Windows Forms编程“一点通”

北京理工大学计算机学院
金旭亮

Windows Form官方文档

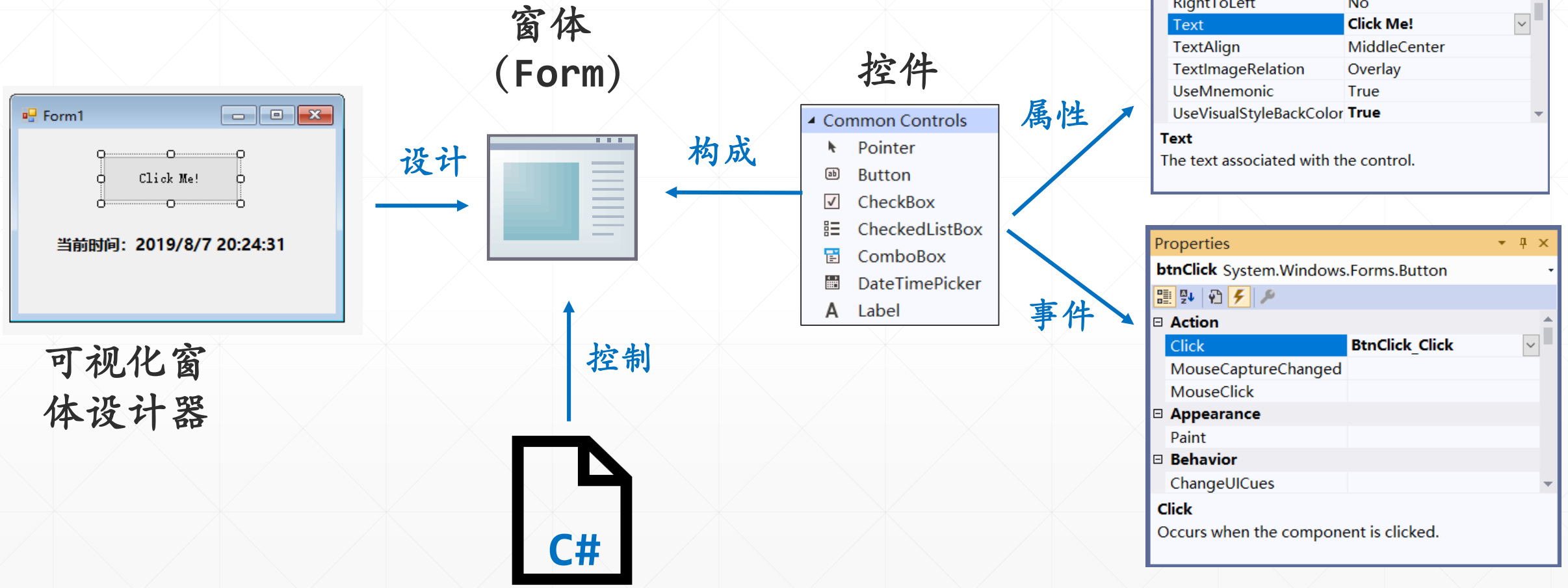
https://docs.microsoft.com/zh-cn/dotnet/framework/winforms/index

推荐与本网站的另一门在线课程“我的第一门编程语言（C#版）”配套学习

http://jinxuliang.com/course/CoursePortal/Details/543b979c137e481e6cbdb267

快速把握Windows Forms编程模型

Windows Forms编程模型与开发流程



Windows Forms的两个项目模板

1

Windows Forms App (.NET Framework)

A project for creating an application with a Windows Forms (WinForms) user interface

C# Windows Desktop

2

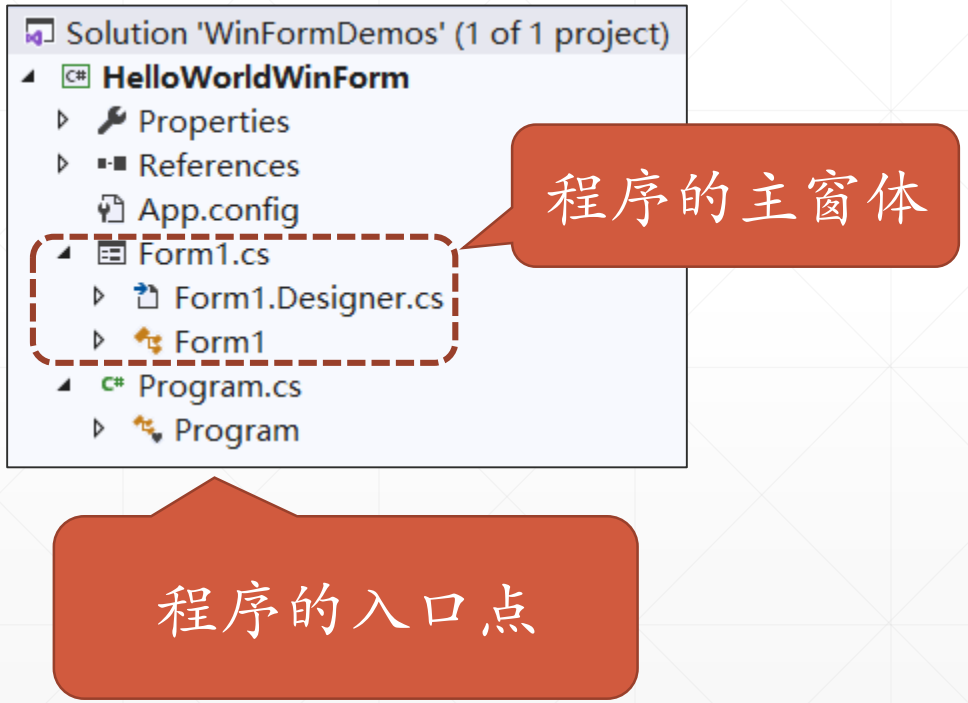
Windows Forms App (.NET Core)

A project for creating an application with a Windows Forms (WinForms) user interface

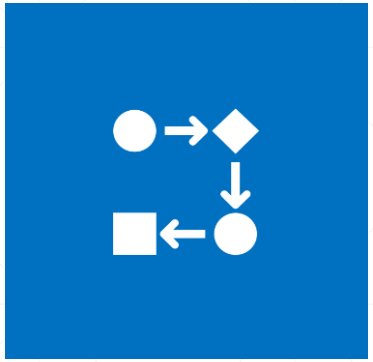
C# Windows Desktop

.NET Core 3.0以上版本支持，需要使用Visual Studio 2019

默认创建的项目文件



Windows Forms程序的入口点



```
0 references
static class Program
{
    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]
    0 references
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        //显示主窗体Form1，启动消息循环
        Application.Run(new Form1());
    }
}
```

Application类代表一个WinForm应用程序，包容N多的静态方法和属性。

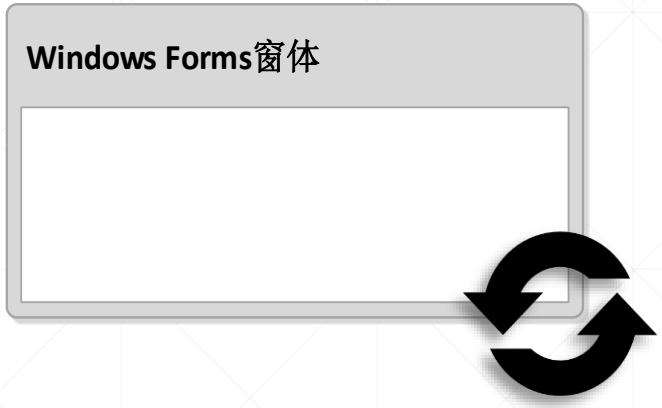
public sealed class **Application**

Member of System.Windows.Forms

Summary:

Provides static methods and properties to manage an application, such as methods to start and stop an application, to process Windows messages, and properties to get information about an application. This class cannot be inherited.

消息队列与消息循环




```
MSG msg;  //代表一条消息
BOOL bRet;
//从UI线程消息队列中取出一条消息
while((bRet = GetMessage(&msg, NULL, 0, 0 )) != 0)
{
    if (bRet == -1)
    {
        //错误处理代码，通常是直接退出程序
    }
    else
    {
        TranslateMessage(&msg); //转换消息格式
        DispatchMessage(&msg);  //分发消息给相应的窗体
    }
}
```

Win32 GUI程序中的消息循环（C++代码）

System.Windows.Forms.Form类中的窗体过程，由Windows Forms框架实现，你可以重写它以对特定的消息进行特定的处理。

protected override **void DefWndProc**(ref System.Windows.Forms.Message *m*)

Member of System.Windows.Forms.Form

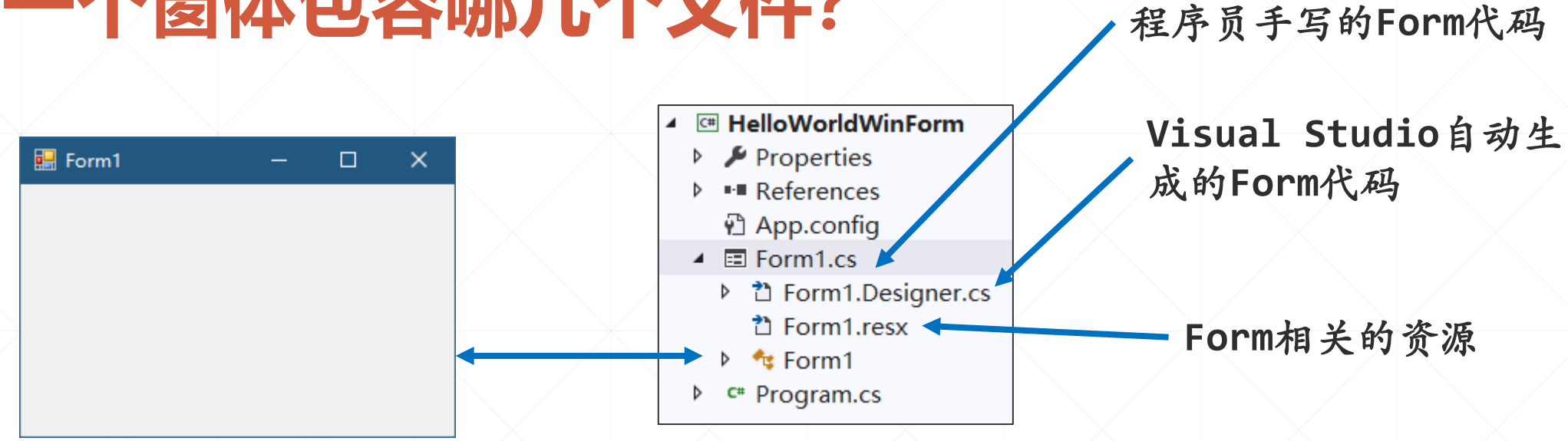
Summary:

Sends the specified message to the default window procedure.

Parameters:

m: The Windows System.Windows.Forms.Message to process.

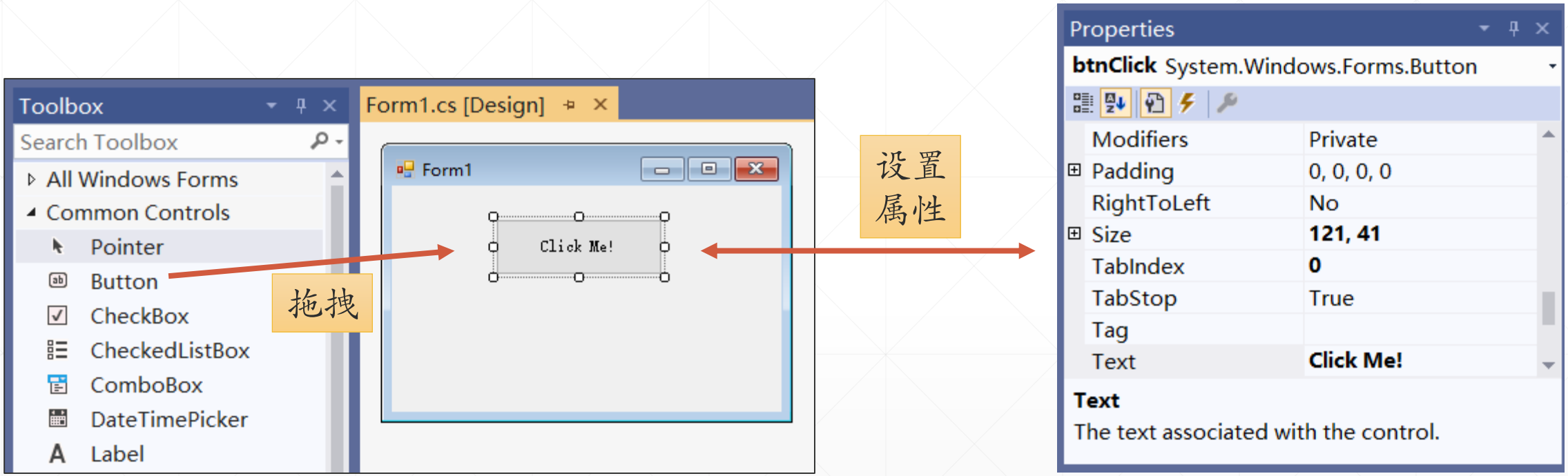
一个窗体包容哪几个文件?

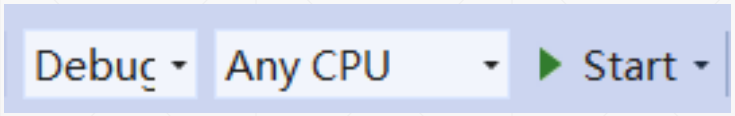


程序运行时，每个窗体都是相应Form类的实例

C#支持一种“分部类（partial class）”特性，允许把一个类放到不同的文件中，Visual Studio使用这个特性将自动生成的代码与程序员手写代码隔离开。

以“画图”的方式设计UI，所见即所得！





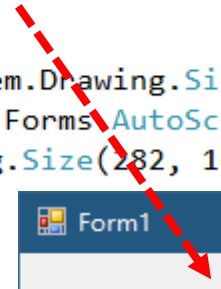
设计完窗体之后，在工具栏上点击“绿色三角形”按钮（见左图），启动程序，立即就可以看到窗体出现在屏幕上，与你在窗体设计器设计的一模一样！

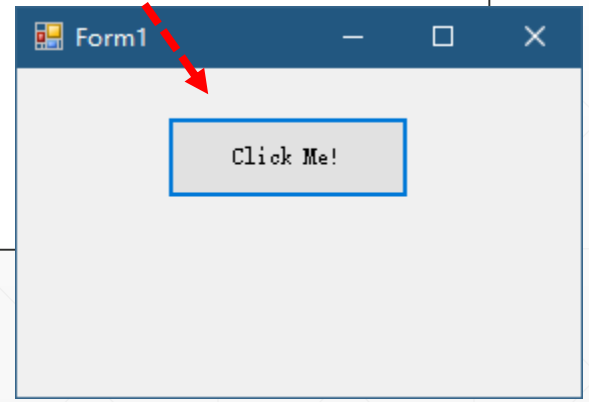
Form1.Designer.cs

```
1 reference
private void InitializeComponent()
{
    this.btnClick = new System.Windows.Forms.Button();
    this.SuspendLayout();
    //
    // btnClick
    //
    this.btnClick.Location = new System.Drawing.Point(75, 24);
    this.btnClick.Name = "btnClick";
    this.btnClick.Size = new System.Drawing.Size(121, 41);
    this.btnClick.TabIndex = 0;
    this.btnClick.Text = "Click Me!";
    this.btnClick.UseVisualStyleBackColor = true;
    //
    // Form1
    //
    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 12F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(282, 164);
    this.Controls.Add(this.btnClick);
    this.Name = "Form1";
    this.Text = "Form1";
    this.ResumeLayout(false);

}
```

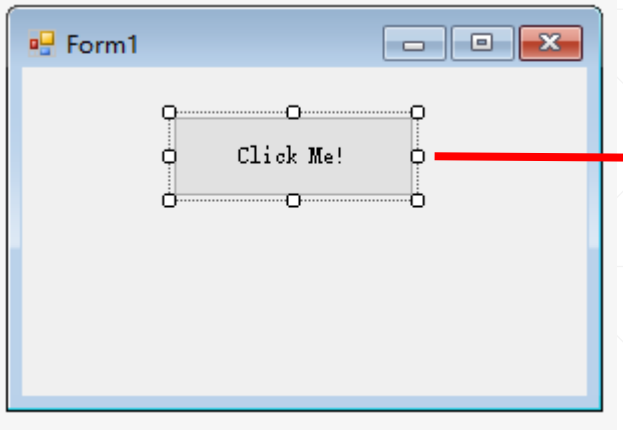
```
3 references
public partial class Form1 : Form
{
    1 reference
    public Form1()
    {
        //实例化各个控件
        InitializeComponent();
    }

}
```

Form1.cs





程序运行时的所有控件，都是new出来的……

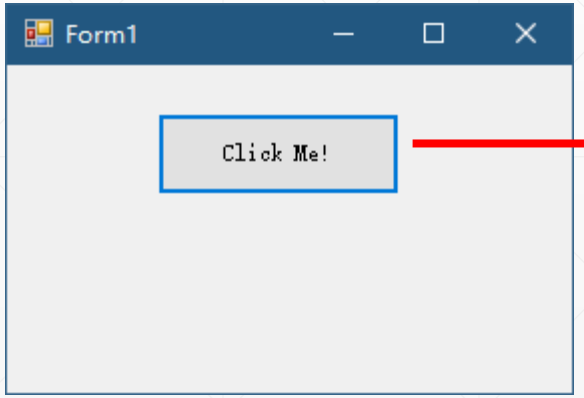


双击

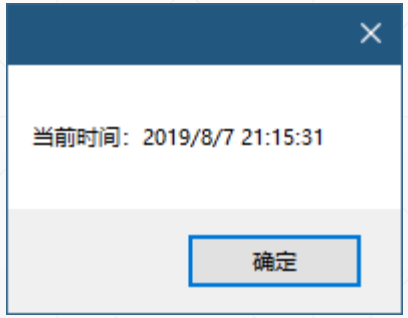
```
3 references
public partial class Form1 : Form
{
    1 reference
    public Form1()...

    //按钮单击事件响应代码
    1 reference
    private void BtnClick_Click(object sender, EventArgs e)
    {
        MessageBox.Show($"当前时间：{DateTime.Now}");
    }
}
```

开发时

运行时



单击

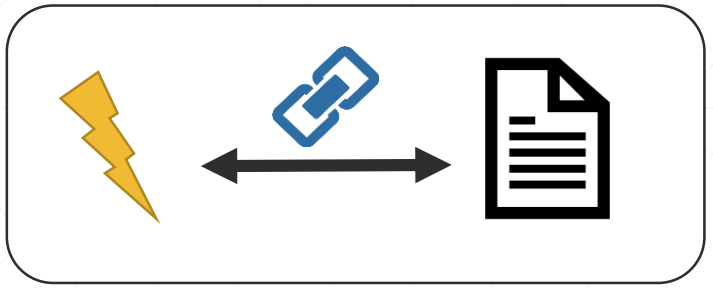


“事件驱动”的运行模式

事件与事件响应方法如何挂接?

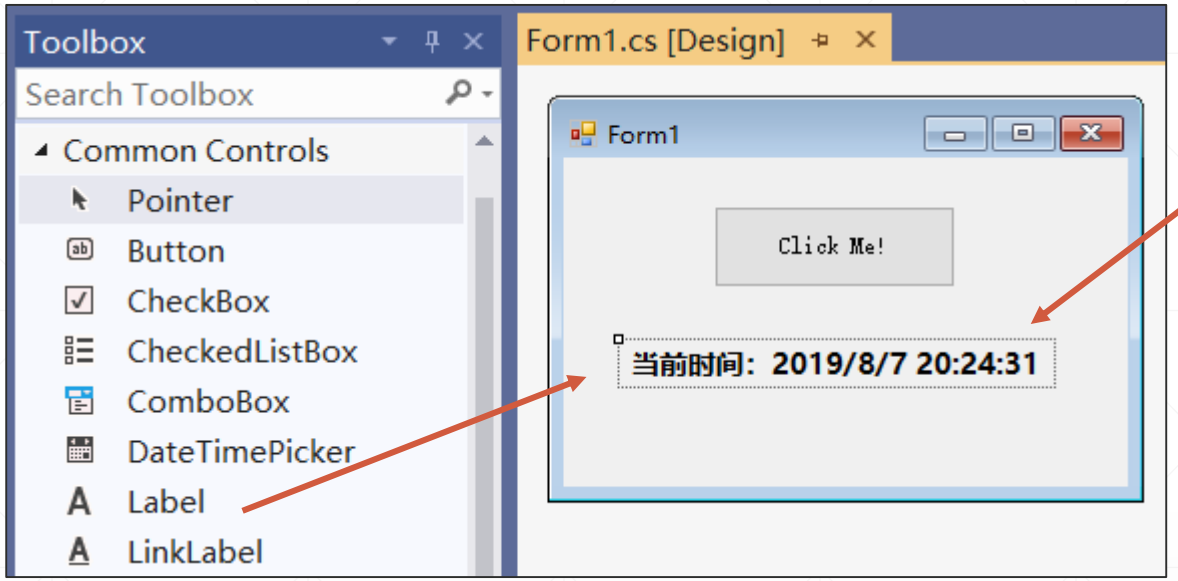
Form1.Designer.cs

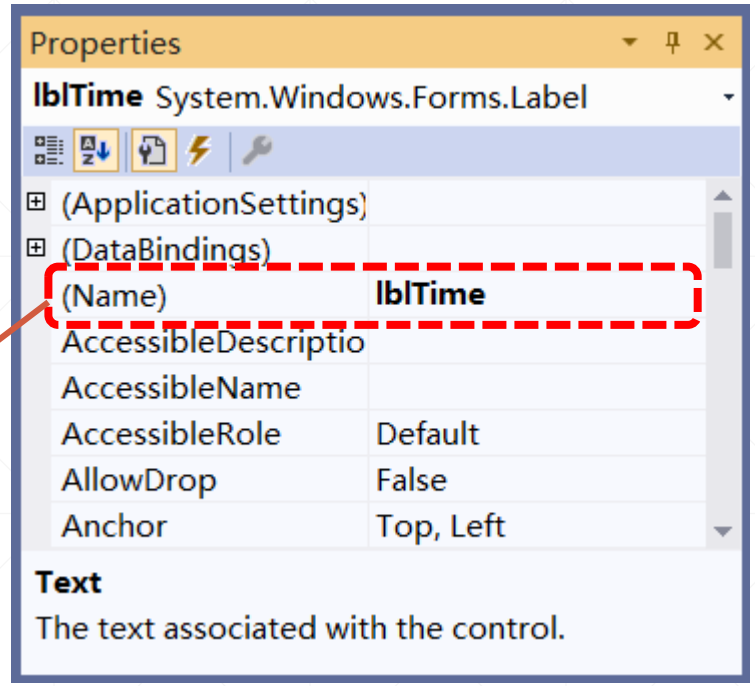
```
1 reference
private void InitializeComponent()
{
    this.btnClick = new System.Windows.Forms.Button();
    this.SuspendLayout();
    //
    // btnClick
    //
    this.btnClick.Location = new System.Drawing.Point(75, 24);
    this.btnClick.Name = "btnClick";
    this.btnClick.Size = new System.Drawing.Size(121, 41);
    this.btnClick.TabIndex = 0;
    this.btnClick.Text = "Click Me!";
    this.btnClick.UseVisualStyleBackColor = true;
    this.btnClick.Click += new System.EventHandler(this.BtnClick_Click);
    //
    // Form1
    //
    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 12F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(282, 164);
    this.Controls.Add(this.btnClick);
    this.Name = "Form1";
    this.Text = "Form1";
    this.ResumeLayout(false);

}
```



Windows Forms窗体的事件驱动机制，是建立在“委托（Delegate）”基础之上的。

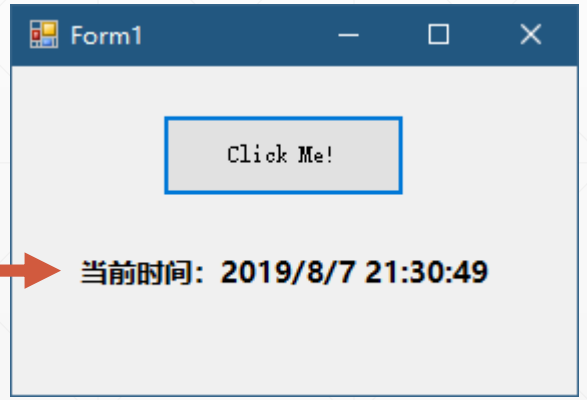
控件名字的重要性





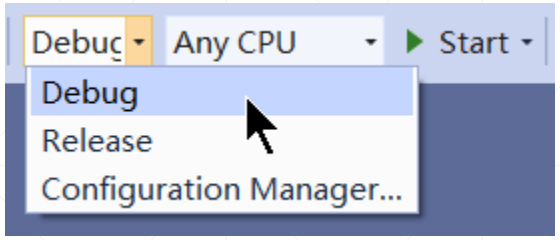

```
//按钮单击事件响应代码
1 reference
private void BtnClick_Click(object sender, EventArgs e)
{
    // MessageBox.Show($"当前时间：{DateTime.Now}");

    //通过控件Id访问特定的控件
    lblTime.Text=$"当前时间：{DateTime.Now}";
}
```

点击按钮则自动更新

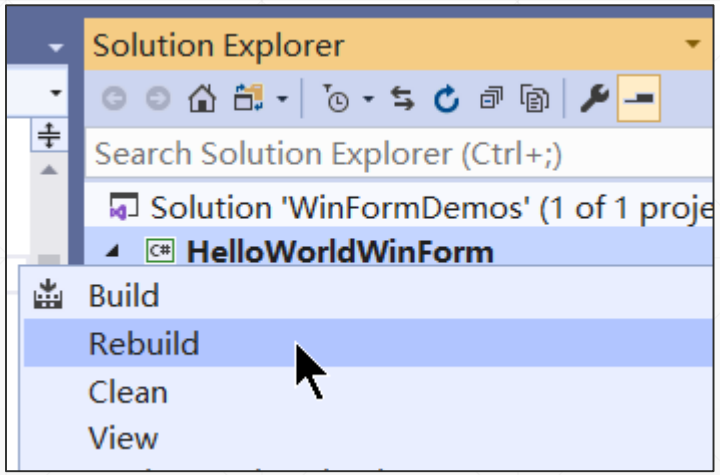


分发Windows Forms应用

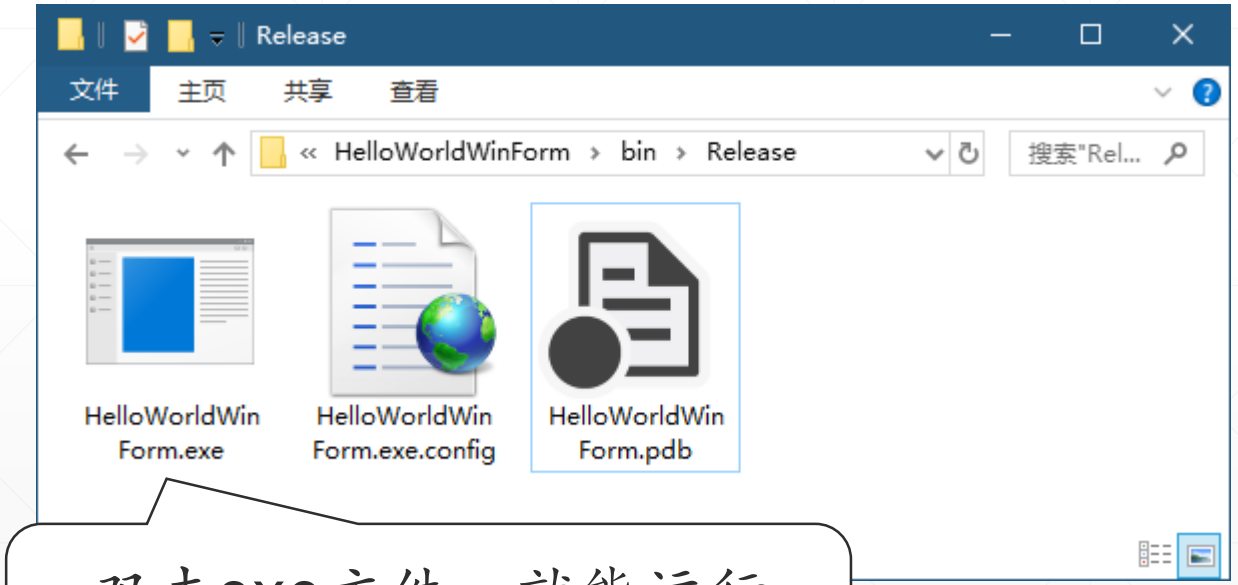
1 改为Release状态



2 重新生成项目



3 将整个Release文件夹拷贝到另一个安装有相应版本.NET Framework或.NET Core的计算机上即可。



双击exe文件，就能运行WinForm程序。

小结

- Windows Forms采用所见即所得的UI设计方式，全面向对象的编程模型和事件驱动的运行模式，具有很高的开发效率和很好的兼容性。
- Windows Forms最适合的场景：
  （1）开发具有标准界面风格的小规模的带有工具性质的桌面应用程序
  （2）原型工具：快速编写一个可以跑的程序，以捕获或验证用户需求。
- Windows Forms的编程模型比较直观，可视化界面设计器易于使用，学习曲线平滑，是学习.NET技术的极佳切入点。