.NET Remoting

.NET Remoting infrastruktūra ir servisu kopa, kas atbild par attālinātu objektu aktivizāciju, vada objektu dzīves laiku, nodrošina apmaiņu ar paziņojumiem caur kanāliem.

Kanāli (communication channels) – ir objekti, kas atbild par paziņojumu transportēšanu starp attālinātiem objektiem. Jebkurš paziņojums, kurš tiek nosūtīts caur kanālu, tiek kodēts un dekodēts izmantojot iekšējus .NET Remoting objektus, kuri tiek saukti par formātētājiem un kas "saprot" noteiktus formātus, tādus, kā binārais un SOAP.

Eksistē divi kanālu tipi: TCP kanāls un HTTP kanāls.

TCP kanāls pārsūta datus starp klientu un serveri izmantojot bināru formātu. TCP kanāls ir divpusējs (datus var sūtīt un saņemt pa šo kanālu). Tas ir visātrdarbīgākais kanāls.

HTTP kanāls sūta datus izmantojot HTTP protokolu. Dati tiek sūtīti SOAP formātā.

Datu sūtīšana no viena datora citam datoram notiek caur portiem. .NET Remoting tehnoloģijā programmētājs var pats izvēlēties, kādu portu izmantos kanāls. Pēc noklusēšanas HTTP kanāls izmanto 80 portu pieprasījumu sūtīšanai. Ja Jūs paši gribat izvēlēties portu, tad nav ieteicams izmantot portu, kuras numurs ir mazāks par 1000 (var būt rezervēts). Ja Jūs izmantojat operētājsistēmu Windows 2000/NT/XP, tad noteikt, kādi porti tiek lietoti Jūsu datorā, Jūs varat atvērot Notepadā failu c:\Windows\System32\Drivers\Etc\Services.

Pirms paziņojuma sūtīšanas attālinātam objektam, kanāls ir jāiereģistrē Remoting vidē (kanāls ir jāiereģistrē gan servera objektam, gan klienta objektam).

Lai iereģistrētu kanālu, var izmantot klases `ChannelServices` statisku metodi `RegisterChannel`.

Piemēram, lai iereģistrētu HTTP kanālu, kas izmanto portu 8080, var izmantot sekojošas rindas:

VB.NET
```
Dim channel As New HttpChannel(8080)
ChannelServices.RegisterChannel(channel)
```

C#
```
HttpChannel channel = new HttpChannel(8080);
ChannelServices.RegisterChannel(channel);
```

Līdzīgi var iereģistrēt TCP kanālu:

VB.NET
```
Dim channel As New TcpChannel(8080)
ChannelServices.RegisterChannel(channel)
```

C#
```
TcpChannel channel = new TcpChannel(8080);
ChannelServices.RegisterChannel(channel);
```

Dalīta lietojumā visus objektus var sadalīt uz attālinātiem (remootable) un lokāliem (nonremootable). Lokāli objekti ir pieejami tikai tekošas programmas domēnā (procesā). Pie attālinātiem objektiem var piekļūt no citiem domēniem (programmu processiem). Attālinātus objektus var sadalīt divās grupās:

- objekti, kas tiek nodoti pēc vērtības (marshal-by-value) - tiek kopēti klientā lietojumā
- objekti, kas tiek nodoti klienta lietojumam pēc norādes (marshal-by-reference) - lai piekļūtu pie tāda objekta, ir vajadzīgs objekts proksi.

Objekti, kas tiek nodoti pēc vērtības, tiek pārsūtīti citas programmas domēnā ar serializācijas palīdzību. Serializācija tā ir tekošā objekta stāvokļa saglabāšana bitu secībā. Lai Remoting vide varētu automātiski serializēt objektus, ir nepieciešams realizēt tajos interfeisu `ISerializable` vai atzīmēt atbilstošu klasi ar atribūtu `Serializable`. Piemēram:

VB.NET
```
<Serializable()> _
Public Class Some Class
      ...
End Class
```

C#.NET
```
[Serializable]
public class SomeClass
{
      ...
}
```

Gadījumā, kad ir nepieciešams, lai klients izsauktu objektu, kas atrodas konkrētā domēnā, nevis izsauktu objekta kopiju, kas atrodas klientā domēnā, ir jāizmanto objektus, kas tiek nodoti pēc norādes. Lai kādu objektu būtu iespējams nodod pēc norādes, šo objektu ir jāapraksta kā klases `System.MarshalByRefObject` pēcteci. Piemēram:

VB.NET
```
Public Class SomeClass
      Inherits MarshalByRefObject
...
End Class
```

C#
```
class SomeClass : MarshalByRefObject
{
      ...
}
```

Kad klienta puses objekts izsauc servera objektu, kurš tiek nodots pēc norādes, tad servera objektu ir jāaktivizē, lai tas varētu atbildēt uz pieprasījumiem (objektus, kas tiek nodoti pēc vērtības, nevajag aktivizēt, tāpēc ka tie tiek kopēti serializācijas procesā un tiek faktiski aktivizēti deserializācijas procesā). Servera objekta palaišanu sauc par aktivizāciju (activation), bet laika intervālu, kurā servera objekts paliek aktīvs, par dzīves laiku (life time). Tehnoloģija Remoting piedāvā sekojošas aktivizācijas metodes, katrai no kurām atbilst savs dzīves laiks:

- Single call
- Singleton
- Client Activated objects (CAO – objekti, kurus aktivizē klients)

Ja objektam ir aktivizācijas veids single call, tad katram klienta pieprasījumam tiek aktivizēta jauna servera objekta kopija. Single call tipa objekts "dzīvo" tik daudz, cik ir nepieciešams, lai apstrādātu vienu klienta pieprasījumu.

Ja servera objekts izmanto aktivizācijas tipu singleton, tad viens šī objekta eksemplārs apkalpo visus klientu pieprasījumus. Objektam singleton ir ierobežots dzīves laiks (katrs objekts saņem pieprasīto dzīves laiku, bet var piešķirt mūžīgu dzīves laiku). Kad šis laiks beidzies, tekošais objekta eksemplārs tiks iznīcināts. Gadījumā, ja pēc servera objekta iznīcināšanas parādīsies jauns klienta pieprasījums, tiks izveidots jauns servera objekta eksemplārs.

CAO tipa objektus aktivizē un iznicina klienti. Katrs CAO objekts satur norādi uz atbilstošu klientu. Katram klientam tiks veidots savs CAO objekts. CAO objekts kalpo, kamēr nebeidzas tā rente (lease). Kad rente beidzas, klienta objekts var pagarināt renti. Ir iespējams objektam dod arī mūžīgu renti. Mūžīgi objekti ar klienta aktivizāciju atšķiras no mūžīgiem objektiem ar servera aktivizāciju ar to, ka pirmie apkalpo tikai vienu klientu, bet otrie – visus klientus.

CAO tipa objekti (objekti ar klienta aktivizāciju) atšķiras no objektiem ar servera aktivizāciju (single call un singleton) ar to, ka tiek veidoti brīdī, kad klients veido tā eksemplāru ar operatoru `new`. Objekti ar servera aktivizāciju tiek veidoti, tad, kad klients izsauc kādu no tā metodēm.

Vēl viena atšķirība starp aktivizācijas veidiem ir saistīta ar to, ka objekti tipa single call nevar saglabāt savu stāvokli (savu īpašību vērtības) starp izsaukumiem. Tādus objektus sauc par stateless objektiem (objektiem bez stāvokļa). CAO tipa objekti var saglabāt savu stāvokli starp izsaukumiem (tie eksistē kamēr klients tos neiznīcinās). Tādus objektus sauc par statefull objektiem (objektiem ar stāvokli). Singleton tipa objekti var saglabāt savu stāvokli, bet tikai līdz sava dzīves laika beigām.

Piezīme:

Aktivizācijas režīmu nosaka ne objekta tips, bet infrastruktūras konfigurācija. Viens un tās pats tips vienā lietojumā var būt aktivizēts izmantojot servera aktivizāciju, bet otrā – klienta aktivizāciju.

Lai nokonfigurētu programmas izpildes laikā `SomeMBRType` tipa objektu kā singleton objektu, servera puses programmā var izmantot sekojošu operatoru:

VB.NET

```
RemotingConfiguration.RegisterWellKnownServiceType( _
      GetType( SomeMBRType ), _
      "SomeURI", _
      WellKnownObjectMode.Singleton )
```

C#

```
RemotingConfiguration.RegisterWellKnownServiceType(
      typeof( SomeMBRType ),
      "SomeURI",
      WellKnownObjectMode.Singleton );
```

Piezīme: Kā URI var norādīt kādu unikālu identifikatoru, piemēram programmas vārdu.

Klientam arī ir nepieciešams nokonfigurēt `SomeMBRType` kā singleton objektu sekojošā veidā:

VB.NET

```
RemotingConfiguration.RegisterWellKnownClientType( _
      GetType( SomeMBRType ), _
      "http://SomeWellKnownURL/SomeURI" )
```

C#
```
RemotingConfiguration.RegisterWellKnownClientType(
      typeof( SomeMBRType ),
      "http://SomeWellKnownURL/SomeURI" );
```

Lai servera programma nokonfigurētu objektu kā single call objektu, ir jāievieto sekojošas rindas:

VB.NET
```
RemotingConfiguration.RegisterWellKnownServiceType( _
      GetType( SomeMBRType ), _
      "SomeURI", _
      WellKnownObjectMode.SingleCall )
```

C#
```
RemotingConfiguration.RegisterWellKnownServiceType(
      typeof( SomeMBRType ),
      "SomeURI",
      WellKnownObjectMode.SingleCall );
```

Klienta puses programmā single call objektu ir jākonfigurē tieši tāpat, kā singleton objektu.

CAO tipa objektu servera programmā ir jākonfigurē sekojošā veidā:

VB.NET
```
RemotingConfiguration.RegisterActivatedServiceType( _
      GetType( SomeMBRType ))
```

C#.NET
```
RemotingConfiguration.RegisterActivatedServiceType(
      typeof( SomeMBRType ));
```

Bet klienta programmā:

VB.NET
```
RemotingConfiguration.RegisterActivatedClientType( _
      GetType( SomeMBRType ), _
      "http://SomeURL" )
```

C#.NET
```
RemotingConfiguration.RegisterActivatedClientType(
      typeof( SomeMBRType ),
      "http://SomeURL" );
```

Piezīme: kā URL var pierakstīt rindu, līdzīgu sekojošai: "http://localhost:4000" – šeit 4000 ir porta numurs.

Objektiem, kas tiek nodoti pēc norādes, var norādīt dzīves laiku (single call objekti neatkarīgi no dzīves laika apkalpo tikai vienu lietotāja pieprasījumu. Kad parādās jauns pieprasījums, tad neatkarīgi no tā, vai vēl eksistē iepriekšējais single call objekts vai nē, tiks veidots jauns objekta eksemplārs, bet

iepriekšējais tiks iznīcināts). Pēc dzīves laika beigšanas objekti tiek atzīmēti iznicināšanai un drazu savākšanas procesā atzīmētie objekti tiks iznīcināti.

Rentes dispečeri (lease manager), kas atrodas katra pielikuma domēnā, meklē gatavus iznīcināšanai objektus savā domēnā, un atzīmē tos. Bet objekts sponsors (sponsor), ja tas ir iereģistrēts dispečerā, var pagarināt renti. Kad rentes ilgums beidzies, tad dispečers iztaujā sponsorus, vai viņi grib pagarināt renti vai nē, un, ja neviens no sponsoriem renti nepagarina, dispečers atzīmē objektu kā gatavu iznīcināšanai un drazu savācējs atbrīvo atmiņu, kuru aizņem objekts.

Objekta rentes inicializācijai ir nepieciešams pārdefinēt klases `MarshalByRefObjekt` funkciju `InitializeLifetimeService`. Atbilstošais kods var izskatīties sekojošā veidā:

VB.NET

```
Imports System.Runtime.Remoting.Lifetime

Public Class SomeMBRType
      Inherits MarshalByRefObject
      Public Overrides Function InitializeLifetimeService() As Object
            Dim lease As ILease = CType(MyBase.InitializeLifetimeService(), Ilease)
            If lease.CurrentState == LeaseState.Initial Then
                  lease.InitialLeaseTime = TimeSpan.FromMinutes(2)
                  lease.SponsorshipTimeout = TimeSpan.FromMinutes(3)
                  lease.RenewOnCallTime = TimeSpan.FromSeconds(3)
            End If
            Return lease;
      End Function
End Class
```

C#.NET

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

public class SomeMBRType : MarshalByRefObject
{
      public override object InitializeLifetimeService()
      {
            ILease lease = (ILease) base.InitializeLifetimeService();
            if (lease.CurrentState == LeaseState.Initial)
            {
                  lease.InitialLeaseTime = TimeSpan.FromMinutes(2);
                  lease.SponsorshipTimeout = TimeSpan.FromMinutes(3);
                  lease.RenewOnCallTime = TimeSpan.FromSeconds(3);
            }
            return lease;
      }
}
```

Ja ir nepieciešams, lai objektam būtu piešķirts mūžīgs dzīves laiks, tad funkcijai `InitializeLifetimeService` ir jādod atpakaļ vērtību `null` (C#.NET) vai `Nothing` (VB.NET).

Dotā piemērā īpašība `InitialLeaseTime` nosaka sākotnēju rentes garumu. Īpašība `RenewOnCallTime` tiek lietota, lai pagarinātu renti gadījumā, ja kāds no klientiem izsauc atbilstošā objekta metodi (rente tiks pagarināta tikai gadījumā, ja rentes ilgums ir mazāks nekā `RenewOnCallTime`). Īpašība `SponsorshipTimeout` nosaka, cik ilgi ir jāgaida atbilde no objekta sponsoriem.

Interfeisā ILease ir aprakstīta arī ceturtā īpašība `CurrentLeaseTime`, kas nosaka laika intervālu līdz rentes beigām. Šī īpašība ir pieejama tikai lasīšanai.

Lai izstrādātu objekta sponsoru, tas ir jāapraksta kā klases `MarshalByRefObject` pēctecis un jārealizē tajā interfeisu `ISponsor`. Izstrādātāju ērtībai .NET Remoting satur klasi `ClientSponsor`, kas dod iespēju reģistrēt norādes uz attālinātiem objektiem, kuru dzīves laiku ir nepieciešams vadīt. Laika intervālu, uz kuru sponsors pagarinās renti, nosaka īpašība `ClientSponsor.RenewalTime`. Piemēram, klasi `ClientSponsor` var izmantot sekojošā veidā:

VB.NET
```
Dim cp As New ClientSponsor(TimeSpan.FromMinutes(5))
cp.Register(someMBR)
```

C#.NET
```
ClientSponsor cp = new ClientSponsor(TimeSpan.FromMinutes(5));
cp.Register(someMBR);
```

Šeit `someMBR` ir norāde uz objektu, kuru renti ir jāpagarina sponsoram.

Klienta programma arī var pagarināt renti izmantojot metodi `ILease.Renew`. Piemēram:

VB.NET
```
Dim obj As New someMBR()
Dim lease As ILease = CType(RemotingServices.GetLifetimeService(obj), ILease)
Dim expireTime As TimeSpan = lease.Renew(TimeSpan.FromSeconds(30))
```

C#.NET
```
someMBR obj = new someMBR();
ILease lease = (ILease)RemotingServices.GetLifetimeService(obj);
TimeSpan expireTime = lease.Renew(TimeSpan.FromSeconds(30));
```

Piezīme:

Lai izmantotu iepriekšminētās funkcijas, programmā ir nepieciešams pievienot programmai sekojošas vārdu telpas:

VB.NET
```
Imports System
Imports System.Runtime.Remoting
Imports System.Runtime.Remoting.Channels
Imports System.Runtime.Remoting.Channels.Tcp
Imports System.Runtime.Remoting.Lifetime
```

C#.NET
```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
using System.Runtime.Remoting.Lifetime;
```

Nokonfigurēt servera objektu var ne tikai programmas izpildes laikā (izmantojot aprakstītās funkcijas), bet arī izmantojot konfigurācijas failus (skaties laboratorijas darbu "**.**NET Remoting izmantošana"). Ārējo objektu konfigurācijas parametri var būt norādīti failā Machine.config (kuru .NET Framework apstrādā automātiksi) vai pielikuma konfigurācijas failā, kuru var ielādēt ar operatoru

`RemotingConfiguration.Configure(faila_vārds)`. Konfigurācijas fails var saturēt sekojošus atribūtus:

Elements	Apraksts
`<application>`	Satur informāciju par attālinātiem objektiem, kurus eksportē vai izmanto programma
`<service>`	Satur objektus, kurus programma eksportē. Elements `<application>` var saturēt vienu vai vairākus elementus `<service>`
`<client>`	Satur objektus, kurus programma importē. Elements `<application>` var saturēt vienu vai vairākus elementus `<client>`
`<wellknown>`	Satur informāciju par objektiem ar servera aktivizāciju, kurus programma eksportē. Elements `<service>` var saturēt vienu vai vairākus elementus `<wellknown>`. Elementam `<wellknown>` ir trīs obligāti atribūti: mode, `type` un `objectUri`. Atribūtam mode ir jāpiešķir vērtība `Singleton` vai `SingleCall`; atribūts `type` nosaka klases un salikuma vārdu, kas satur klases kodu; atribūts `ObjectUri` nosaka objekta URI.
`<activated>`	Satur informāciju par objektiem ar klienta aktivizāciju, kurus eksportē programma. Elements `<service>` var saturēt vienu vai vairākus elementus `<activated>`. Dotajam elementam ir vienīgais atribūts `type`, kas nosaka klases un salikuma vārdu, kas satur klases kodu

Pēc servera objekta konfigurācijas servera un klienta programmā, klienta programmā ar operatoru `new` var iegūt norādi uz attālinātu objektu (izveidot attālināta objekta proxy eksemplāru (proxy ir nepieciešams tikai objektiem, kuri tiek nodoti pēc norādes)), ar kuras palīdzību būs iespējams izsaukt attālināta objekta metodes, tāpat kā lokāla objekta metodes.
Attālināta objekta tipa reģistrācija ir vajadzīga ne vienmēr, bet tikai tad, kad Jūs vēlaties izmantot vārdu `new` attālināta objekta norādes izveidošanai. Cita iespēja iegūt norādi uz attālinātu objektu (izveidot proxy) – izmantot metodi `Activator.GetObject`. Piemēram:

VB.NET
```
Dim channel As New TcpChannel()
ChannelServices.RegisterChannel(channel)
Dim h As someMBR
h = CType(Activator.GetObject(GetType(someMBR), _
      "tcp://127.0.0.1:4000/someURI"), someMBR)
h.Method1()
```

C#.NET
```
TcpChannel channel = new TcpChannel();
ChannelServices.RegisterChannel(channel);
someMBR h = (someMBR) Activator.GetObject(
            typeof(someMBR),
            "tcp://127.0.0.1:4000/someURI");
h.SayHello();
```

Pēc proxy eksemplāra izveidošanas klients var izsaukt servera objekta metodes (skatieties faila "piemeri.NET.doc" VI. piemēru). Datu apmaiņai starp klientu un serveri var izmantot arī notikumus (skaties laboratorijas darbu ".NET Remoting izmantošana"). Papildinformāciju par notikumu izmantošanu programmās var atrast failos C#.dos (valodai C#) un VB.NET.doc (valodai VB.NET).

Piezīme:

Piekļūt pie attālināta objekta statiskiem (static vai shared) elementiem nav iespējams

Notikumu izmantošanas gadījumā serveris saņem iespēju izsaukt klienta metodes (notikumu, kuru izraisa serveris, apstrādes funkcijas atrodas klienta domēnā). Tādējādi, notikuma parādīšanās gadījumā klienta programma funkcionē kā serveris. Tāpēc laboratorijas darbā ".NET Remoting izmantošana" objekts ChatClient tika izveidots klases `MarshalByRefObject` bāzē, lai serveris varētu izsaukt tā metodes.

Parasti kodu, kas konfigurē attālinātu objektu, atdala no servera objekta realizācijas koda (skaties laboratorijas darbu "**.**NET Remoting izmantošana"), bet tas nav obligāti jādara (skaties faila "piemeri .NET.doc", VI. piemēru). Servera programmu var realizēt kā konsoli, WindowsForms vai Windows serviss lietojumu.