地理信息系统应用程序设计与开发

第3章 ArcEngine简介及C#语言

主讲：张宝一
Email：zhangbaoyi.csu@qq.com

教学目标

- 熟悉组件及相关概念
- 熟悉ArcEngine的组成
- 掌握C#开发ArcEngine的环境部署及方法
- 熟悉C#语言

教学重点

- ArcEngine组成
- C#开发ArcEngine技术

教学内容

- 3.1 组件的基本概念
- 3.2 ArcEngine简介
- 3.3 C#语言简介
- 3.4 C#开发AE示例

3.1 组件的概念

- （1）组件技术的发展
  - 软件重用和开发效率是软件开发的一个核心问题
  - 早期，开发人员将一些基本的数学计算等简单功能设计成函数库（类库），通过应用程序接口让其他软件开发人员调用，特点：
    - 粒度太小
    - 功能单一
    - 程序组织复杂
  - 90年代，开发界提出功能更大、粒度更大的复用构件，其能够对外提供标准接口，内部自动工作。这就是组件技术的原始想法。

3.1 组件概念

（1）组件

- 组件是可重用、具有一定功能、被封装的模块，由一组处理过程、数据封装和用户接口组成的业务对象（Rules Object），可通过接口访问。
- 组件之间能够快速组织并建成一个系统。
- 组件技术的目的
  - 降低开发的周期和成本
  - 提高软件重用和共享效率
  - 使软件更具开放性、灵活性
  - 使应用程序更易于定制、更灵活

3.1 组件的概念

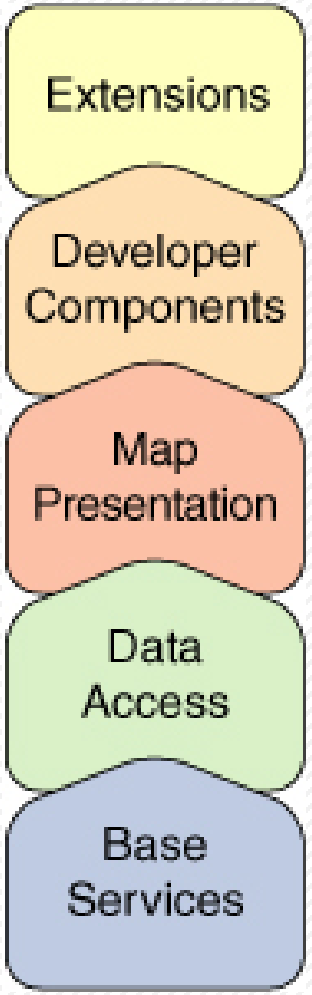
（1）组件

- 组件与类的关系
  - 组件可以在另一个称为容器（有时也称为承载者或宿主）的应用程序中使用，也可以作为独立过程使用；
  - 组件可以由一个类构成，也可以由多个类组成，或者是一个完整的应用程序；
  - 组件为模块重用，而对象为代码重用。

- （2）ComGIS
  - 组件式GIS的基本思想是把GIS的各大功能模块划分为几个组件，每个组件完成不同的功能。
  - 各个GIS组件之间，以及GIS组件与其他非GIS组件之间，可以方便地通过可视化的软件工具集成起来，形成最终的GIS应用。
  - 组件式GIS 开发平台通常可设计为三级结构
    - 基础组件
    - 高级通用组件
    - 行业性组件
  - GIS组件以数据访问、查询、制图、图层控制、地图缩放等为其主要功能

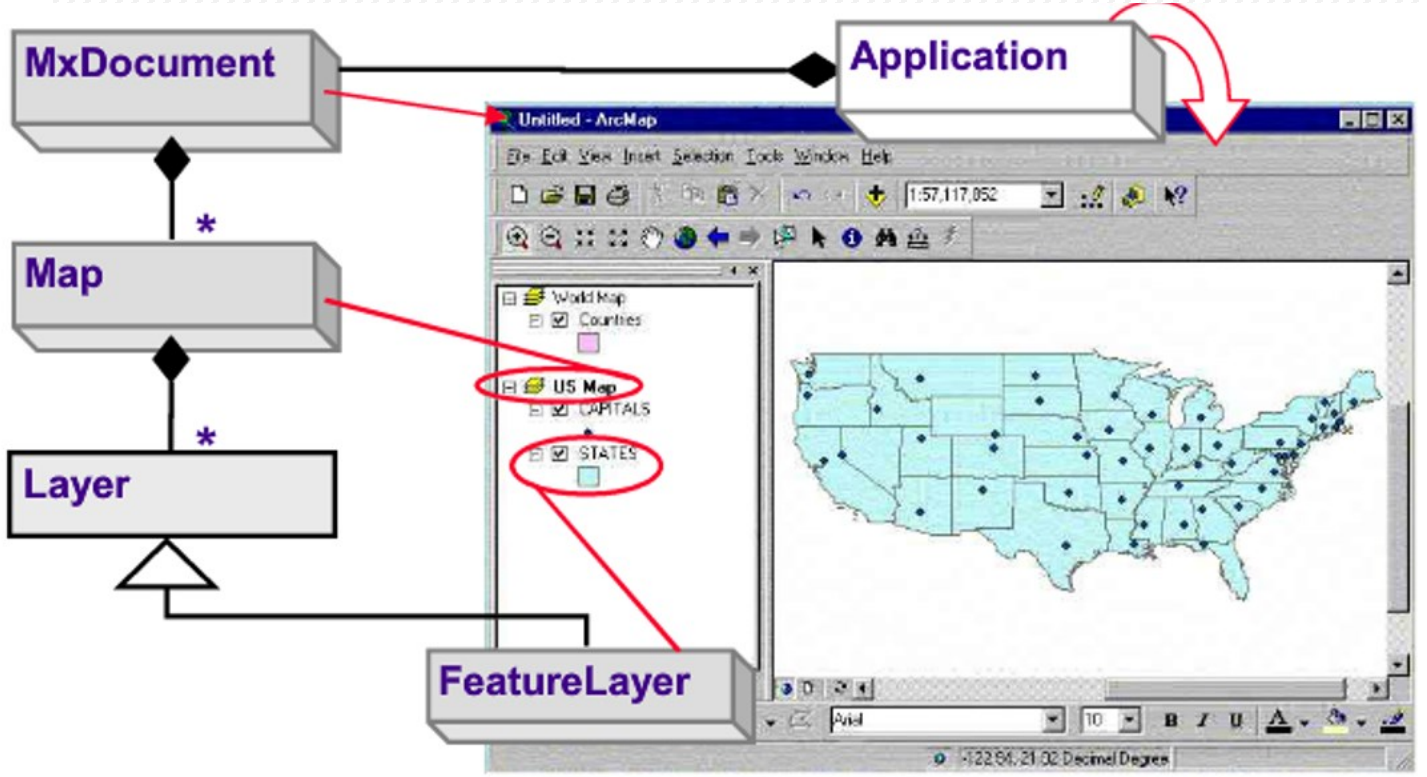
(1)ArcEngine组成

- 基础服务：几乎所有GIS应用都会用到核心GIS对象，比如geometry、display。
- 数据访问：对多种矢量和栅格数据格式和数据库的访问。
- 地图展现：制图和显示，包括符号、标注、专题图等。
- 开发者组件：高层用户RAD(快速应用开发)组件。
- 扩展模块：开发高级GIS功能。



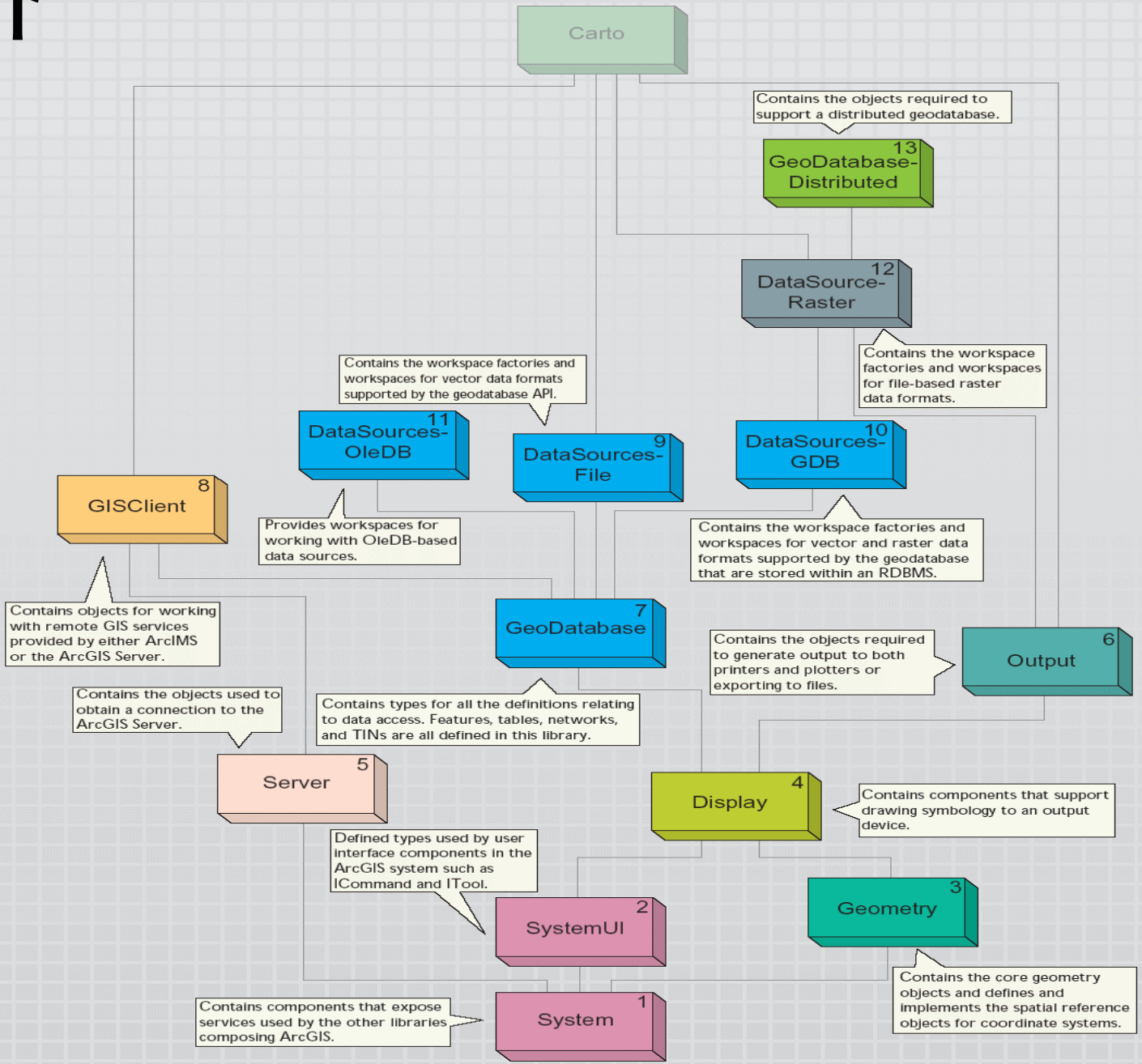
(2)ArcEngine SDK控件

- MapControl
- PageLayoutControl
- SceneControl
- GlobeControl
- ToolbarControl
- TOCControl
- LicenseControl
- SymbologyControl

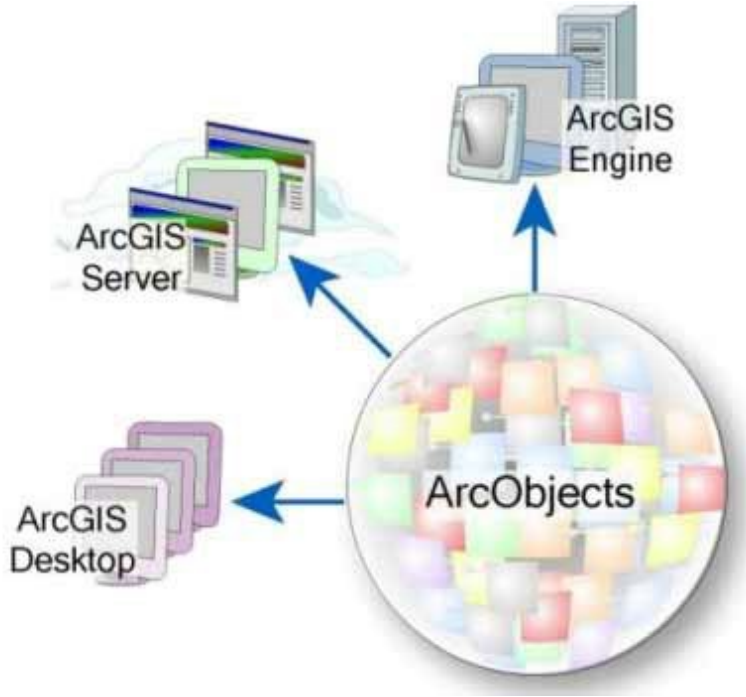


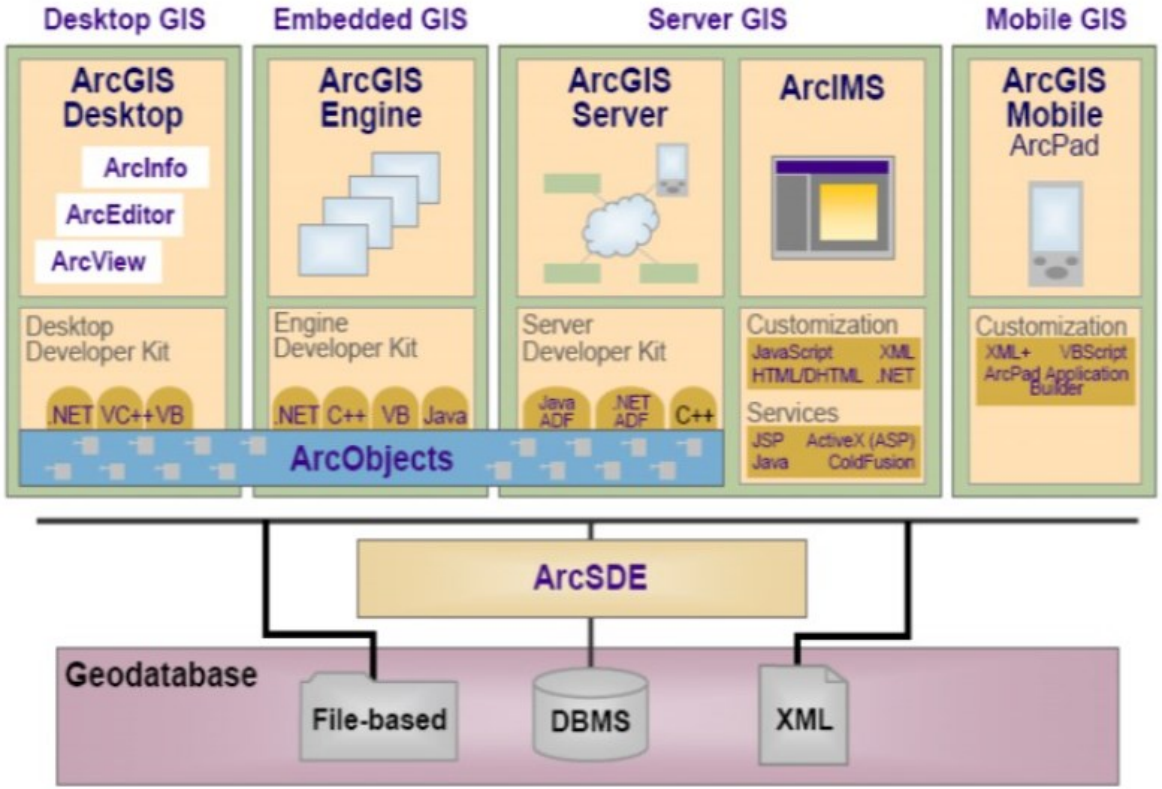
3.2 ArcEngine简介

- ArcEngine库结构



(2)ArcEngine与ArcObjects





(1)C#示例

```
using System;                                          //导入System命名空间
namespace HelloWorld                                   //声明命名空间
{
   class Program                                       //声明类
   {
      static void Main()                               //程序主入口
      {
                Console.WriteLine("Hello World");      //控制台输出
      }
   }
}
```

- C#语言使用C语法，标识命名建议使用大驼峰法

(2)C#变量类型

- 数值型：Byte, Int16(或short),Int32(或int),Int64(或long),Single(或float), Double(或double), Decimal
- 字符型：Char
- 字符串：String
- 逻辑型：Boolean
- 示例：
  - int i =100;Int64 a=100L;float f=100.0f; Double d = 100.0;
  - String str = “I love China”;

- (3)字符串操作
  - 抽取和定位子串

```
string poem = “In Xanadu did Kubla Khan”;
string poemSeg = poem.Substring(10);
poemSeg = poem.Substring(0,9);
int index = poem.IndexOf( “I” );
index = poem.LastIndexOf( “n” );
```

  - 比较字符串

```
bool isMatch;
string title = "Ancient Mariner";
isMatch = (title == "ANCIENT AMRINER");
isMatch = (title.ToUpper() == "ANCIENT MARINER");
isMatch = title.Equals("Ancient Mariner");
```

(3)字符串常用函数

- Equals() ： 比较两个字符串的值是否相等
- ToLower() ： 将字符串转换成小写形式
- IndexOf() ： 查找某个字符在字符串中的位置
- SubString() ： 从字符串中截取子字符串
- Join() ： 连接字符串
- Split() ： 分割字符串
- Trim() ： 去掉字符串两边的空格

(4)数组

- 数组定义
  - `int[] array = new int[6];`
  - `int[,] array = new int[6,6];`
- 数组初始化
  - string[] week = { "Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun" };
- 数组访问

```
for(int i=0;i<6;i++)
    array[i]=i;
```

(5)List：存储泛类型列表

- 定义
  - **`List<类型> list = new List<类型>();`**
- 示例

```
List<int> list = new List<int>();
list.Add(100);
list.Add(200);
for (int i = 0; i < list.Count; i++)
    Console.WriteLine(list[i]);
```

- (6)运算符
  - 算术运算符 ：+, -, *, /, %, ++, --,等
  - 三元运算符 ：表达式1? 表达式2，表达式3
  - 比较运算符 ：>, <, >=, <=, ==, !=
  - 逻辑运算符 ：&&, ||, !
  - 赋值运算符 ：=, +=, -=, *=, /=, %=

(6)运算符 优先级

优先级	说明	运算符	结合性
1	括号	( )	从左到右
2	自加/自减运算符	++/--	从右到左
3	乘法运算符 除法运算符 取模运算符	* / %	从左到右
4	加法运算符 减法运算符	+ -	从左到右
5	小于 小于等于 大于 大于等于	< <= > >=	从左到右
6	等于 不等于	== !=	从左到右 从左到右
7	逻辑与	&&	从左到右
8	逻辑或	||	从左到右
9	赋值运算符和快捷运算符	= += *= /= %= -=	从右到左

- (7)类型转换
  - int a = int.Parse(字符串)； 其它类型的数值也可以使用Parse
  - 使用C语言的强制转换
- (8)程序控制结构
  - 顺序结构
  - 选择结构： if/else或 if/else if/else, switch...case
  - 循环结构：for, while,do... while, foreach
  - 示例
    - foreach(int a in list)
    - Console.WriteLine(a);

3.3 C#语言简介

(9)类与对象

- C#所有代码均在类中
- 类的定义示例
- C#中构造函数有三种：
  - 实例构造
  - 私有构造
  - 静态构造

```
class Demo
    {
        private int a;
        protected int b;
        public int c;
        public int d { get; set; }
        public int A { get => a; set => a = value; }
        public int B
        {
            get { return b; }
            set { b = value; }
        }
        public Demo()
        { }
        public int Add(int a, int b)
        {
            return a + b;
        }
    }
```

(10)接口

- 接口定义了所有类继承接口时应遵循的语法协议，派生类定义了语法协议 "怎么做" 部分。
- 接口定义了属性、方法和事件，这些都是接口的成员。
- 接口只包含了成员的声明,成员的定义是派生类的责任。
- 示例

```
interface IMyInterface
{    void MethodToImplement();
}
class InterfaceImplementer : IMyInterface
{
    public void MethodToImplement()
    {
        Console.WriteLine("MethodToImplement() called.");
    }
}
```

(11) is 运算符

- 检查对象与类之间的关系
- 示例：
  - Int32 b = 100;
  - if(b is Int32)
  - { … … }

(12) as 运算符

- 类型转换，先用is检查，再执行对象类型转换
- 示例：
  - Sphere obj = new Sphere(1);
  - ICalAreaAndVolumn myICal;
  - myICal = obj as ICalAreaAndVolumn;

(13) 异常处理

- 异常：与程序无关的异常原因造成的错误。
- 捕捉异常：

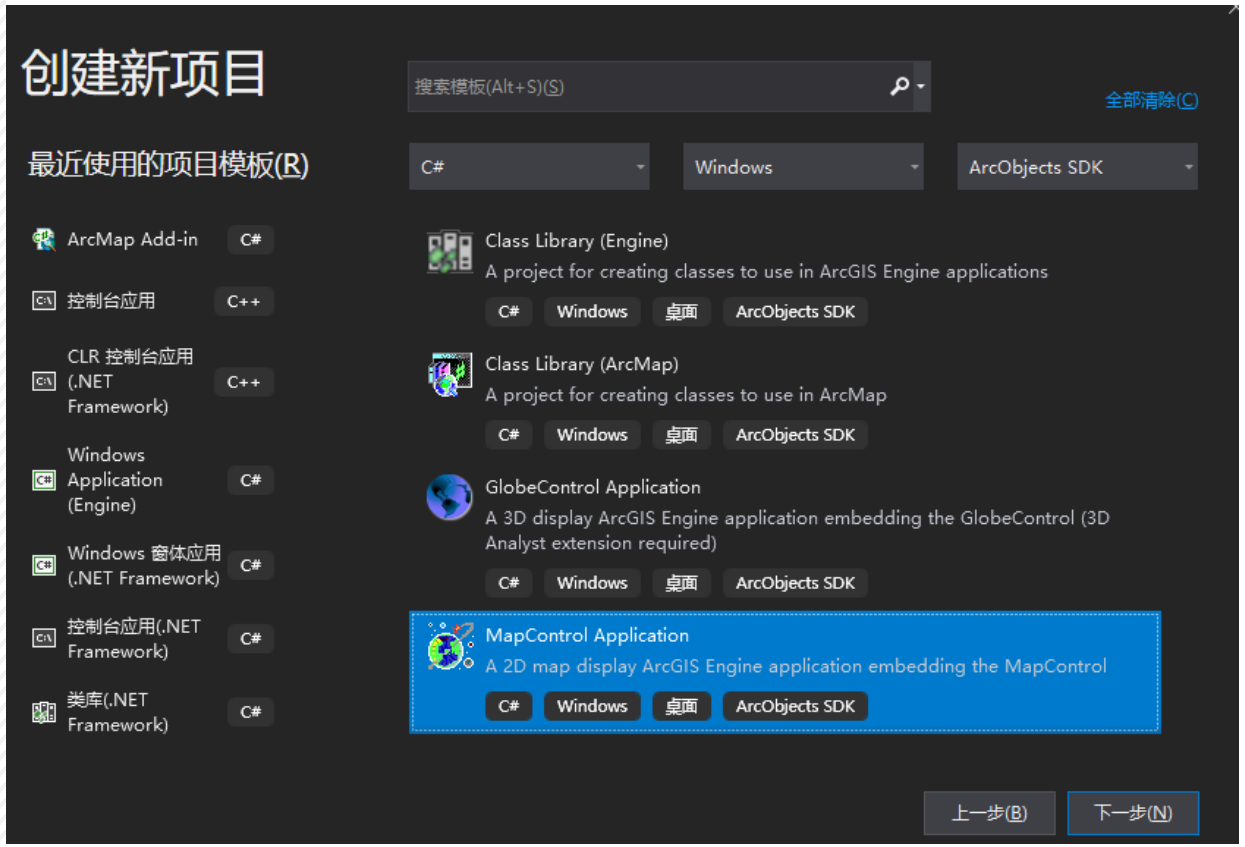
```
try
{
        要执行的程序体
}
catch (Exception e)
{
        要处理的异常
}
 finally
{
        最后执行程序
}
```

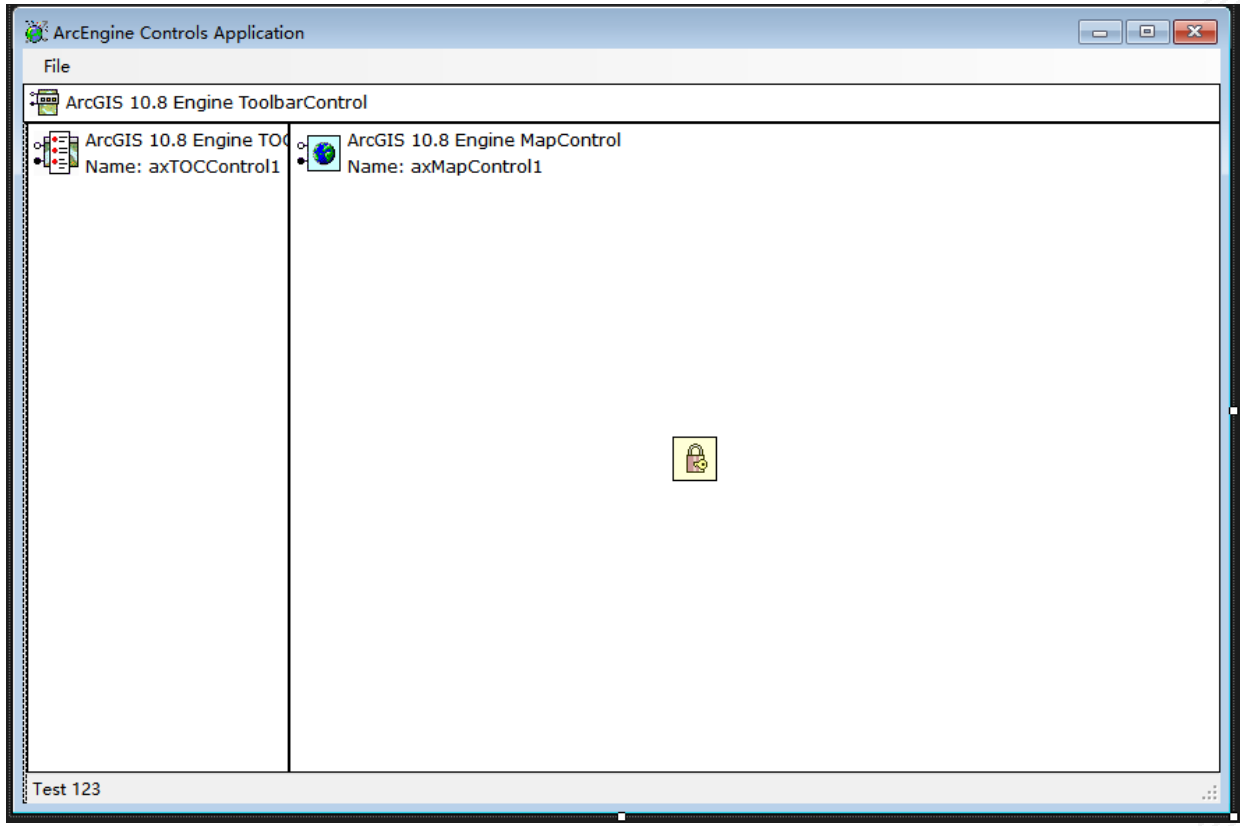
示例:

```
try
{
    int b = 0;
    int a = 1 / b;
}
catch (Exception e)
{
      MessageBox.Show("异常错误：" + e.Message);
}
```

3.4 AO开发示例

- 在VS2019里新建解决方案，利用向导生成项目MapControl Application项目

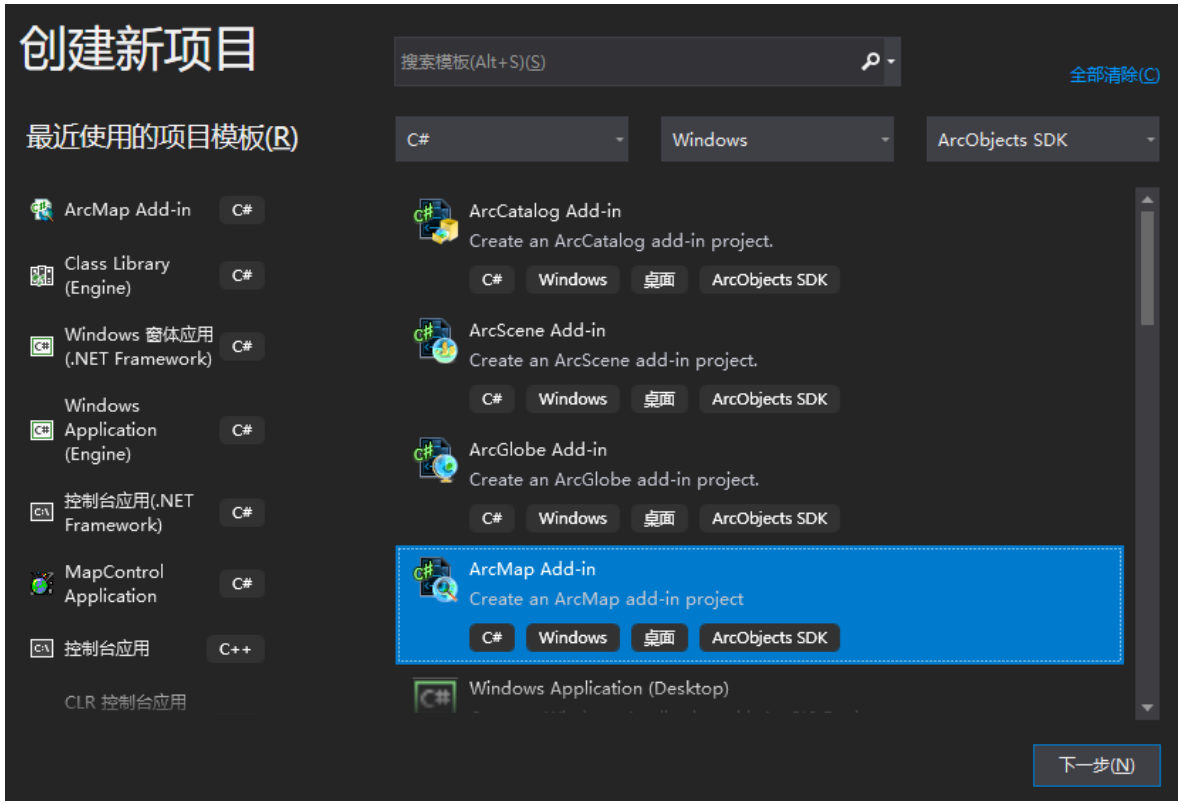


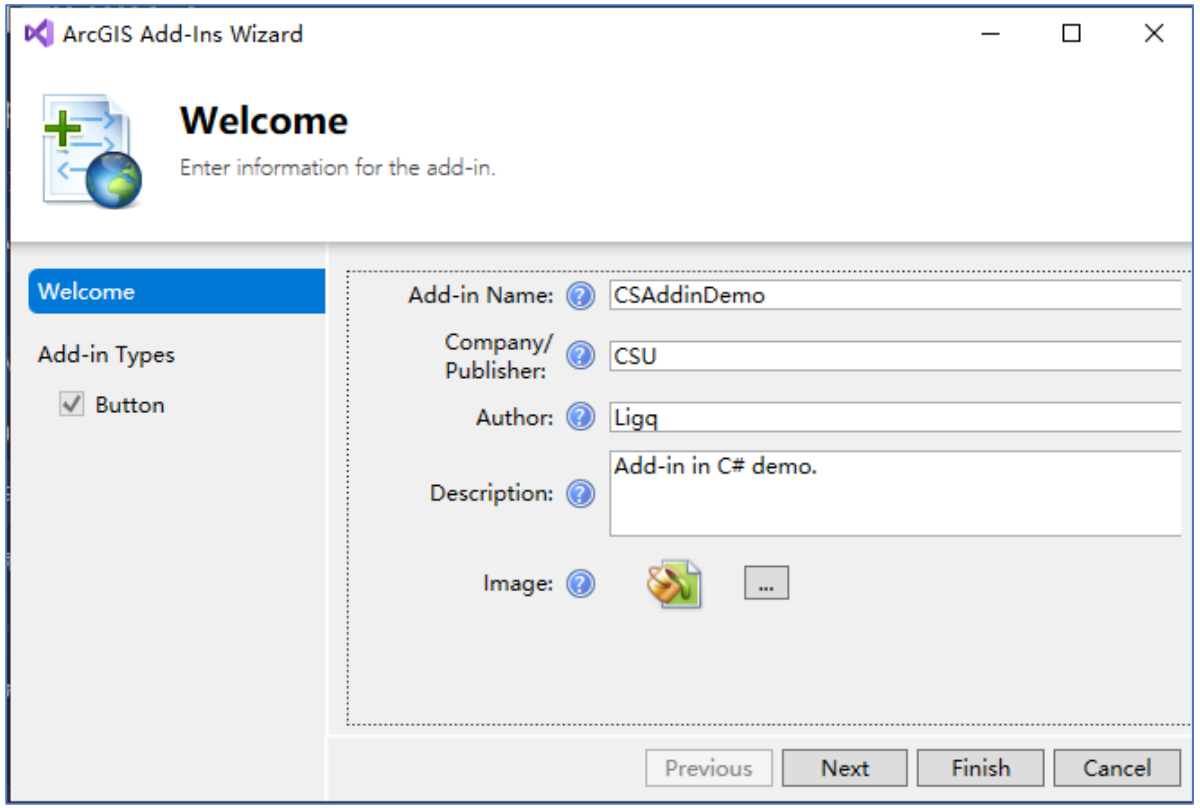


AO开发示例

Add-in项目实例

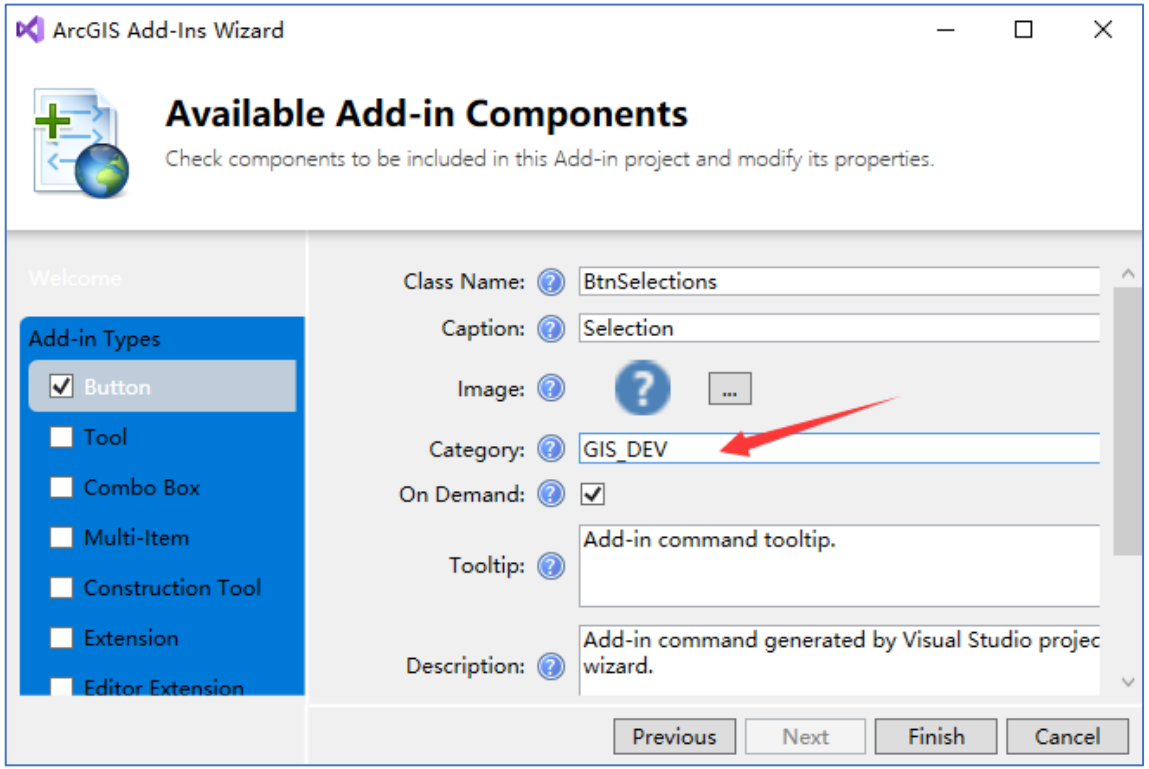
- 在VS2019中创建ArcMap Add-in项目





Add-in项目实例

- 添加组件



AO开发示例

Add-in项目实例

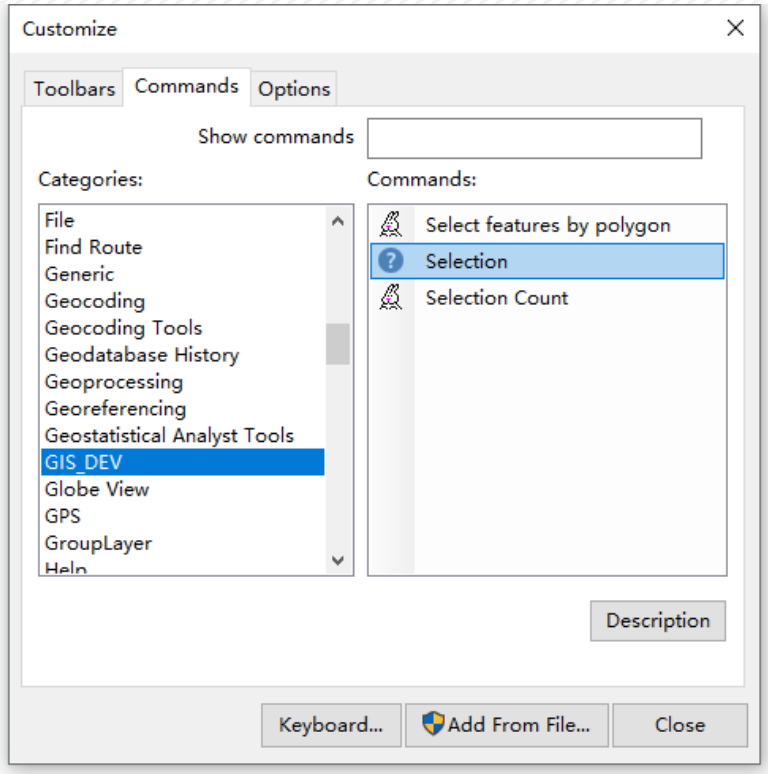
- 在BtnSelections类的OnClick事件中添加以下代码：

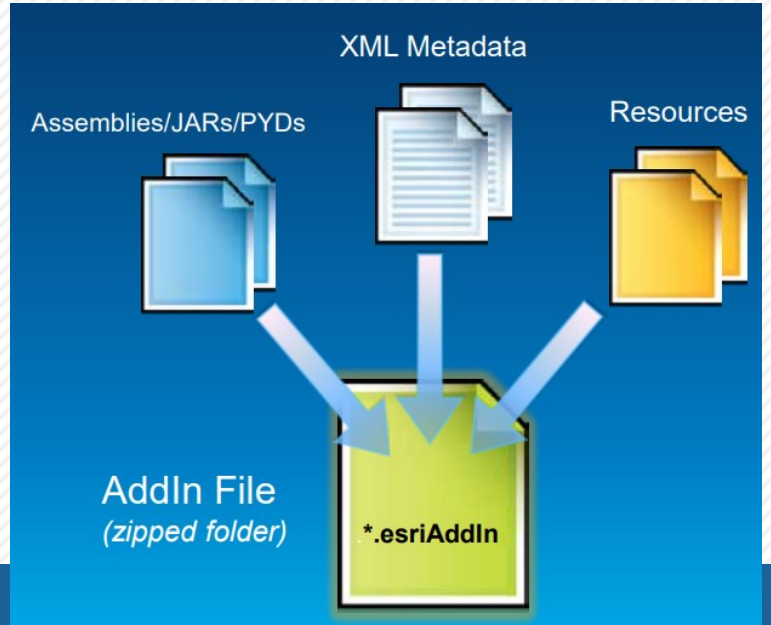
```
IMap pMap = ArcMap.Document.FocusMap;
List<IFeature> list = new List<IFeature>();
IEnumFeature pEnumFeat = (IEnumFeature)pMap.FeatureSelection;
pEnumFeat.Reset();
IFeature pfeat = pEnumFeat.Next();
while (pfeat != null)
{
   list.Add(pfeat);
   pfeat = pEnumFeat.Next();
}
MessageBox.Show("Selected Features:" + list.Count.ToString(), "GIS_DEV");
```

- **注意:**需要添加引用ESRI.ArcGIS.Carto和ESRI.ArcGIS.Geodatabase

3.4 AO开发示例

- Add-in项目实例
  - 点击运行，VS在编译完程序以后，自动启动ArcMap
  - 在ArcMap Customize对话框的Commands页框中选择GIS_DEV，从Commands列表中找到按钮，将其拖放到工具栏中.
  - 单独部署Add-in
    - 项目编译后，会在bin目录中生成esriAddIn文件，双击即可安装部署





本章小结

- 组件概念
- ArcEngine和ArcObjects简介
- C#语言介绍
- C#开发AE实例

开发实例

- 1. 在绘制的多边形范围内生成随机点
- 2. 点击地图添加点
- 3. 生成Voronoi多边形
- 4. 计算每个多边形中的点数量