Introduction til .NET remoting i C#

Denne artikel beskriver teorien bag .NET remoting og viser nogle simple kode eksempler i C#.

Den forudsætter kendskab til C#. Der er en anden artikel med præcis samme indhold bare i VB.NET !

Skrevet den **04. Feb 2009** af **arne_v** I kategorien **Programmering / C#** | ★★★☆☆

Historie:
V1.0 - 18/01/2004 - original
V1.1 - 31/01/2004 - forbedret formatering
V1.2 - 09/02/2004 - flere ændringer af formatering
V1.3 - 11/04/2004 - tilføj avanceret eksempel
Baggrund
.NET remoting svarer konceptuelt til Java RMI, men har dog
en noget anderledes implementering.

.NET remoting bygger på det såkaldte Proxy Pattern (et af de originale
patterns fra Design Patterns af Erich Gamma m.fl.).

Ideen er at en client kan kalde remote server kode på samme måde
som lokal kode.

Selve kaldet foregår som:

```
client kode--stub kode--(netværk)--skeleton kode--server kode
```

Stub koden har samme interface som server koden. Stub koden
pakker alle argumenter ned (serialiserer) og sender dem
over en socket til skeleton koden.

Skeleton koden udpakker alle argumenter (deserialiserer) og
kalder server koden.

Retur værdien sende tilbage på samme vis bare modsat.

Både stub og skeleton koden er 100% dynamisk i .NET frameworket.
(der skal ikke genereres noget kode).

Man skelner mellem:
SAO (Server Activated Objects)
CAO (Client Activated Objects)

Og med SAO skelner man mellem:
SingleCall
Singleton

SAO Singlecall betyder at der instantieres et remote objekt per kald.

SAO Singleton betyder at der instantieres et enkelt remote objekt ad gangen (bemærk at det ikke er et ægte singleton objekt - det kan blive garbage collectet og et nyt objekt instantieret - der er bare aldrig mere end et ad gangen).

CAO betyder at der instantieres et remote objekt hver gang klienten instantierer et lokalt objekt.

Eksempler
Lad os tage nogle eksempler.

Calc.cs (fælles for alle 3 eksempler)

```
using System;

// remote object klasse
// skal extende MarshalByRefObject
public class Calc : MarshalByRefObject
{
    // object nummer ud af total
    private int no;
    private static int total = 0;
    // synkroniserings objekt
    private object sync = new object();
    // constructor som sætter objekt nummer
    public Calc()
    {
        lock(sync)
        {
            total++;
            no = total;
        }
    }
    // metoder der skal kaldes remote
    public int add(int a, int b)
    {
        return (a + b);
    }
    public int mul(int a, int b)
    {
        return (a * b);
    }
    // metode til at checke egenskaberne
    public string ID
    {
        get
        {
            return (no + " of " + total);
        }
    }
}
```

SAO Singlecall:

Server1.cs

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;

// server klasse
class Server1
{
    public static void Main(string[] args)
    {
        // registrere channel
        ChannelServices.RegisterChannel(new TcpServerChannel(50000));
        // registrere well known type for singlecall
        RemotingConfiguration.RegisterWellKnownServiceType(typeof(Calc),
"Calc",

WellKnownObjectMode.SingleCall);
        Console.Write("Press enter to exit");
        Console.ReadLine();
    }
}
```

Client1.cs

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;

// test client klasse
class Client1
{
    // måde 1 til at kalde remote server
    private static void test1()
    {
        // get proxy objekt
        Calc clc = (Calc)Activator.GetObject(typeof(Calc),
"tcp://localhost:50000/Calc");
        // kald metoder for at teste funktionalitet
        Console.WriteLine(clc.add(2, 3));
        Console.WriteLine(clc.mul(2, 3));
        // kald metode som viser at der instatieres et nyt objekt per kald
        Console.WriteLine(clc.ID);
    }
    // måde 2 til at kalde remote serverr
    private static void test2()
    {
        // get proxy objekt
        Calc clc = (Calc)RemotingServices.Connect(typeof(Calc),
```

```
"tcp://localhost:50000/Calc");
        // kald metoder for at teste funktionalitet
        Console.WriteLine(clc.add(2, 3));
        Console.WriteLine(clc.mul(2, 3));
        // kald metode som viser at der instatieres et nyt objekt per kald
        Console.WriteLine(clc.ID);
    }
    // måde 3 til at kalde remote server
    private static void test3()
    {
        // get proxy objekt
        RemotingConfiguration.RegisterWellKnownClientType(typeof(Calc),
"tcp://localhost:50000/Calc");
        Calc clc = new Calc();
        // kald metoder for at teste funktionalitet
        Console.WriteLine(clc.add(2, 3));
        Console.WriteLine(clc.mul(2, 3));
        // kald metode som viser at der instatieres et nyt objekt per kald
        Console.WriteLine(clc.ID);
    }
    public static void Main(string[] args)
    {
        // brug TCP/IP
        ChannelServices.RegisterChannel(new TcpClientChannel());
        // test 3 måder
        test1();
        test2();
        test3();
    }
}
```

build

```
csc /optimize+ /t:library /out:Calc.dll Calc.cs
csc /optimize+ /t:exe /r:Calc.dll /out:Server1.exe Server1.cs
csc /optimize+ /t:exe /r:Calc.dll /out:Client1.exe Client1.cs
```

SAO Singleton:

Server2.cs

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;

// server klasse
class Server2
{
    public static void Main(string[] args)
    {
        // registrere channel
```

```
        ChannelServices.RegisterChannel(new TcpServerChannel(50000));
        // registrere well known type for singleton
        RemotingConfiguration.RegisterWellKnownServiceType(typeof(Calc),
"Calc",

WellKnownObjectMode.Singleton);
        Console.Write("Press enter to exit");
        Console.ReadLine();
    }
}
```

Client2.cs

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;

// test client klasse
class Client2
{
    // måde 1 til at kalde remote server
    private static void test1()
    {
        // get proxy objekt
        Calc clc = (Calc)Activator.GetObject(typeof(Calc),
"tcp://localhost:50000/Calc");
        // kald metoder for at teste funktionalitet
        Console.WriteLine(clc.add(2, 3));
        Console.WriteLine(clc.mul(2, 3));
        // kald metode som viser at der kun instatieres et objekt som deles af
alle kald
        Console.WriteLine(clc.ID);
    }
    // måde 2 til at kalde remote serverr
    private static void test2()
    {
        // get proxy objekt
        Calc clc = (Calc)RemotingServices.Connect(typeof(Calc),
"tcp://localhost:50000/Calc");
        // kald metoder for at teste funktionalitet
        Console.WriteLine(clc.add(2, 3));
        Console.WriteLine(clc.mul(2, 3));
        // kald metode som viser at der kun instatieres et objekt som deles af
alle kald
        Console.WriteLine(clc.ID);
    }
    // måde 3 til at kalde remote server
    private static void test3()
    {
        // get proxy objekt
        RemotingConfiguration.RegisterWellKnownClientType(typeof(Calc),
"tcp://localhost:50000/Calc");
```

```
        Calc clc = new Calc();
        // kald metoder for at teste funktionalitet
        Console.WriteLine(clc.add(2, 3));
        Console.WriteLine(clc.mul(2, 3));
        // kald metode som viser at der kun instatieres et objekt som deles af
alle kald
        Console.WriteLine(clc.ID);
    }
    public static void Main(string[] args)
    {
        // brug TCP/IP
        ChannelServices.RegisterChannel(new TcpClientChannel());
        // test 3 måder
        test1();
        test2();
        test3();
    }
}
```

build

```
csc /optimize+ /t:library /out:Calc.dll Calc.cs
csc /optimize+ /t:exe /r:Calc.dll /out:Server2.exe Server2.cs
csc /optimize+ /t:exe /r:Calc.dll /out:Client2.exe Client2.cs
```

CAO:

Server3.cs

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;

// server klasse
class Server3
{
    public static void Main(string[] args)
    {
        // registrere channel
        ChannelServices.RegisterChannel(new TcpServerChannel(50000));
        // registrere type under navn
        RemotingConfiguration.ApplicationName = "Calc";
        RemotingConfiguration.RegisterActivatedServiceType(typeof(Calc));
        Console.Write("Press enter to exit");
        Console.ReadLine();
    }
}
```

Client3.cs

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
using System.Runtime.Remoting.Activation;

// test client klasse
class Client3
{
    // måde 1 til at kalde remote server
    private static void test1()
    {
        // get proxy objekt
        Calc clc = (Calc)Activator.CreateInstance(typeof(Calc), null, new
object[] { new UrlAttribute("tcp://localhost:50000/Calc") });
        // kald metoder for at teste funktionalitet
        Console.WriteLine(clc.add(2, 3));
        Console.WriteLine(clc.mul(2, 3));
        // kald metode som viser at der instatieres et server objekt for hver
client objekt
        Console.WriteLine(clc.ID);
    }
    // måde 2 til at kalde remote server
    private static void test2()
    {
        // get proxy objekt
        RemotingConfiguration.RegisterActivatedClientType(typeof(Calc),
"tcp://localhost:50000/Calc");
        Calc clc = new Calc();
        // kald metoder for at teste funktionalitet
        Console.WriteLine(clc.add(2, 3));
        Console.WriteLine(clc.mul(2, 3));
        // kald metode som viser at der instatieres et server objekt for hver
client objekt
        Console.WriteLine(clc.ID);
    }
    public static void Main(string[] args)
    {
        // brug TCP/IP
        ChannelServices.RegisterChannel(new TcpClientChannel());
        // test 2 måder
        test1();
        test2();
    }
}
```

build

```
csc /optimize+ /t:library /out:Calc.dll Calc.cs
csc /optimize+ /t:exe /r:Calc.dll /out:Server3.exe Server3.cs
csc /optimize+ /t:exe /r:Calc.dll /out:Client3.exe Client3.cs
```

Avanceret eksempel
Vi laver nu et eksempel med:
- SAO Singleton
- adskillelse af interface og implementation
- 2 forskellige implementationer
- configurations fil på server side
- brug af namespaces

X.cs

```
namespace X.Common {
    // interface
    public interface IX
    {
        string WhoAmI();
    }
}
```

XImpl.cs

```
using System;

using X.Common;

namespace X.Impl {
    // implementation
    // skal extende MarshalByRefObject og interfacet
    public class XRealA : MarshalByRefObject, IX {
        public string WhoAmI()
        {
            return "I am A";
        }
    }
    // implementation
    // skal extende MarshalByRefObject og interfacet
    public class XRealB : MarshalByRefObject, IX {
        public string WhoAmI()
        {
            return "I am B";
        }
    }
}
```

XServer.cs

```
using System;
using System.Runtime.Remoting;

namespace X.Server {
```

```
    // server klasse
    class XServer
    {
        public static void Main(string[] args)
        {
            // konfigurer
            RemotingConfiguration.Configure("XServer.exe.config");
            Console.Write("Press enter to exit");
            Console.ReadLine();
        }
    }
}
```

XServer.exe.config

```
<configuration>
   <system.runtime.remoting>
      <application name="XServer">
         <service>
            <wellknown mode="Singleton" type="X.Impl.XRealA, XImpl"
objectUri="A"/>
            <wellknown mode="Singleton" type="X.Impl.XRealB, XImpl"
objectUri="B"/>
         </service>
         <channels>
            <channel port="50000" ref="tcp"/>
         </channels>
      </application>
   </system.runtime.remoting>
</configuration>
```

XClient.cs

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;

using X.Common;

namespace X.Client {
    // test client klasse
    class XClient
    {
        public static void Main(string[] args)
        {
            // test
            ChannelServices.RegisterChannel(new TcpClientChannel());
            IX a = (IX)Activator.GetObject(typeof(IX),
"tcp://localhost:50000/A");
```

```
            Console.WriteLine(a.WhoAmI());
            IX b = (IX)Activator.GetObject(typeof(IX),
"tcp://localhost:50000/B");
            Console.WriteLine(b.WhoAmI());
        }
    }
}
```

build

```
csc /optimize+ /t:library /out:X.dll X.cs
csc /optimize+ /t:library /r:X.dll /out:XImpl.dll XImpl.cs
csc /optimize+ /t:exe /out:XServer.exe XServer.cs
csc /optimize+ /t:exe /r:X.dll /out:XClient.exe XClient.cs
```

Videre
.NET frameworket indeholder et meget sofistikeret framework
af channels, sinks og formatters som gøre det muligt at
bruge remoting på næsten uendeligt mange måder.

Kommentar af cogitans d. 10. Jun 2007 | 1

Hvis koden kopieres direkte, kan man så forvente et tilfredsstillende kørsel?
F.eks. "XServer.exe.config" i linien
RemotingConfiguration.Configure("XServer.exe.config");
hvor skal den findes? Altså hvor skal man manuelt oprette den fil?

Kommentar af danielhep d. 25. May 2004 | 2

fino fino

Kommentar af j_jorgensen d. 06. Jan 2005 | 3

Manglere meget beskrivende tekst. Dog ganske rart med eksempler i massevis!

Kommentar af nielle d. 02. Jul 2007 | 4

Udemærket artikel, men den kunne passende udvides med en lille begrundelse for hvornår man bør vælge SAO SingleCall, SAO Singleton eller CAO respektiv.