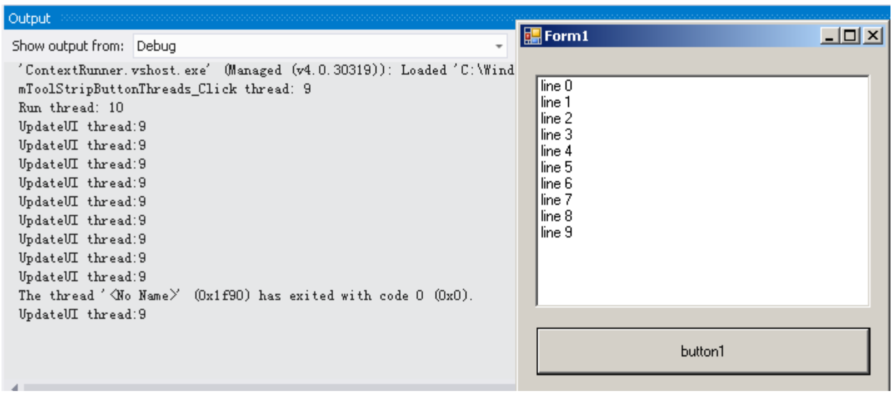
Post aync result to the main UI thread

```
using System;
using System.Diagnostics;
using System.Threading;
using System.Windows.Forms;

namespace ContextRunner
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Trace.WriteLine("mToolStripButtonThreads_Click thread: " +
Thread.CurrentThread.ManagedThreadId);

            // grab the sync context associated to this
            // thread (the UI thread), and save it in uiContext
            // note that this context is set by the UI thread
            // during Form creation (outside of your control)
            // also note, that not every thread has a sync context attached to it.
            SynchronizationContext uiContext = SynchronizationContext.Current;

            // create a thread and associate it to the run method
            Thread thread = new Thread(Run);

            // start the thread, and pass it the UI context,
            // so this thread will be able to update the UI from within the thread
            thread.Start(uiContext);
        }

        private void Run(object state)
        {
            Trace.WriteLine("Run thread: " + Thread.CurrentThread.ManagedThreadId);
            SynchronizationContext uiContext = state as SynchronizationContext;

            for (int i = 0; i < 10; i++)
            {
                // normally you would do some code here to grab items from the
database.
               // or some long computation
                Thread.Sleep(10);
                // use the ui context to execute the UpdateUI method,
                // this insure that the UpdateUI method will run on the UI thread.
                uiContext.Post(UpdateUI, "line " + i.ToString());

                //try
                //{
                //    uiContext.Send(UpdateUI, "line " + i.ToString());
                //}
                //catch (Exception e)
                //{
                //    Trace.WriteLine(e.Message);
                //}
            }
        }

        /// <summary>
        /// This method is executed on the main UI thread.
        /// </summary>
        private void UpdateUI(object state)
```

```
            {
                Trace.WriteLine("UpdateUI thread:" +
Thread.CurrentThread.ManagedThreadId);
                string text = state as string;
                mListBox.Items.Add(text);

                //throw new Exception("Boom");
            }
        }
}
```



```
'ContextRunner.vshost.exe' (Managed (v4.0.30319)): Loaded 'C:\Wind
mToolStripButtonThreads_Click thread: 9
Run thread: 10
UpdateUI thread:9
UpdateUI thread:9
UpdateUI thread:9
UpdateUI thread:9
UpdateUI thread:9
UpdateUI thread:9
UpdateUI thread:9
UpdateUI thread:9
UpdateUI thread:9
The thread '<No Name>' (0x1f90) has exited with code 0 (0x0).
UpdateUI thread:9
```

```
line 0
line 1
line 2
line 3
line 4
line 5
line 6
line 7
line 8
line 9
```

```
// ***
//This means that Post will not wait for the execution of the delegate to complete.
//Post will "Fire and Forget" about the execution code within the delegate.
//It also means that you cannot catch exceptions as we did with the Send method.
//Suppose an exception is thrown, it will be the UI thread that will get it;
//unhanding the exception will terminate the UI thread.

*** When try-catch added, exception will be thrown to the Run function, not arrived
UI thread, in this case.
```