

Forum:[Hack Forums](#)

Thread: [Asynchronous communication between threads](#)

michalmonday - July 20, 2018, 4:41 am

Hi, I found a tutorial about [asynchronous socket server](#), it's great but 1 thing troubles me about it...

It has separate threads for each client to receive data, these separate clients communicate with the main (GUI) thread to display received data. And it's the way of this communication that seems to be a little bit too complex to me and I wonder if it could be done in a slightly different way. To transfer received text to the GUI thread it uses the following:

1.

Code `private void AppendToRichEditControl(string msg) // (the purpose of it is to allow calling the same function from client threads and main GUI thread too, it's just a wrapper for the function that actually does append the text)`

2.

Code `richTextBoxReceivedMsg.BeginInvoke(new UpdateRichEditCallback(OnUpdateRichEdit), pList);`

Where "UpdateRichEditCallback" is a delegate function defined as:

Code `public delegate void UpdateRichEditCallback(string text);`

"OnUpdateRichEdit" is the function which actually appends the text

Summarizing, it's a complex process just because it is multithreaded, in order to create some functionality usually I would:

- create a desired function
- call a function

In this case it's:

- create a function wrapper (to detect which thread is calling the function and act accordingly)
- define and use delegate function which calls the desired function
- create desired function
- call a function wrapper

E.g.:

```
-private void AppendToRichEditControl(string msg)
-public delegate void UpdateRichEditCallback(string text);
-private void OnUpdateRichEdit(string msg)
-the function wrapper contains: "richTextBoxReceivedMsg.BeginInvoke(new
UpdateRichEditCallback(OnUpdateRichEdit), pList);"
```

Another example from that code:

```
-private void UpdateClientListControl()
-public delegate void UpdateClientListCallback();
-void UpdateClientList()
-the function wrapper contains: "listBoxClientList.BeginInvoke(new
UpdateClientListCallback(UpdateClientList), null);"
```

So for each communication feature between client threads and GUI thread there's a "4-step" procedure. It isn't messy in the example but if I wanted to expand the code and add more such "4-step" features I fear that the code will become a mess very quickly...

Would it be a good way to replace "AppendToRichEditControl(msg)" procedure with something like: "ParseClientMsgAndTakeAction_from_GUI_thread(client_number_id, msg)"? So it would be decided on the GUI thread what to do with it and there would be only 1 or 2 of these "4-step" communicational procedures instead. Would that be a proper way of implementing it? Or would it be just too much work for the GUI thread to parse all messages from all the other threads?

Daedun - July 20, 2018, 6:15 pm

This can be easily done with a single function :P

The Control.InvokeRequired property lets you know if you're on a thread other than the one used to create the form or control

[https://msdn.microsoft.com/en-us/library/system.windows.forms.control.invokerequired\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.windows.forms.control.invokerequired(v=vs.110).aspx)

```
Code private void UpdateRichTextBox(string text) {  
    if (InvokeRequired)  
        BeginInvoke(new Action<string>(UpdateRichTextBox), text);  
    else {  
        // On GUI Thread  
    }  
}
```

This way the method can be called from any thread

If its on the GUI thread, it just updates the text

If its on any other thread, it will invoke the GUI thread to update the text

ppaida - July 22, 2018, 8:49 pm

Use the async/await Task pattern.

The Task is the wrapper for your async method calls.