

Re: About the remoting event

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2008-07/msg00166.html>

- *From:* fairyvoice <fairyvoice@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Tue, 1 Jul 2008 18:50:00 -0700
-

thanks Nicholas, when you said "callback interface" did you mean the one in the asynchronous model, i am not very clear.
May you tell me more precisely or show me a few codes? thanx

"Nicholas Paldino [.NET/C# MVP]" wrote:

I find that using events is a PITA when using remoting. Rather, I would define a callback interface that the client implements, and then make sure that the class that implements it derives from MarshalByRefObject, and send that to the server to call back onto when you want to fire an event. It's much cleaner, and it works.

--
- Nicholas Paldino [.NET/C# MVP]
- mvp@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

"fairyvoice" <fairyvoice@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message news:4604E3F5-AF6B-4DDD-9C50-4FCBBD4EBB1F@xxxxxxxxxxxxxxxxxxxx

in a remoting application, i set a event in the host, and let the client to book it, and in the host side i set the TypeFilterLevel to Full and open the callback port in the client side, but told that these was an exception on the invoked object.
can anyone tell my why? It is a very simple application just to test, and here is the code, i am using vs2008, thanx in advaned.

```
-----server side-----  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using System.Runtime.Remoting;  
using System.Runtime.Remoting.Channels;  
using System.Runtime.Remoting.Channels.Ipc;
```

Re: About the remoting event

```
using System.Runtime.Remoting.Channels.Tcp;
using System.Runtime.Serialization.Formatters;
using System.IO;
using System.Collections;

namespace RcServer
{
    class Program
    {
        static void Main(string[] args)
        {
            IDictionary pros = new Hashtable();
            pros["portName"] = "host";
            BinaryServerFormatterSinkProvider ss = new
            BinaryServerFormatterSinkProvider();
            ss.TypeFilterLevel = TypeFilterLevel.Full;
            IChannel cnl = new IpcChannel(pros, new
            BinaryClientFormatterSinkProvider(), ss);

            ChannelServices.RegisterChannel(cnl, false);
            RemotingConfiguration.RegisterWellKnownServiceType(typeof(Cat),
            "cat", WellKnownObjectMode.Singleton);
            Cat cat = new Cat();
            Console.WriteLine("go");
            Console.ReadLine();
        }
    }

    public class Cat : MarshalByRefObject
    {
        public event EventHandler OnScreaming;
        public void Scream()
        {
            Console.WriteLine("i am wake");
            if (this.OnScreaming != null)
                this.OnScreaming(this, null);
        }
    }
}

-----client side-----
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Ipc;
using System.Runtime.Serialization.Formatters;
```

Re: About the remoting event

```
using System.Collections;
using RcServer;

namespace RcClient
{
class Program
{
static void Main(string[] args)
{
IChannel cnl = new IpcChannel("MyCallback");
ChannelServices.RegisterChannel(cnl, false);
RemotingConfiguration.RegisterWellKnownClientType(typeof(Cat),
@"ipc://host/cat");
Cat c = (Cat)Activator.CreateInstance(typeof(Cat));
Rat jy = new Rat();
jy.ThereIsaCat(c);
c.Scream();

}
}

[Serializable]
public class Rat:MarshalByRefObject
{
public void ThereIsaCat(Cat cat)
{
cat.OnScreaming += new EventHandler(this.cat_OnScreaming);
}

void cat_OnScreaming(object sender, EventArgs e)
{
Console.WriteLine("cat's awake, run quick!");
}
}
}
```