在子线程内刷新主线程的控件（一个线程操不属于本线程的控件）

【注意】一个线程读取不属于本线程控件的属性，直接操作控件即可，

不需要利用控件的invoke和begininvoke方法

控件的invoke和begininvoke方法的本质：并没有开辟一个新线程运行刷新控件的方法，

而是把该方法传送给主线程，让主线程执行之，这样就保证了，所有刷新控件的动作都是

在主线程上执行的，避免了多个线程刷新同一个控件带来的错误。

1，更新控件不需要参数

if(control.invokerequest)

{

control.invoke(new Action(() =>{control.text = "emmmm";}))

}

else

{

control.text = "emmmm";

}

2，更新控件需要参数

if(control.invokerequest)

{

control.invoke(new Action<string>((str) =>{control.text = str;},param));

}

else

{

control.text = param;

}

如果只需要一个参数，param是一个变量，如果需要多个参数，param是object[]，数组引用

param[] = { "bingo", 100 };

new Action<string，int>((str,i) =>{control.text = str;},param)

————————————————————————————————————————————

下面是例子，不是特别重要

【在主线程内另辟子线程，子线程执行一些计算任务，同时会更新控件】

访问控件的函数，函数内部是更新控件的代码

public void upgradeControl( ) {

此处是更新控件的代码。

尽量不要含有其他任何代码；

【如果必须需要一些计算，使用其

计算结果来更新UI，那么请把这些计算代码

放置到

】

}

定义访问控件的委托类型

public delegate void upgradeControlDelegate( )

在主线程内新开辟的子线程

Thread mythread = new Thread(startThreadMethod);

mythread.start( );

线程的入口函数

public void startThreadMethod( ) {

//线程内的计算任务，可以是消耗时间的计算（访问数据库，从网络取数据等）代码

upgradeControlDelegate ucd = upgradeControl;

if(control.InvokeRequired)

control.begininvoke(ucd );

else

control.text = "我更新了";

//子线程可以继续执行代码

}

上述代码执行流程如下：

1. 点击button，子线程和主线程并行
2. 先运行子线程的计算任务代码
3. 子线程把invork里面的方法送给主线程，主线程执行该方法，同时，子线程并不阻塞，继续往下执行
4. 主线程执行完毕来自invork的方法，继续执行主线程的剩余代码

【感悟】

1. 主线程的代码一定不要含有任何计算型的代码，只能含有修改控件的代码，包括控件的click事件等，都属于主线程。
2. 子线程修改主线程的控件的属性，需要使用invork；读控件的属性不需要invork。

namespace WindowsFormsApplication3

{

//定义委托类型，和 子线程如何更新控件的方法一致

public delegate void upgradeTextBoxDelegate(string str);

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

System.Windows.Forms.Control.CheckForIllegalCrossThreadCalls = true;

}

/// <summary>

/// button的click事件

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void myButton\_Click(object sender, EventArgs e)

{

//此处禁止有计算任务型的代码

Thread mythread = new Thread(upgradeTextBoxStartThreadMethod);

mythread.Start();

}

/// <summary>

/// 子线程的入口函数

/// </summary>

private void upgradeTextBoxStartThreadMethod()

{

upgradeTextBoxDelegate utbd = new upgradeTextBoxDelegate(upgradeTextBox);

/\*模拟从网络获取数据internetStr\*/

Thread.Sleep(3000);//取数据耗时3秒

string internetStr = " love";

string myString = myLabel.Text + internetStr;

//子线程修改控件的属性

if(myButton.InvokeRequired)

{

myTextBox.Invoke(utbd,myString);

}

else

{

myTextBox.BackColor = Color.Gray;

myTextBox.Text = myString;

}

/\*子线程等待UI更新完毕，继续执行下面的代码\*/

Thread.Sleep(2000);

myTextBox.BeginInvoke(utbd, myTextBox.Text + " code !");

}

/// <summary>

/// 子线程（如何）修改控件的方法

/// </summary>

/// <param name="msg"></param>

private void upgradeTextBox(string msg)

{

myTextBox.BackColor = Color.White;

myTextBox.Text = msg;

}

}

}