Introduction til .NET remoting i VB.NET

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

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

Skrevet den **03. Feb 2009** af **arne_v** I kategorien **Programmering / Visual Basic .NET** | ★★☆☆☆

Historie:
V1.0 - 08/02/2004 - original
V1.1 - 09/02/2004 - forbedret formatering
V1.2 - 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.vb (fælles for alle 3 eksempler)

```
Imports System

Public Class Calc
Inherits MarshalByRefObject
    Private no As Integer
    Private Shared total As Integer = 0
    Private sync As Object = New Object

    Public Sub New()
        SyncLock sync
            total = total + 1
            no = total
        End SyncLock
    End Sub

    Public Function add(ByVal a As Integer, ByVal b As Integer) As Integer
        Return (a + b)
    End Function

    Public Function mul(ByVal a As Integer, ByVal b As Integer) As Integer
        Return (a * b)
    End Function

    Public ReadOnly Property ID() As String
        Get
            Return (no & " of " & total)
        End Get
    End Property
End Class
```

SAO Singlecall:

Server1.vb

```
Imports System
Imports System.Runtime.Remoting
Imports System.Runtime.Remoting.Channels
Imports System.Runtime.Remoting.Channels.Tcp
```

```
Class Server1

    Public Shared Sub Main(ByVal args As String())
        ChannelServices.RegisterChannel(New TcpServerChannel (50000))
        RemotingConfiguration.RegisterWellKnownServiceType(GetType(Calc),
"Calc", WellKnownObjectMode.SingleCall)
        Console.Write("Press enter to exit")
        Console.ReadLine
    End Sub
End Class
```

Client1.vb

```
Imports System
Imports System.Runtime.Remoting
Imports System.Runtime.Remoting.Channels
Imports System.Runtime.Remoting.Channels.Tcp

Class Client1

    Private Shared Sub test1()
        Dim clc As Calc = CType(Activator.GetObject(GetType(Calc),
"tcp://localhost:50000/Calc"), Calc)
        Console.WriteLine(clc.add(2, 3))
        Console.WriteLine(clc.mul(2, 3))
        Console.WriteLine(clc.ID)
    End Sub

    Private Shared Sub test2()
        Dim clc As Calc = CType(RemotingServices.Connect(GetType(Calc),
"tcp://localhost:50000/Calc"), Calc)
        Console.WriteLine(clc.add(2, 3))
        Console.WriteLine(clc.mul(2, 3))
        Console.WriteLine(clc.ID)
    End Sub

    Private Shared Sub test3()
        RemotingConfiguration.RegisterWellKnownClientType(GetType(Calc),
"tcp://localhost:50000/Calc")
        Dim clc As Calc = New Calc
        Console.WriteLine(clc.add(2, 3))
        Console.WriteLine(clc.mul(2, 3))
        Console.WriteLine(clc.ID)
    End Sub

    Public Shared Sub Main(ByVal args As String())
        ChannelServices.RegisterChannel(New TcpClientChannel )
        test1
        test2
        test3
    End Sub
End Class
```

build

```
vbc /optimize+ /t:library /out:Calc.dll Calc.vb
vbc /optimize+ /t:exe /r:Calc.dll /r:System.Runtime.Remoting.dll
/out:Server1.exe Server1.vb
vbc /optimize+ /t:exe /r:Calc.dll /r:System.Runtime.Remoting.dll
/out:Client1.exe Client1.vb
```

SAO Singleton:

Server2.vb

```
Imports System
Imports System.Runtime.Remoting
Imports System.Runtime.Remoting.Channels
Imports System.Runtime.Remoting.Channels.Tcp

Class Server2

    Public Shared Sub Main(ByVal args As String())
        ChannelServices.RegisterChannel(New TcpServerChannel (50000))
        RemotingConfiguration.RegisterWellKnownServiceType(GetType(Calc),
"Calc", WellKnownObjectMode.Singleton)
        Console.Write("Press enter to exit")
        Console.ReadLine
    End Sub
End Class
```

Client2.vb

```
Imports System
Imports System.Runtime.Remoting
Imports System.Runtime.Remoting.Channels
Imports System.Runtime.Remoting.Channels.Tcp

Class Client2

    Private Shared Sub test1()
        Dim clc As Calc = CType(Activator.GetObject(GetType(Calc),
"tcp://localhost:50000/Calc"), Calc)
        Console.WriteLine(clc.add(2, 3))
        Console.WriteLine(clc.mul(2, 3))
        Console.WriteLine(clc.ID)
    End Sub

    Private Shared Sub test2()
        Dim clc As Calc = CType(RemotingServices.Connect(GetType(Calc),
"tcp://localhost:50000/Calc"), Calc)
        Console.WriteLine(clc.add(2, 3))
        Console.WriteLine(clc.mul(2, 3))
        Console.WriteLine(clc.ID)
    End Sub
```

```
    Private Shared Sub test3()
        RemotingConfiguration.RegisterWellKnownClientType(GetType(Calc),
"tcp://localhost:50000/Calc")
        Dim clc As Calc = New Calc
        Console.WriteLine(clc.add(2, 3))
        Console.WriteLine(clc.mul(2, 3))
        Console.WriteLine(clc.ID)
    End Sub

    Public Shared Sub Main(ByVal args As String())
        ChannelServices.RegisterChannel(New TcpClientChannel )
        test1
        test2
        test3
    End Sub
End Class
```

build

```
vbc /optimize+ /t:library /out:Calc.dll Calc.vb
vbc /optimize+ /t:exe /r:Calc.dll /r:System.Runtime.Remoting.dll
/out:Server2.exe Server2.vb
vbc /optimize+ /t:exe /r:Calc.dll /r:System.Runtime.Remoting.dll
/out:Client2.exe Client2.vb
```

CAO:

Server3.vb

```

Imports System
Imports System.Runtime.Remoting
Imports System.Runtime.Remoting.Channels
Imports System.Runtime.Remoting.Channels.Tcp

Class Server3

    Public Shared Sub Main(ByVal args As String())
        ChannelServices.RegisterChannel(New TcpServerChannel (50000))
        RemotingConfiguration.ApplicationName = "Calc"
        RemotingConfiguration.RegisterActivatedServiceType(GetType(Calc))
        Console.Write("Press enter to exit")
        Console.ReadLine
    End Sub
End Class
```

Client3.vb

```

Imports System
Imports System.Runtime.Remoting
Imports System.Runtime.Remoting.Channels
Imports System.Runtime.Remoting.Channels.Tcp
```

```
Imports System.Runtime.Remoting.Activation

Class Client3

    Private Shared Sub test1()
        Dim clc As Calc = CType(Activator.CreateInstance(GetType(Calc),
Nothing, New Object()  {New UrlAttribute ("tcp://localhost:50000/Calc")}),
Calc)
        Console.WriteLine(clc.add(2, 3))
        Console.WriteLine(clc.mul(2, 3))
        Console.WriteLine(clc.ID)
    End Sub

    Private Shared Sub test2()
        RemotingConfiguration.RegisterActivatedClientType(GetType(Calc),
"tcp://localhost:50000/Calc")
        Dim clc As Calc = New Calc
        Console.WriteLine(clc.add(2, 3))
        Console.WriteLine(clc.mul(2, 3))
        Console.WriteLine(clc.ID)
    End Sub

    Public Shared Sub Main(ByVal args As String())
        ChannelServices.RegisterChannel(New TcpClientChannel )
        test1
        test2
    End Sub
End Class
```

build

```
vbc /optimize+ /t:library /out:Calc.dll Calc.vb
vbc /optimize+ /t:exe /r:Calc.dll /r:System.Runtime.Remoting.dll
/out:Server3.exe Server3.vb
vbc /optimize+ /t:exe /r:Calc.dll /r:System.Runtime.Remoting.dll
/out:Client3.exe Client3.vb
```

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.vb

```

Namespace X.Common
    ' interface
    Public Interface IX
        Function WhoAmI() As String
```

```
    End Interface
End Namespace
```

XImpl.vb

```
Imports System

Imports X.Common

Namespace X.Impl
    ' implementation
    ' skal extende MarshalByRefObject og interfacet
    Public Class XRealA
    Inherits MarshalByRefObject
    Implements IX
        Public Function WhoAmI() As String Implements IX.WhoAmI
            Return "I am A"
        End Function
    End Class
    ' implementation
    ' skal extende MarshalByRefObject og interfacet
    Public Class XRealB
    Inherits MarshalByRefObject
    Implements IX
        Public Function WhoAmI() As String Implements IX.WhoAmI
            Return "I am B"
        End Function
    End Class
End Namespace
```

XServer.vb

```
Imports System

Imports System.Runtime.Remoting

Namespace X.Server
    ' server klasse
    Class XServer
        Public Shared Sub Main(ByVal args As String())
            ' konfigurer
            RemotingConfiguration.Configure("XServer.exe.config")
            Console.Write("Press enter to exit")
            Console.ReadLine
        End Sub
    End Class
End Namespace
```

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.vb

```
Imports System
Imports System.Runtime.Remoting
Imports System.Runtime.Remoting.Channels
Imports System.Runtime.Remoting.Channels.Tcp

Imports X.Common

Namespace X.Client
    ' test client klasse
    Class XClient
        Public Shared Sub Main(ByVal args As String())
            ' test
            ChannelServices.RegisterChannel(New TcpClientChannel )
            Dim a As IX = CType(Activator.GetObject(GetType(IX),
"tcp://localhost:50000/A"), IX)
            Console.WriteLine(a.WhoAmI)
            Dim b As IX = CType(Activator.GetObject(GetType(IX),
"tcp://localhost:50000/B"), IX)
            Console.WriteLine(b.WhoAmI)
        End Sub
    End Class
End Namespace
```

build

```
vbc /optimize+ /t:library /out:X.dll X.vb
vbc /optimize+ /t:library /r:X.dll /out:XImpl.dll XImpl.vb
vbc /optimize+ /t:exe /out:XServer.exe XServer.vb
vbc /optimize+ /t:exe /r:System.Runtime.Remoting.dll /r:X.dll /out:XClient.exe
XClient.vb
```

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 kichian d. 09. Feb 2004 | 1

Hmm. En hurtig intro til emnet. Men hvorfor benyttes der ikke interfaces i mindst et eksempel for at vise uafhængigheden imellem klienten og serverimplementation.

Kommentar af wired d. 28. Jul 2004 | 2

Savner også en UI eksempel.