outline

- Introduction of ArcObjects and .Net SDK
- Customize and Extending ArcObjects
- Access Data and Display Layer
- ✓ Feature Creation and Editing
- All Below Is Geometry
- Query Selection and Analysis
- Display and Export Map
- Deploying the Application (Engine)
  - Extending 1: ArcGIS Raster APIs
  - Extending 2: Spatial Network APIs



# esriGeoDatabase Model Overview



**esriGeoDatabase**

# Why Name Object

- Name objects are placeholders for real objects
- Use `IName::Open` to instantiate object represented

Name is a lightweight object. When workspace attributes( category ,type) only wanted, no need to load the whole workspace object into stack.

Name is like signature. Sometime when see that signature, it's enough to fulfill the task. And sometime we may see the leader by myself, that is **Open** Method.

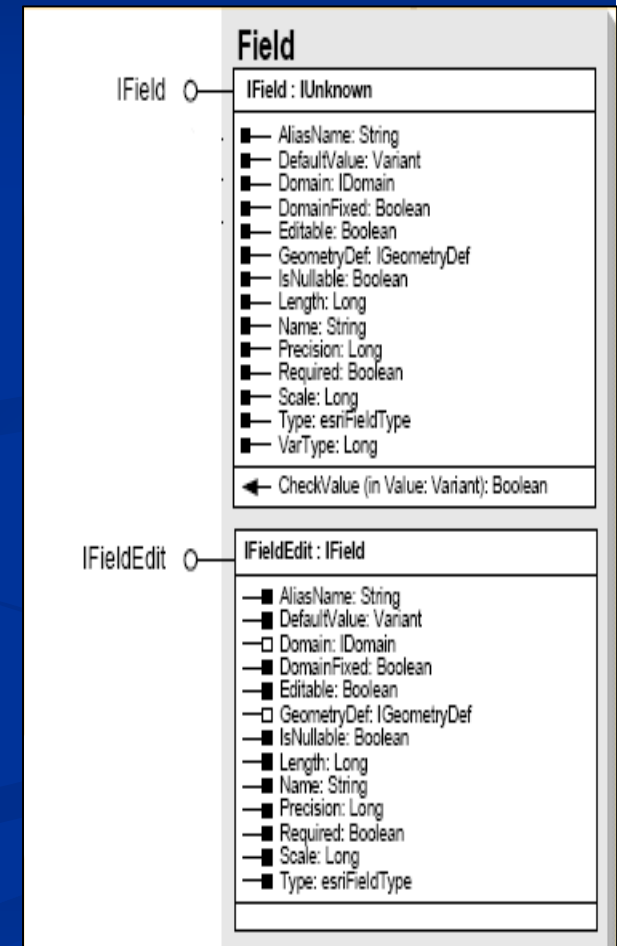
## Sample: Create New Workspace

```
IWorkspaceFactory wsf=new FileGDBWorkspaceFactory();  
// create new workspace (GDB), return error if existed  
IWorkspaceName wsn=wsf.Create( path ,name ,null , null);  
IName pName=(IName) wsn;  
// get the workspace through open method  
IWorkspace ws=pName.open();
```

# IField and IFields

- Field object
  - Both IField and IFieldEdit are Filed
  - Get field properties with IField
  - Set field properties with IFieldEdit
- Fields object
  - Use IFieldsEdit :AddField add field

Errors may occur when some filed properties are set to value. That may be ArcObjects' bug.



# Create New Data

## ■ IFeatureWorkspace Interface

- CreateTable: Required Common Fields, no Geometry

// Null shows CLSID and EXTCLSID custom behavior

// The last parameter shows ArcSDE configuration behavior

```
ITable table=fws.CreateTable("tableName",Fields,null,null,"")
```

- CreateFeatureClass: Required GeometryDef Fields and esriFeatureType

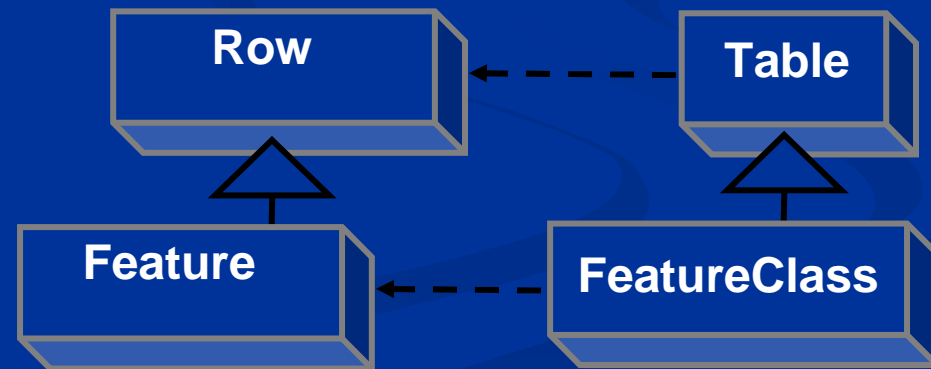
- GeometryDef defined GeometryType and SpatialReference

- esriFeatureType defined feature types (e.g. esriFTSimple)

# Adding Rows and Values to Feature

- ITable :: CreateRow returns a row
- IFeatureClass :: CreateFeature returns a feature

```
IFeature feature=fclass.CreateFeature( );  
feature.value(fclass.FindField("name"))="china";  
feature.store( );
```



# Cursor and FeatureCursor

- Used to accessing a subset of records
- IFeatureCursor inherit from ICursor
- Methods for working with:
  - Cursor: Rows in non-spatial tables
  - FeatureCursor: Features in feature class table
- Can be used for editing
  - Non-editing: Search
  - Editing: Update and Insert

# Editing with a Cursor

- Faster than using CreateRow or CreateFeature
  - Much more efficient for large operations
- Use to add, delete, or modify rows or features
  - Tables
    - ICursor :: InsertRow
    - ICursor :: DeleteRow
    - ICursor :: UpdateRow
  - Feature classes
    - IFeatureCursor :: InsertFeature
    - IFeatureCursor :: DeleteFeature
    - IFeatureCursor :: UpdateFeature



# Editing Cursors

## ■ Update cursor

*// filter is a QueryFilter*

*// false the same as IFeatureLayer.Search Method*

```
IFeatureCursor fCursor=fclass.Update(filter,false);
```

## ■ Insert cursor

*// false means non-use Row Buffer, Insert one by one*

*// true means use Row Buffer, Insert after cursor completed*

```
IFeatureCursor fCursor=fclass.Insert(true);
```

# Sample: Update and Insert Cursor

```
IQueryFilter filter=new QueryFilter();  
Filter.WhereClause=" StateName='Japan' ";  
IFeatureCursor fCursor=fclass.Update(filter,false);  
IFeature feature=fCursor.NextFeature();  
While (feature!=null)  
{  
    feature.Value(fcalss.FindField("StateName" ))="China";  
    fCursor.UpdateFeature(feature);  
    feature=fCursor.NextFeature();  
}
```

## Exercise 02: Create New FeatureClass

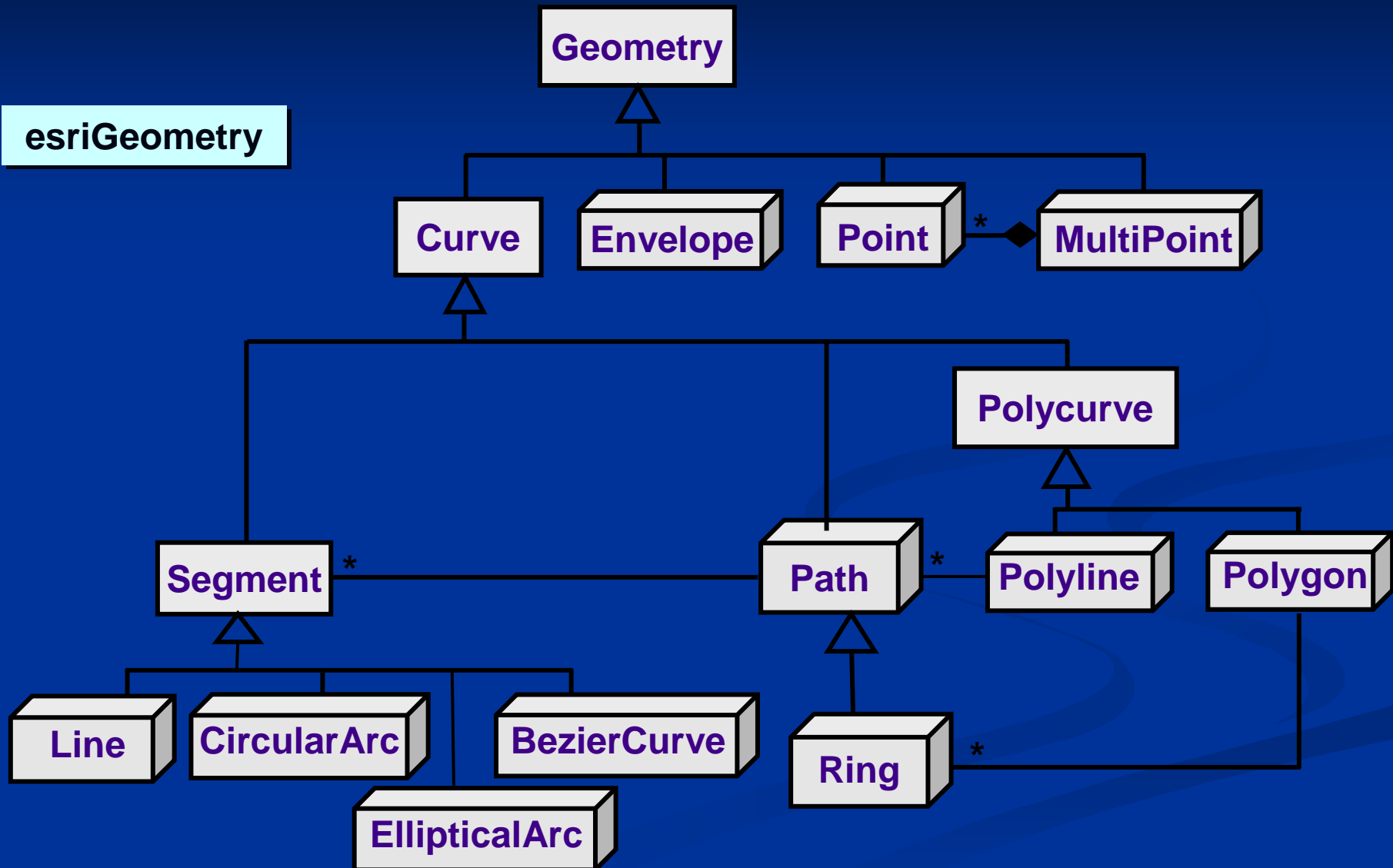
- Access feature workspace
- Create new field
- Add GeometryDef to geometry field
- Add new field to fields
- Using IFieldChecker Interface to check field
- Using CreateFeatureClass method

# Presentation Outline

- Introduction of ArcObjects and .Net SDK
- Customize and Extending ArcObjects
- Access Data and Display Layer
- Feature Creation and Editing
- ✓ *All Below Is Geometry*
- Query Selection and Analysis
- Display and Export Map
- Deploying the Application (Engine)
  - Extending 1: ArcGIS Raster APIs
  - Extending 2: Spatial Network APIs

# Geometry Object Overview

esriGeometry



# Point, Multipoint and PointCollection

- Points are zero dimensional
  - Define coordinates with X and Y properties
  - May have Z and M properties
- Multipoints are collections of points

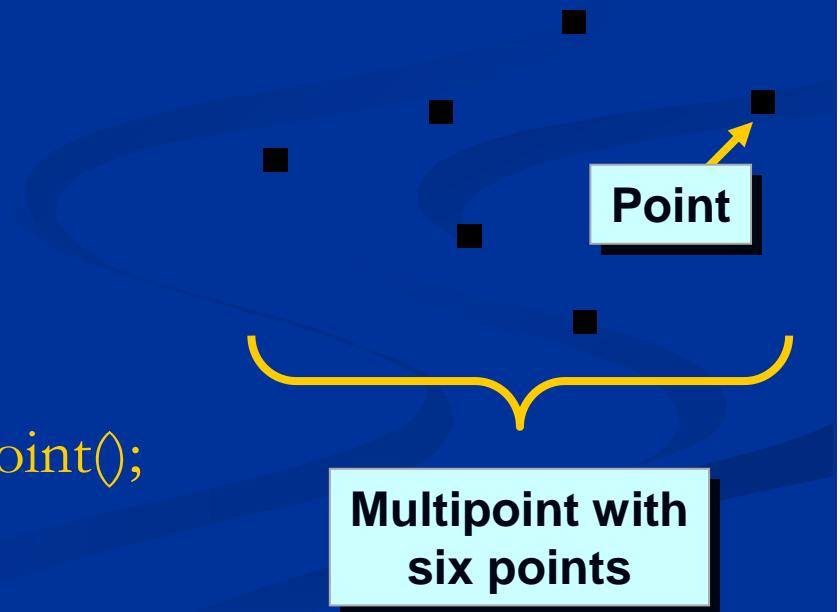
```
IPoint point=new Point();
```

```
point.X=300;
```

```
Point.Y=450;
```

```
IPointCollection points=new MultiPoint();
```

```
Points.AddPoint ( point );
```



# Segments

- Consist of two points and type line between them
  - Types of: Line, BezierCurve, CircularArc, EllipticalArc



- Segments are building blocks for other geometry
  - Paths, polylines, rings, and polygons

```
ILine line=new Line();
```

```
Line.FromPoint=fromPoint;
```

```
Line.ToPoint=toPoint;
```

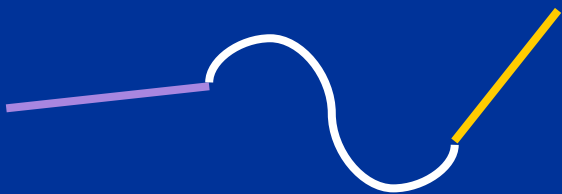
# Polyline and Polygon

- Polyline

- Collections of connected or unconnected paths

- Polygon

- Composed of one or several rings



One polyline with many segments



One polygon with many rings

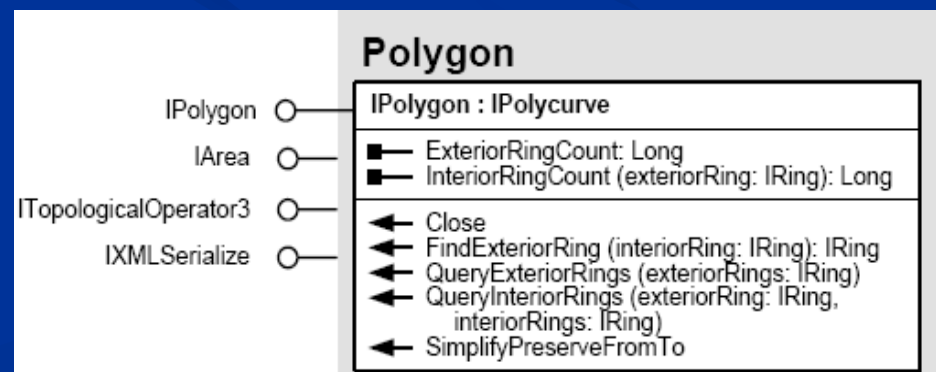
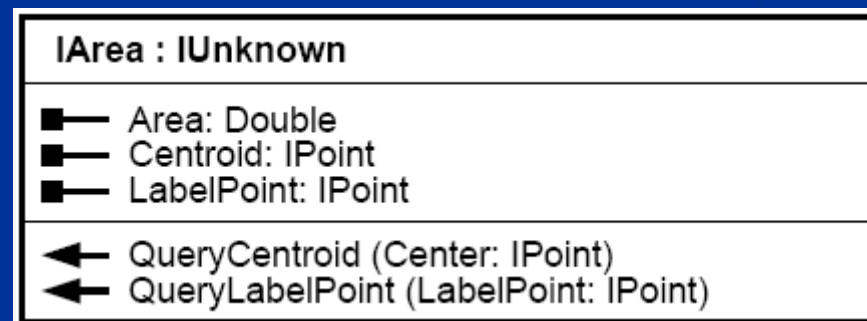


# Area Property

- Many geometries have Area property
  - Polygon, Envelope, Ring
- QI to IArea to get area

IArea poly=( IArea )polygon;

Double area=poly.area ;



# Length Property

- Geometry except point, multiPoints and envelope all have Length property
- ICurve :: Length

```
ILine line=new Line ();
```

```
line.FromPoint=pointA;
```

```
line.Topoint=pointB;
```

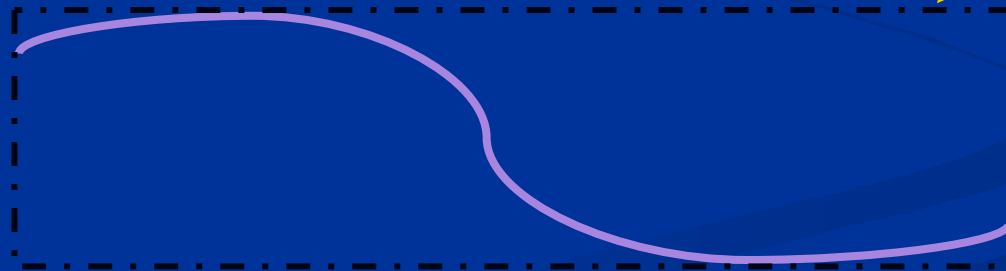
```
Double length=line.Length;
```

```
■ ILine : ICurve
  ■ Angle: Double
  ◀ PutCoords ( from: IPoint, to: IPoint)
  ◀ QueryCoords ( from: IPoint, to: IPoint)
  ■ ICurve : IGeometry
    ■ FromPoint: IPoint
    ■ IsClosed: Boolean
    ■ Length: Double
    ■ ToPoint: IPoint
```

# Envelope

- Define a feature's spatial extent
  - Minimum bounding rectangle
- All geometry has an envelope
  - Get or set with `IGeometry :: Envelope`

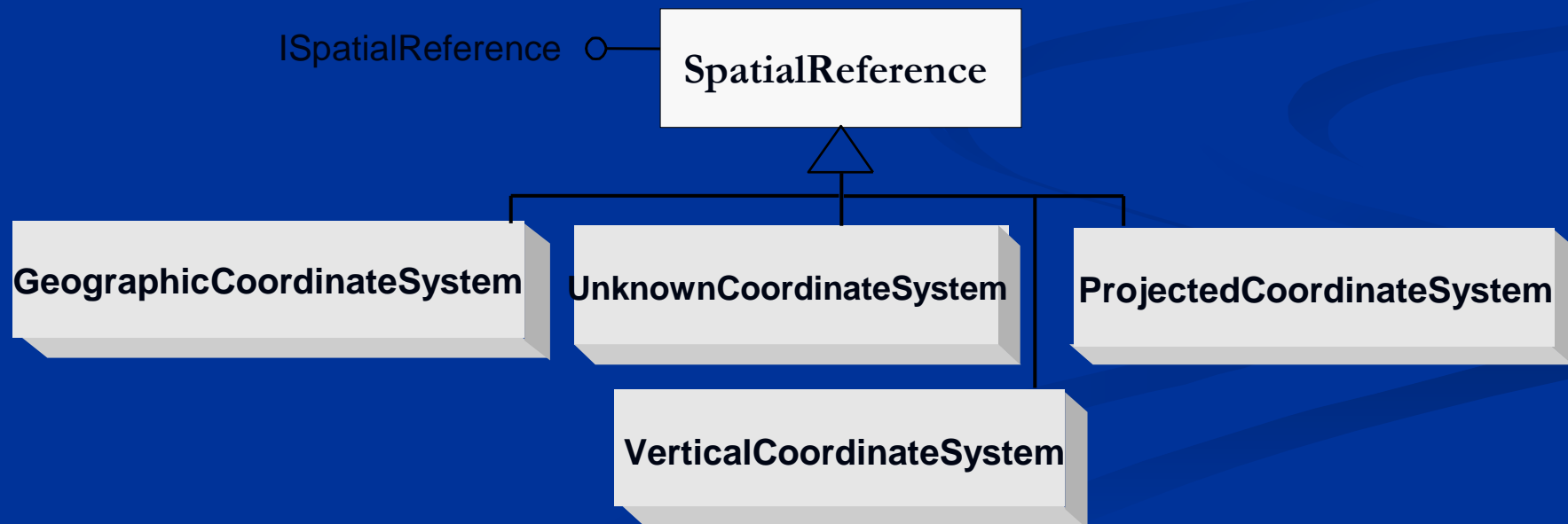
```
Dim pEnvelope As IEnvelope  
Set pEnvelope = pLine.Envelope
```



Envelope

# Spatial Reference

- All geometry has a spatial reference
- Create new coordinate systems
  - ISpatialReferenceFactory contains methods for creation
  - SpatialReferenceEnvironment implement create method

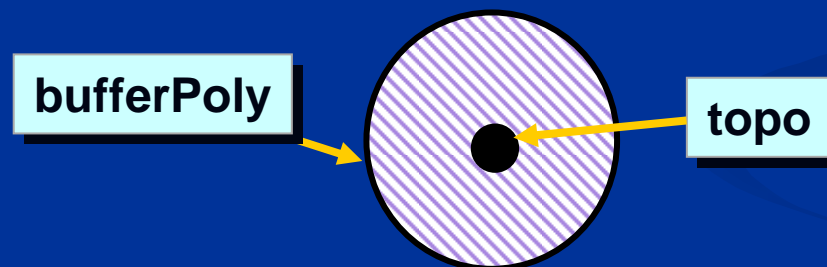


# Spatial Analysis

- Remember operator below are all Geometry
  - ITopologicalOperator
  - IProximityOperator
  - IRelationalOperator
- Use to:
  - Topological perform familiar spatial operations such as **buffer, cut, intersect, simplify, union** and **clip**
  - Proximity measure **distances** between shapes and return the **nearest** point.
  - Relational examine spatial **relationships** such as **within touches, contains, overlaps** and **crosses**.

# ITopologicalOperator

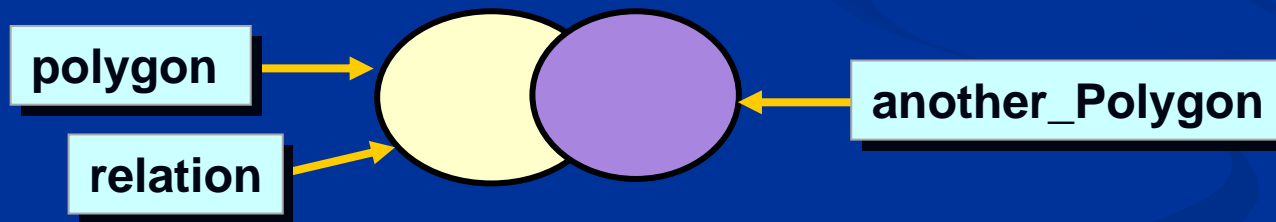
- Provides methods for working with geometry
  - Construct new geometry based on others
  - Perform buffers, intersects, and clips on features
  - Supported by Point, Multipoint, Polyline, and Polygon



```
ITopologicalOperator topo=( ITopologicalOpertor )feature.Shape;  
IPolygon bufferPoly=topo.Buffer ( distance );
```

# IRelationalOperator

- Methods for examining spatial relationships
  - Equals - Are input geometries structurally equivalent?
  - Touches - Do input geometry boundaries intersect?
  - Contains - Is one geometry contained by the other?
  - Disjoint - Are input geometries spatially distinct?
  - All return Boolean

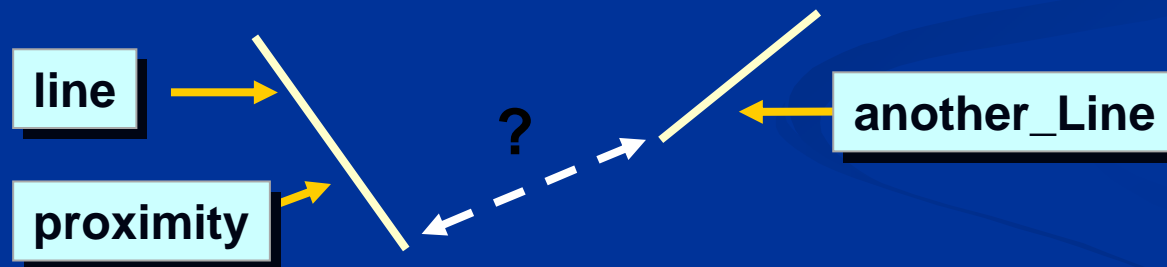


```
IRelationalOperator relation=polygon as IRelationalOperator;
```

```
Boolean isTouched=relation.Touches(another_Polygon)
```

# IProximityOperator

- Examines proximity relationships between features
  - ReturnDistance - Returns the minimum distance between features (return double)
  - ReturnNearestPoint - Finds and returns the nearest point on the specified feature (return point)



```
IProximityOperator proximity=line as IProximityOperator;  
Double distance=proximity.ReturnDistance(another_Line);
```



# Display Transformation

- Transform between map and display units
  - *ToMapPoint*: Convert a display point (pixels) to a map point
  - *FromMapPoint*: Convert a map point to a display point
  - *TransformCoords*: Return map coordinates from a set of display coordinates or vice versa
- Using *IDisplayTransformation* Interface

```
IHookHelper hookHelper=new HookHelper();
```

```
hookHelper.Hook=hook;
```

```
IPoint point;
```

```
point=hookHelper.ActiveView.ScreenDisplay.DisplayTransformation.  
toMapPoint(x , y);
```

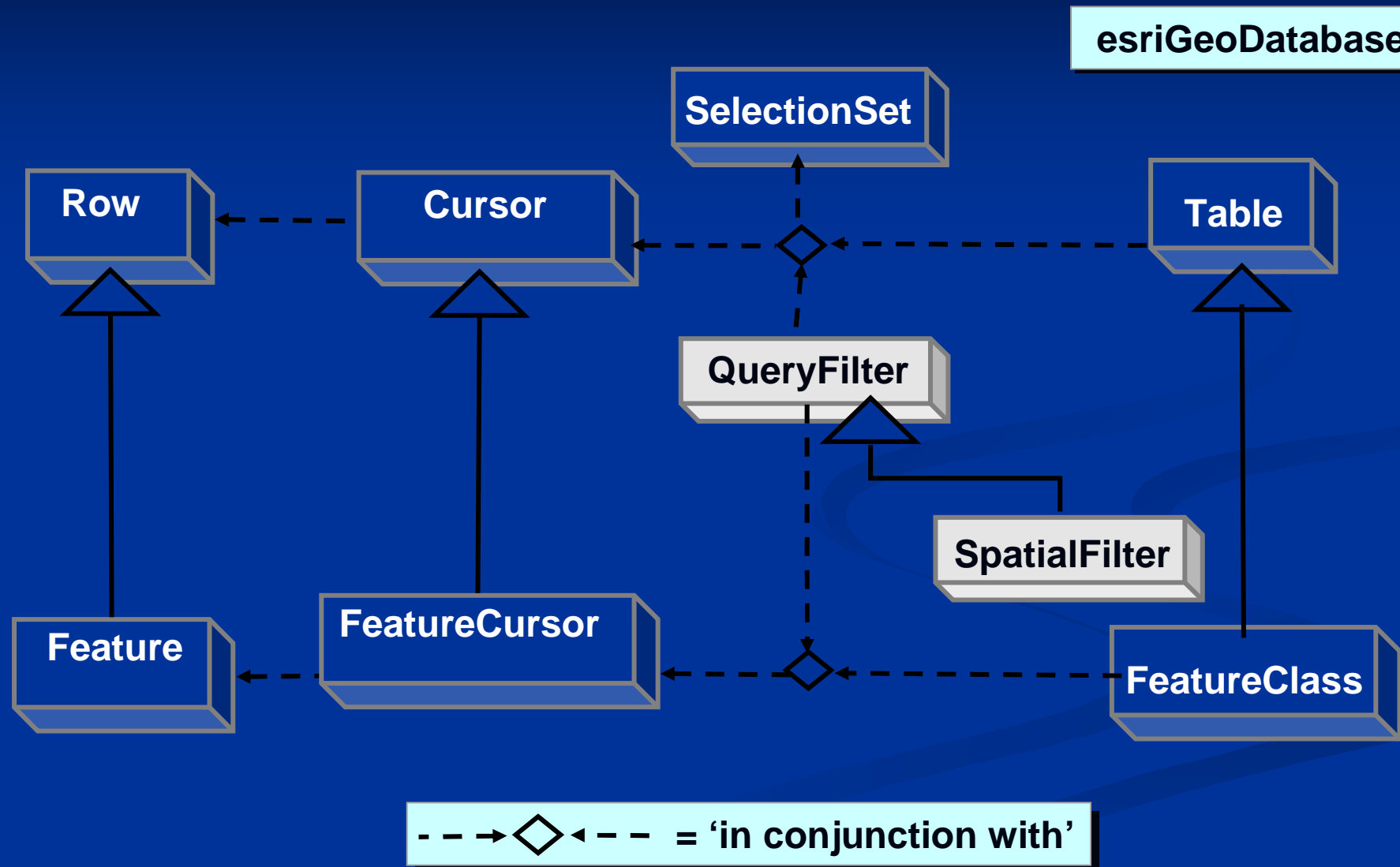
## Exercise 03 : Polyline to Polygon

- Create polyline data
- Using IPointCollection interface
- Convert polyline to polygon
- Add polygon to map

# Presentation Outline

- Introduction of ArcObjects and .Net SDK
- Customize and Extending ArcObjects
- Access Data and Display Layer
- Feature Creation and Editing
- All Below Is Geometry
- ✓ *Query Selection and Analysis*
- Display and Export Map
- Deploying the Application (Engine)
  - Extending 1: ArcGIS Raster APIs
  - Extending 2: Spatial Network APIs

# Object Model Overview



# Cursor and FeatureCursor Again

- Search cursor
  - Search method
  - Use for read-only analysis of a record subset
- Update cursor
  - Update method
  - Use to update or delete records
- Insert cursor
  - Insert method
  - Use to insert new records into the database

# Using Selection

- **IFeatureSelection** performs as a feature layer which controls feature selection.
  - SelectFeatures: performs features selected method
  - SelectionSet: shows selection on a feature layer
- **ISelectionSet** manages a set of selected table rows or features.
  - Search: return cursors based on the selection
  - Select: return a new selection based on the selection

# Creating a Filter

## ■ IQueryFilter interface

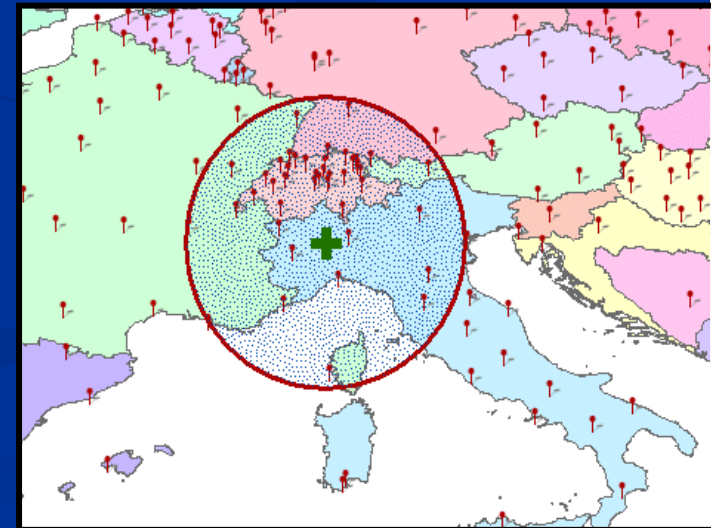
- Set the WhereClause property

```
IQueryFilter filter=new QueryFilter();  
filter.WhereClause="age >30";
```

## ■ ISpatialFilter interface

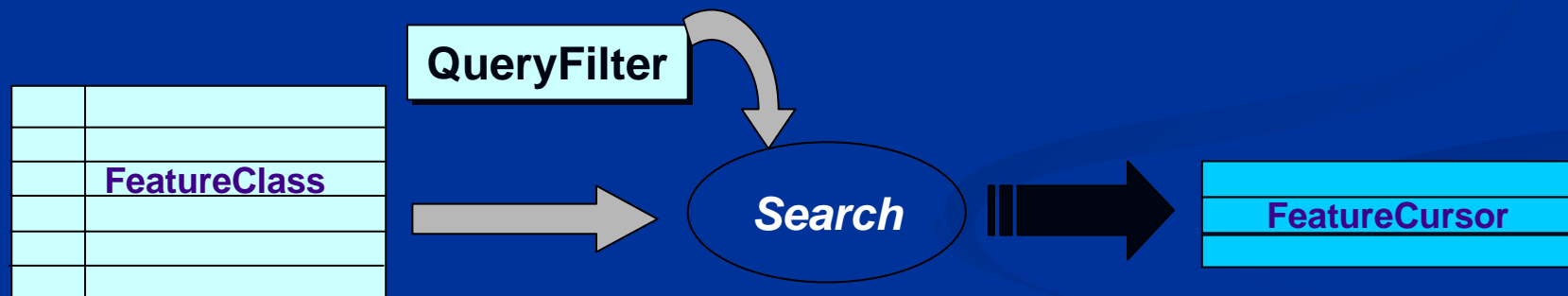
- SpatialFilter is a Type of QueryFilter
- Based on spatial relationship

```
ISpatialFilter filter=new SpatialFilter();  
filter.WhereClause="name='China' ";  
filter.Geometry=point;  
Filter.SpatialRef=esriSpatialRefContains;
```



# Returning a Search Cursor

- Apply to a table or feature class
  - ITable :: Search and IFeatureClass :: Search
  - Returns a cursor or feature cursor



*// false means fCursor recycled until object is destroyed*

```
IFeatureCursor fCursor=fclass.Search(filter, false);
```

```
IFeature feature=fCursor.NextFeature();
```



## Exercise 04: Find Cities Based on Selection

- Select a country in feature layer
- Based on the selection, query the cities which has more than 2,000,000 people

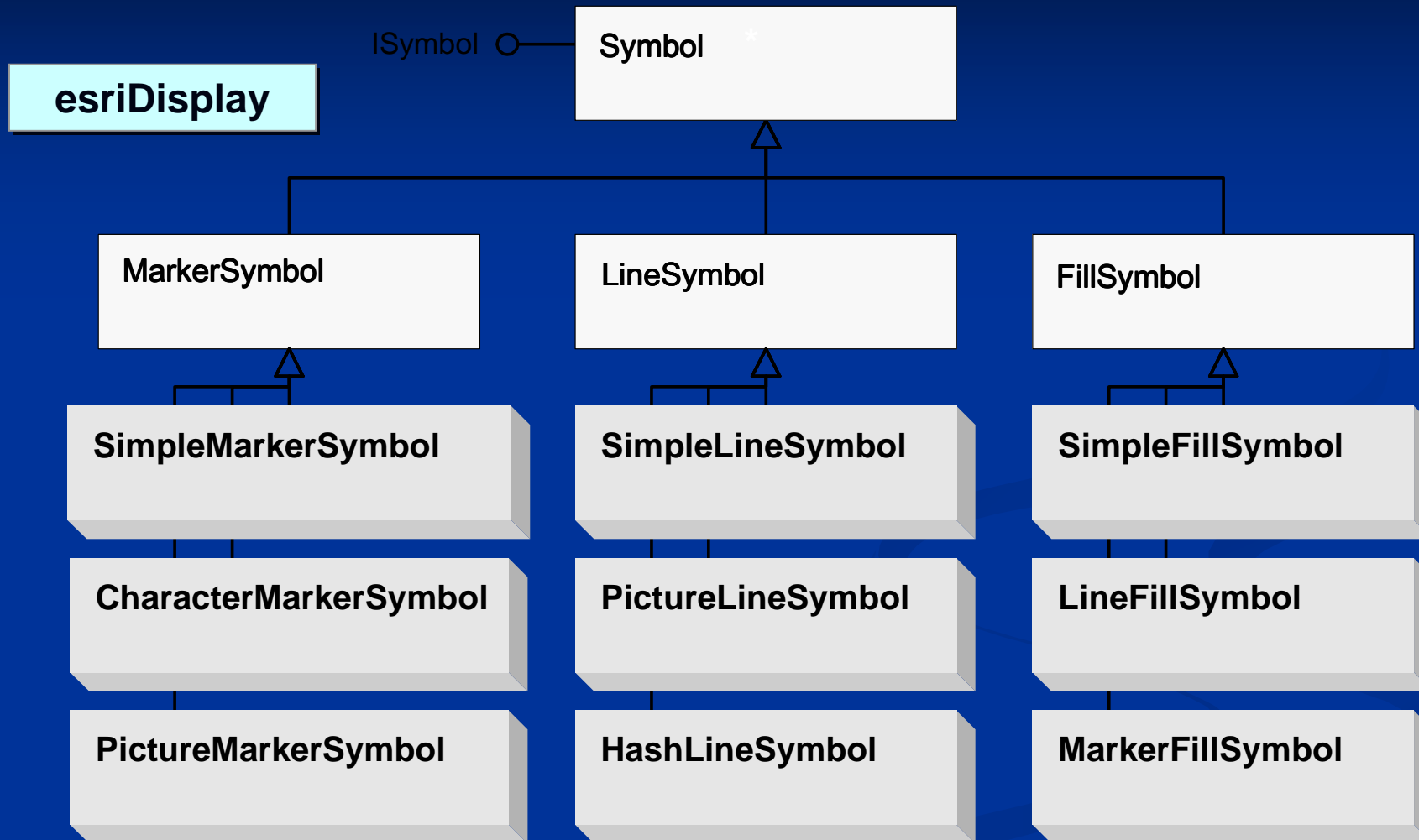
## Exercise 05: Query and Analysis

- Imagine that a fire emergency happened, get the fire point.
- Query and find the parcel where the fire point is.
- Make buffer to the parcel, analysis the area affected in the buffer zone.
- Select other parcels affected in the buffer zone, and add to map.

# Presentation Outline

- Introduction of ArcObjects and .Net SDK
- Customize and Extending ArcObjects
- Access Data and Display Layer
- Feature Creation and Editing
- All Below Is Geometry
- Query Selection and Analysis
- ✓ **Display and Export Map**
- Deploying the Application (Engine)
  - Extending 1: ArcGIS Raster APIs
  - Extending 2: Spatial Network APIs

# ISymbol Interface

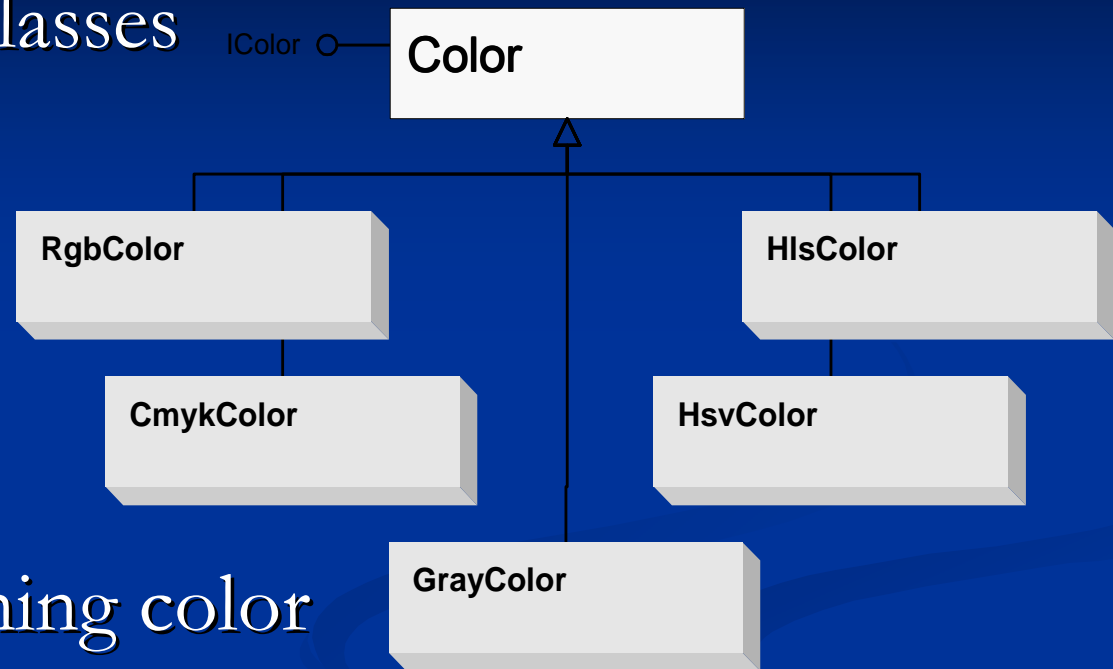


\* Several additional types of symbols are listed on the esriDisplay OMD

# Using Color

## ■ Five creatable subclasses

- RgbColor
- CmykColor
- GrayColor
- HsvColor
- HlsColor



## ■ Properties for defining color

- Red, Green, Blue values (0–255)
- Grayscale (0-255) white to black
- Cyan, Magenta, Yellow, Black

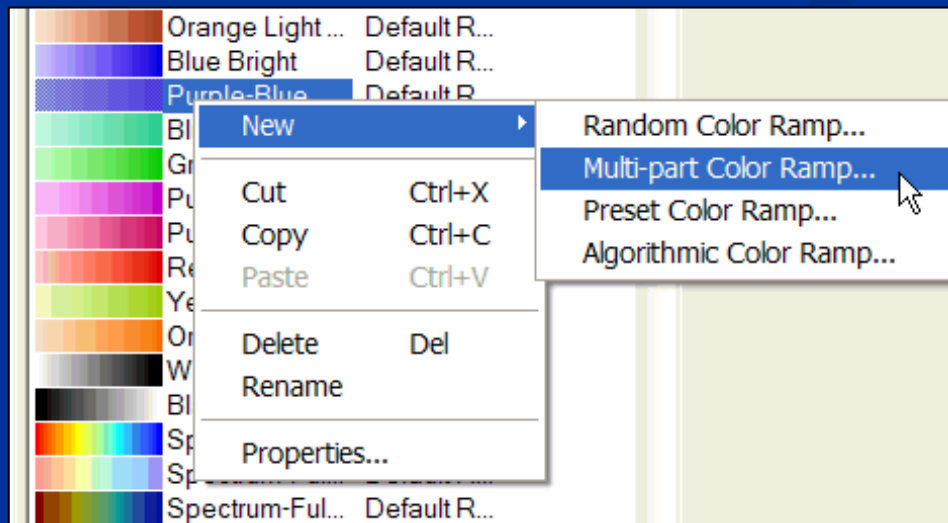
## ■ Using color to assign a symbol's color property

# ColorRamps

## ■ Four subclasses

- AlgorithmicColorRamp: using begin-color and end-color to define
- RandomColorRamp: using common HSV to define colors
- PresetColorRamp: using pre-defined colors to define colors
- MultiPartColorRamp: multi-using above ramps to define colors

## ■ Referenced from the StyleManager



# Creating Elements

- Element is mainly consisted of two types: GraphicElement and FrameElement
  - GraphicElement: **TextElement**, **MarkerElement**, **LineElement**, **PolygonElement**
  - FrameElements: **MapSurround** ( e.g. **scaleBar**, **NorthArrow**, **Legend**) and **PictureElement**
- Using **IGraphicsContainer** interface to add elements
  - **GraphicsContainer** is implemented by **Map** or **PageLayout**
  - Using **AddElement** method
- Element has **Geometry** property, which specify with point, line or polygon.
- Element has **Symbol** property, which set symbol to element.

# Sample: Create Element and Assign Symbol

```
IMarkerElement maker = new MarkerElement();
```

```
ISimpleMarkerSymbol sym=new SimpleMarkerSymbol();
```

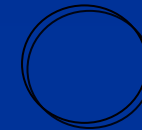
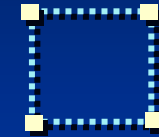
```
sym.Style=esriSimpleMarkerStyle.esriSMSCircle;
```

```
IRgbColor color =new RgbColor();
```

```
color.RGB=RGB (255,255, 0);
```

```
sym.Color = color;
```

```
marker.Symbol = sym
```





# Sample: Define a Position and Add to Map

```
// QI marker to element
```

```
IElement element = marker as IElement;
```

```
IPoint point = new Point ();
```

```
point.PutCoords ( 45, 100 );
```

```
element.Geometry = point;
```

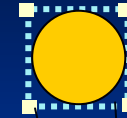
```
// QI map to graphics container
```

```
IGraphicsContainer gc =( IGraphicsContainer )axMapControl1.Map;
```

```
// the second parameter is z-order, 0 means the top layer
```

```
gc.AddElement ( element , 0 );
```

```
axMapControl1.ActiveView.Refresh();
```

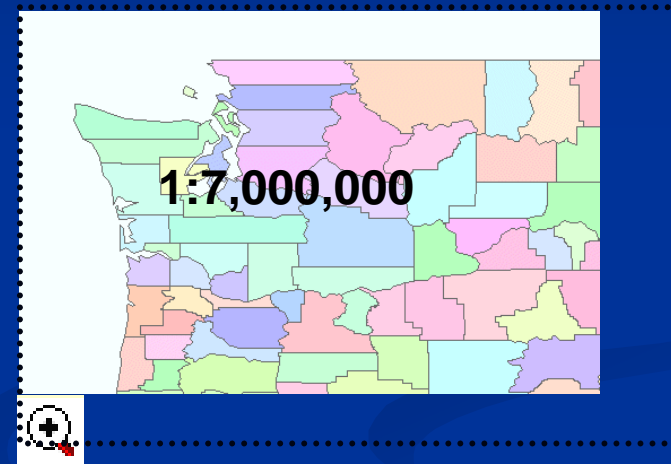
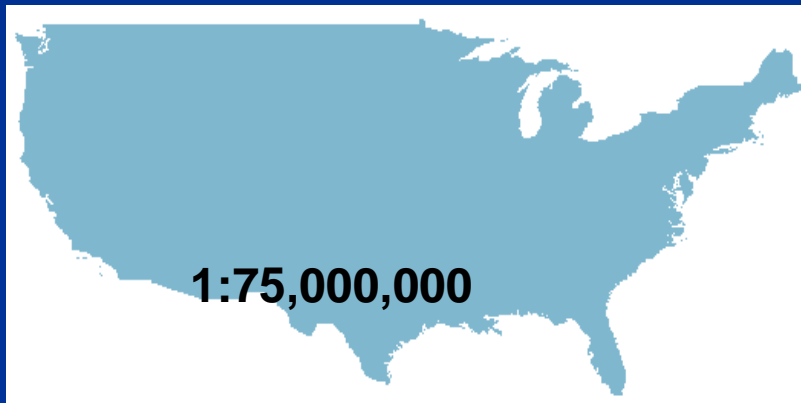


# FeatureRenderers

- Renderers define how a layer is displayed
- Six types of Feature Renderer
  - SimpleRenderer: Simple Render
  - UniqueValueRenderer: Based on Unique Value
  - ClassBreaksRenderer: Divide into Classes to Render
  - ChartRender: Chart Render
  - DotDensityRenderer: Dot Density Render
  - ScaleDependentRenderer: Based on Scales Dependent

# ScaleDependentRenderer

- Render based on Scale Dependent
  - More detail at large scales, less detail at small scales



- Properties
  - Break: cut-off points (scale ranges)
  - Renderer: renderers in the ScaleDependentRenderer
  - RendererCount: number of renderers contained

# IExport Interface

- Set export filter and pixel bound properties.
- Attention output method should within StartExporting and FinishExporting.
- Using IActiveView : Output method to export a bitmap.
  - hdc equals to value of StartExporting
  - dpi means resolution
  - PixelBounds means the size of output bitmap. If 600 \* 800 wanted, then PixelBounds is { 0, 0, 600, 800 }
  - VisibleBounds means the size in original map. That is top-left corner and bottom-right corner in map { 0, 0, 45, -45 }

# Sample: Export a fix location JPG

```
IExport export = new ExportJPEGClass(); // JPEG Export
export.Resolution = 96; // DPI, Resolution
ESRI.ArcGIS.Display.tagRECT exportRECT = new tagRECT();
// PixelBounds 512 * 512
IEnvelope envelope = new EnvelopeClass();
envelope.PutCoords(0,0, 512,512);
export.PixelBounds = envelope;
// Export Scale Lon: 0 - 45 and Lat: 0 - 45
IEnvelope envelope2 = new EnvelopeClass();
envelope2.PutCoords(0, 0, 45, 45);
// Start Exporting
Int32 hDC = export.StartExporting();
activeView.Output(hDC, (System.Int16)export.Resolution, ref exportRECT,
envelope2, null); // Export Data
export.ExportFileName = "actc.jpg";
export.FinishExporting(); // Finish Exporting
export.Cleanup(); // Release and Clean
```

# Exercise 06: Complete Overview Function

- Associate two MapControl
- Code into axMapControl1\_OnExtentUpdated and axMapControl2\_onMouseDown Event.
- Set overview symbol display

# Presentation Outline

- Introduction of ArcObjects and .Net SDK
- Customize and Extending ArcObjects
- Access Data and Display Layer
- Feature Creation and Editing
- All Below Is Geometry
- Query Selection and Analysis
- Display and Export Map
- ✓ **Deploying the Application (Engine)**
  - Extending 1: ArcGIS Raster APIs
  - Extending 2: Spatial Network APIs
  - Extending 3: ArcGIS Server Flex APIs

# General Deployment Process

- Develop application to deploy
  - Executable or custom component
- Create a setup package
  - .NET Setup project, batch file, etc.
  - Verify installation prerequisites
  - Ensure installation and authorization of ArcGIS Engine Runtime
- Test deployment
  - Test under variety of target conditions



# What About ArcGIS Engine Runtime

- Shared Core ArcObjects Library functions
  - ArcGIS Engine: Libraries and components
  - ArcGIS Engine Java Runtime: Java archives
  - ArcGIS Engine .NET Runtime: .NET Assemblies
  - Python 2.6 (ArcGIS 10 )
  - *Must install to run ArcGIS Engine applications*
- Some applications may require extensions
  - e.g., Geodatabase Update
  - Approximately 300 MB, and 750 MB completed installed
- Cannot install if previous versions are present

# Visual Studio Setup Process

- ① Add to solution containing project, and create a new setup project to solution
- ② Define what will be installed
  - ① Project output
  - ② Dependent assemblies
  - ③ Supporting files (data, images, fonts, etc.)
- ③ Determine the main output
- ④ Exclude dependent assemblies except .Net Framework
- ⑤ Add short-cut to the deployment
- ⑥ Generate the project (press F6)

Add `msiexec.exe` short-cut, and assign the solution's product-code, if want to uninstall.

# Presentation Outline

- Introduction of ArcObjects and .Net SDK
- Customize and Extending ArcObjects
- Access Data and Display Layer
- Feature Creation and Editing
- All Below Is Geometry
- Query Selection and Analysis
- Display and Export Map
- Deploying the Application (Engine)
  - ✓ Extending 1: ArcGIS Raster APIs
  - Extending 2: Spatial Network APIs

# RasterWorkspace

- Based on Files (e.g. .tif, .jpg, .bmp and so on)
  - using [RasterWorkspaceFactory.openFromFile](#) method
  - using [RasterWorkspaceFactory.open](#) method to access server or ArcSDE files.
- Based on GeoDatabase (e.g. such files in a GDB)
  - using [FileGDBWorkspaceFactory.openFromFile](#) method
  - using [FileGDBWorkspaceFactory.open method](#) to access server or ArcSDE geodatabase.

# RasterDataset Object

- The **RasterDataset** object represents a dataset on disk or in a geodatabase. It is composed of one or more persistent raster bands.
- Using RasterWorkspace.CreateRasterDataset method to create a new one.

// last parameter means whether raster is on disk or in memory

```
rasterWs.CreateRasterDataset(name, "TIFF", origin, width, height, xCell,  
yCell, NumBand, rstPixelType.PT_UCHAR, spatialReference, true)
```

- Using RasterWorkspace.OpenRasterDataset method to open a existed one.

```
rasterWs.CreateRasterDataset("china .tif")
```

# Raster Object

- The Raster object is a transient representation of raster data that performs resampling and reprojection.
- Raster is always transient. When modified Raster using ISaveAs or IRasterBandCollection interface to save.
- Raster is associated with one or more raster bands, which provide a source for data to be read through the raster.
- Types of get raster:
  - RasterLayer's Raster property
  - IRasterDataset CreateFullRaster or CreateDefaultRaster method
  - Create a new raster, and add to source by IRasterBandCollection :: AppendBand method

# RasterBand and RasterBandCollection

- The RasterBand object represents an existing band of a raster dataset.
- The RasterBand object represents an existing band of a raster dataset.
- RasterBandCollection can be get from Raster or RasterDataset object.

# PixelBlock Object

- PixelBlock contains a pixel array that can be read from a raster or a raster band.
- PixelBlock can create from both Raster and RasterBand.
- PixelBlock can be created in any size, but cannot be changed after created. PixelBlock can be the size of the entire dataset for small raster.
- Using IRasterEdit :: Write method to modify the pixel values, and write pixel values to a raster band.
  - PixelData method to read and write pixel values
  - IRasterEdit :: Write method allow to modify the values. Another way to modify values can use IRawPixels interface.



# RasterProps, RasterInfo and RasterCursor

- **RasterProps** object can set **width, height, nodata and spatialreference**. RasterProps object can be get from Raster or RasterDataset object.
- **RasterInfo** object can set **blockWidth, blockHeight, cellSize, statistics and spatialreference**.
- **RasterCursor** object controls in a Raster. **enumeration through the PixelBlocks**. Using `IRaster::CreateCursor` or `IRaster2::CreateCursorEx` method to create RasterCursor.

# PixelFilter Object

- IPixelFilter defines a simple pixel block filtering operation, which filters (changes) pixel values in place, and does not change pixel type.
- Custom pixel filter would implement IPixelFilter.  
PansharpeningFilter (Mean) and ConvolutionFilter (e.g. 3\*3 or 5\*5, low pass and high pass), And Apply to Raster to using IPixelOperation:PixelFilter

```
// Through PixelFilter properties below to operate pixel value  
IPixelOperation pixelOp = ( IPixelOperation )raster;  
pixelOp.PixelFilter = ( IPixelFilter )filter;
```

# RasterGeometryProc Object

- The **RasterGeometryProc** object performs geometric processing, such as flipping, scaling, rotation, clip, merge rectify (Format rectify), resample and mirror.

*// Rotate the raster into 45 angles, and register to save*

```
IRasterGeometryProc rasterGP = new RasterGeometryProc ();  
rasterGP.Rotate(null, 45, raster);  
rasterGP.register(raster)
```

# Others about Raster

- **SaveAs** Object saves a raster or raster dataset to one of the supported raster formats.

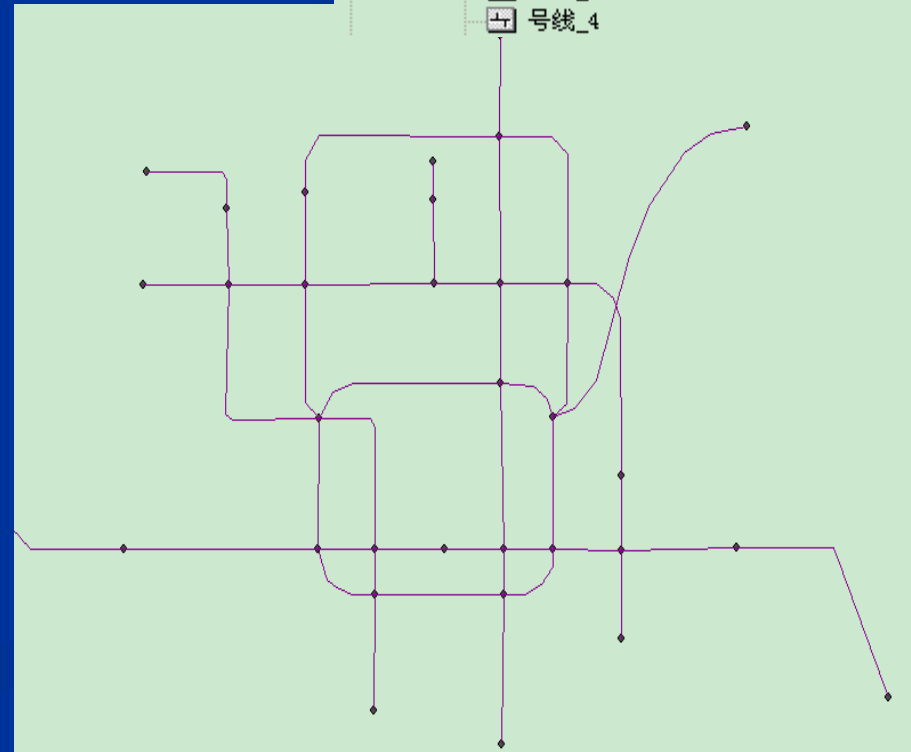
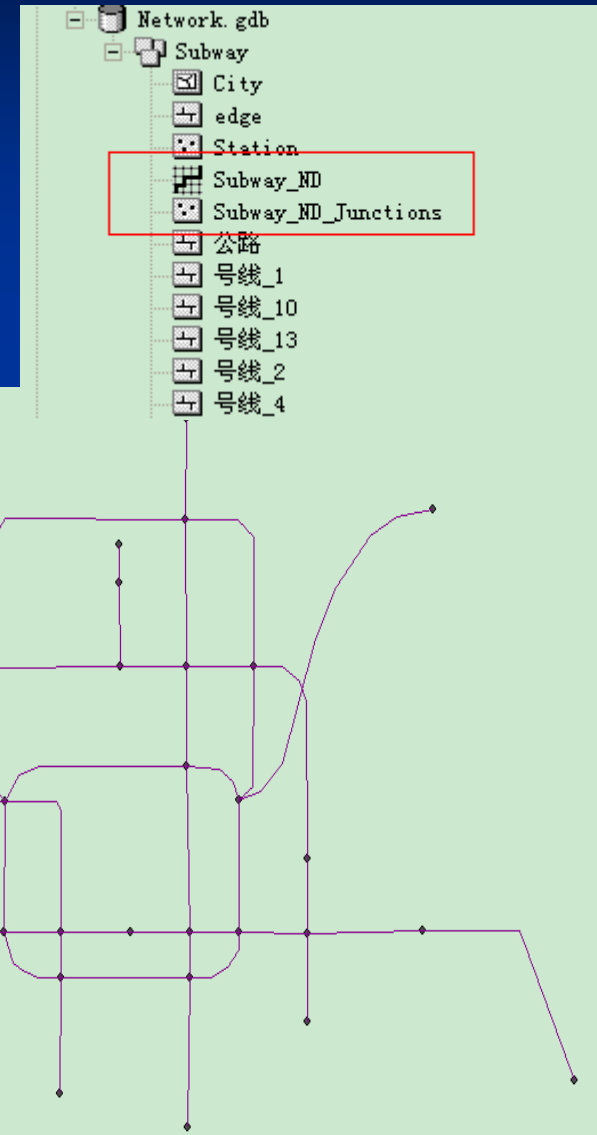
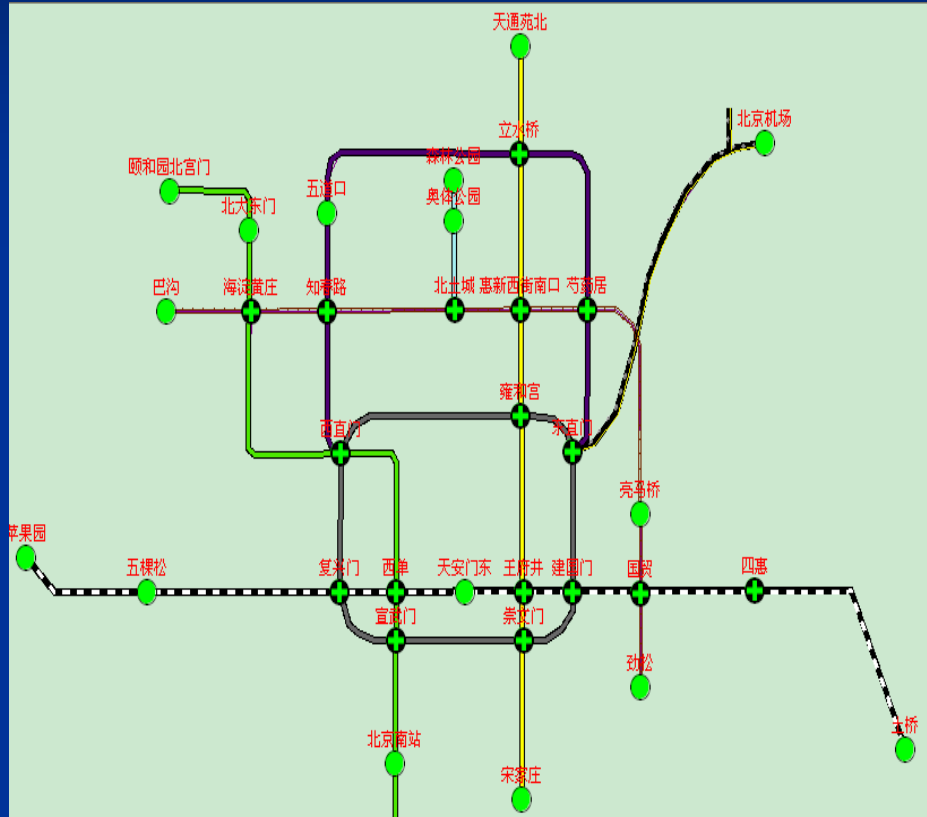
```
saveAs.SaveAs("raster.tif", ws, "tiff"); // "tiff" is a format
```

- **PixelOperation** Object use Filter property to operate pixel values.
- **RasterPyramid** Object controls pyramid of raster dataset through PyramidLevel, MinimumSize, Create and BuildPyramid members.
- **ConversionOp** Object controls conversion between features and raster, which is in GeoAnalysis Library.

# Presentation Outline

- Introduction of ArcObjects and .Net SDK
- Customize and Extending ArcObjects
- Access Data and Display Layer
- Feature Creation and Editing
- All Below Is Geometry
- Query Selection and Analysis
- Display and Export Map
- Deploying the Application (Engine)
  - Extending 1: ArcGIS Raster APIs
  - ✓ Extending 2: Spatial Network APIs

# Create New Network Dataset



# Access Network Dataset

- IFeatureDatasetExtensionContainer
  - Access Extension Dataset Container
  - `fws.OpenFeatureDataset(featureDatasetName)`
  - FindExtension method to access IDatasetContainer
- NetworkDataset Object
  - By `IDatasetContainer.get_DatasetbyName` method

# NAContext Object

- NAContext is the key Object in analysis.
  - Obtain references of NetworkDataset, NAsolver, NAClasses, NALocator
  - Create by INASolver.CreateContext method
- INAContextEdit interface
  - Access editable properties of NAContext
  - Bind method prepares the context for analysis based on Network Dataset Schema.



# INASolver Interface

- INASolver interface is common for all solvers.
  - NAClosestFacilitySolver
  - NARouteSolver
  - NAServiceAreaSolver
- Two important methods
  - CreateContext method to create NAContext
  - Solve method returns boolean to perform analysis
    - False means completed solved
    - True means partial solved

# NAClass Object

- Holds features input or output during the analysis.
- Different solver, different NAClass name
  - If NARouteSolver: Stops, Routes and Barriers.
  - If NAClosestFacilitySolver: Facilities, Incidents, CFRoutes and Barriers.
  - Attention: these words can't be changed!

# INAClassLoader Interface

- The Locator property specifies how network locations will be found.
- Load method returns features based on FeatureCursor and located locations

# Add Analysis Result to MapControl

- Use ISolver.Solve method to perform analysis
- In NAClass Routes or CFRoutes represent the solved result.
- QI NAClass to FeatureClass
- Add FeatureClass to MapControl

# Analysis Result

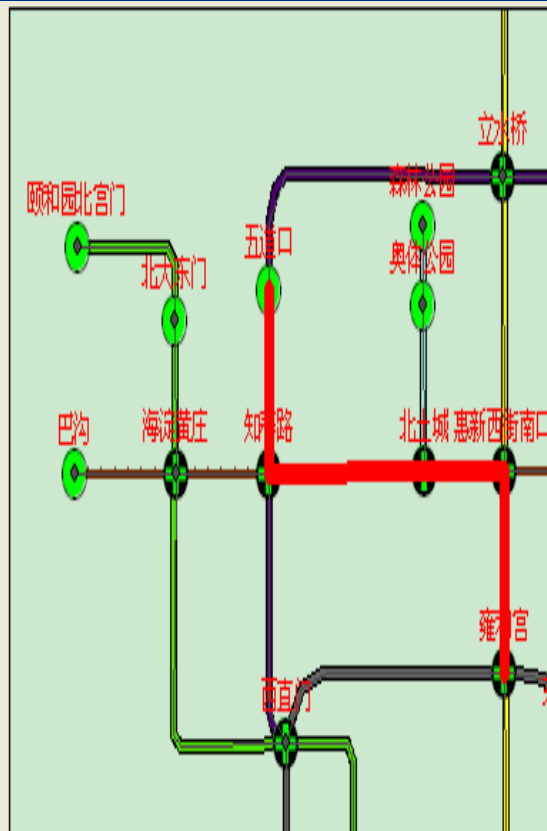
请选择起始站

请选择终点站

约束条件

查询

起始站: 五道口 至终点站: 雍和宫



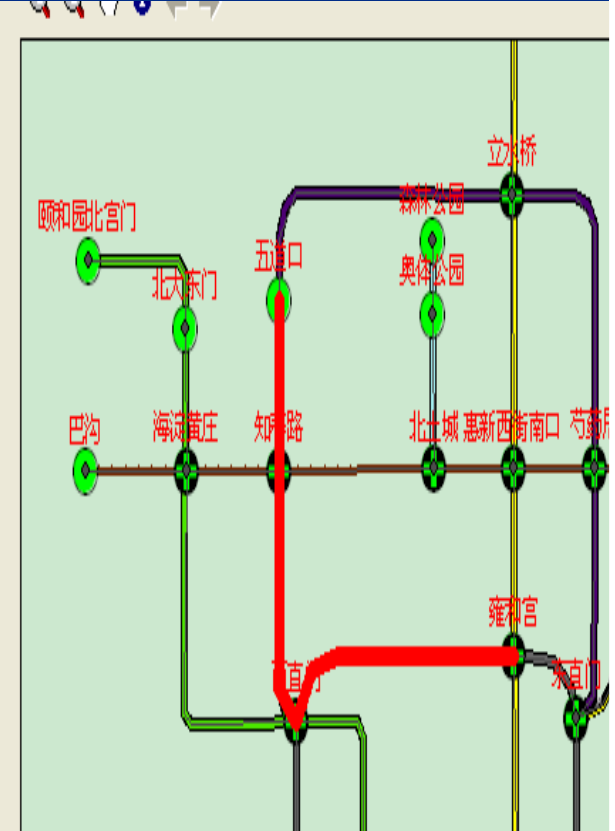
请选择起始站

请选择终点站

约束条件

查询

起始站: 五道口 至终点站: 雍和宫  
地铁线路:  
13号线至 西直门 换乘: 2号线



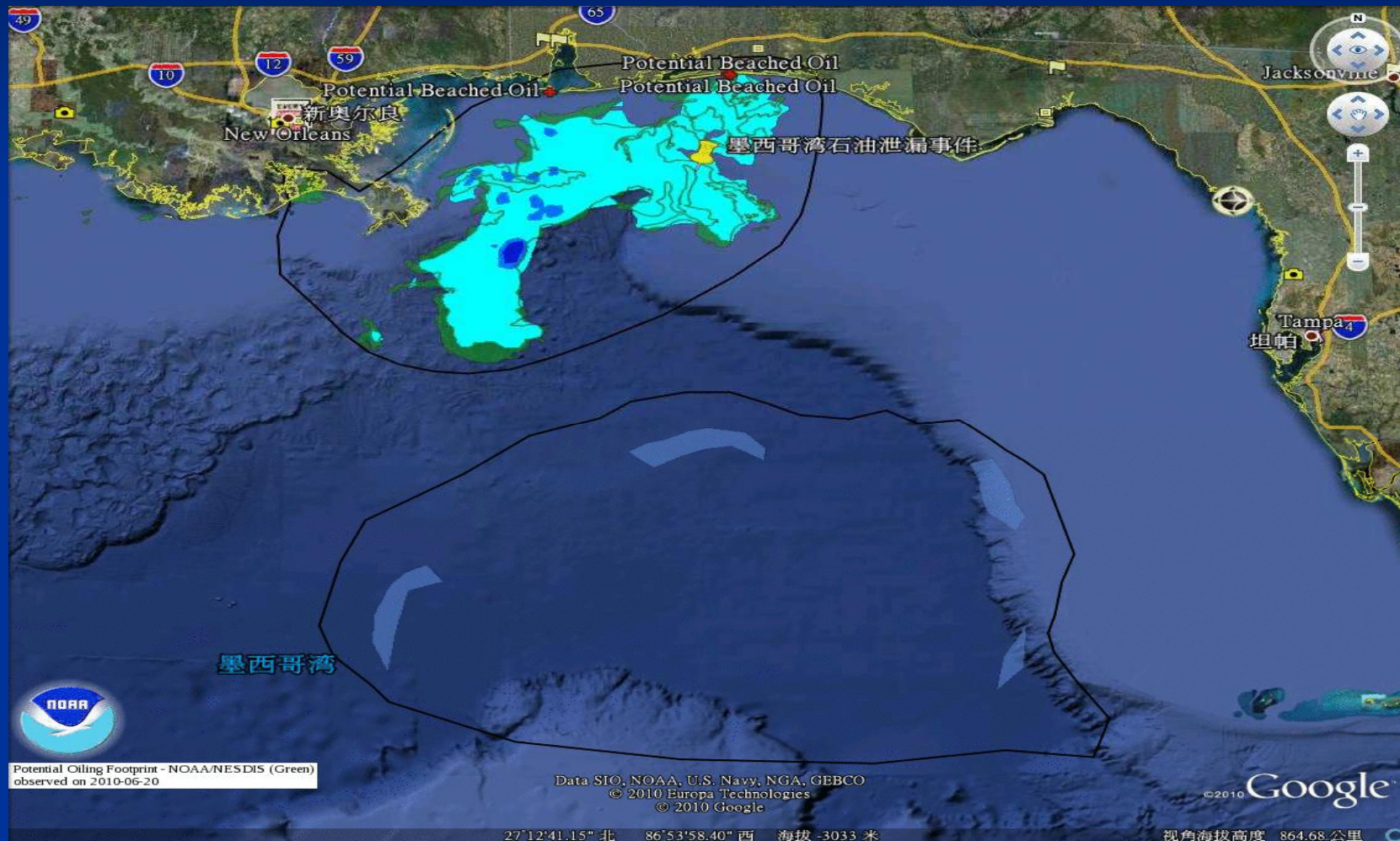
# Developer Topics

- About Future
- About Developer Career

# ArcGIS Server



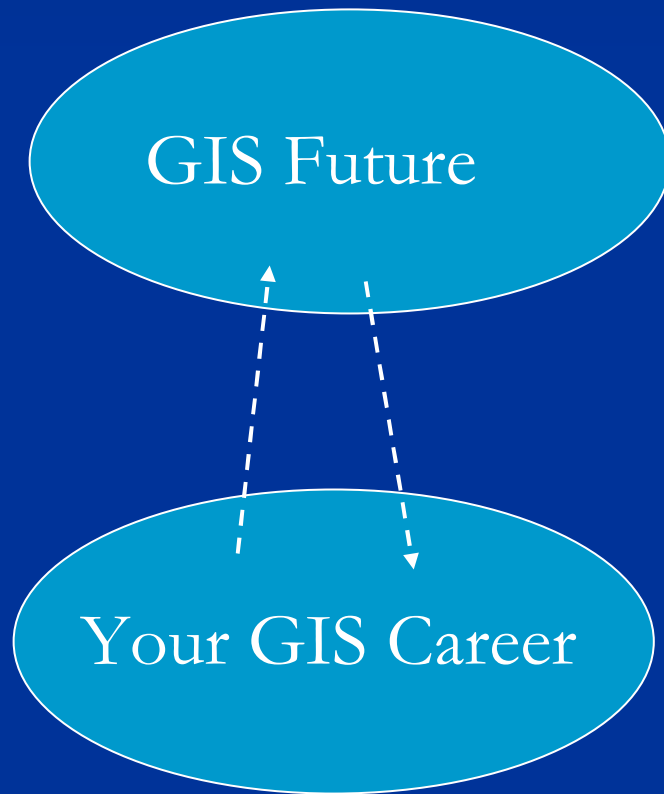
# Dynamic Tracing





# Developer Topics

- Think about questions :



**Thank you for your listening!**

**[zyk13032@gmail.com](mailto:zyk13032@gmail.com)**