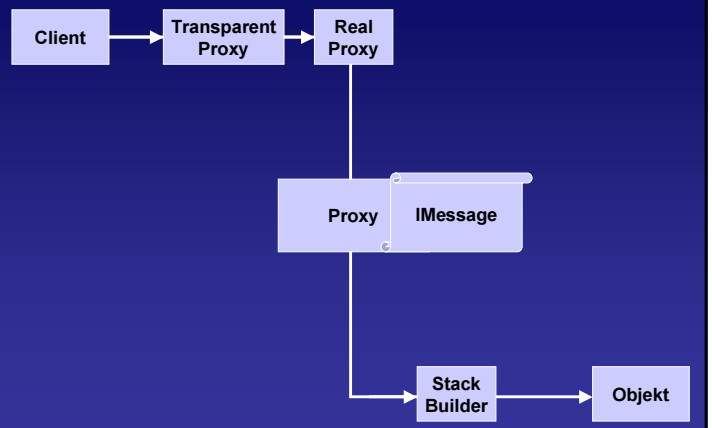
Softwareentwicklung mit MS.NET und C#

.NET Remoting

Ingo Rammer
http://www.DotNetRemoting.cc
rammer@sycom.at

Lokaler Proxy



Proxy

- DEMO

Remoting

- Remoting ist der Zugriff auf Objekte über bestimmte Grenzen hinweg
- Grenzen:
  - Maschine
  - Prozess
  - AppDomain
  - Context

Inhalt einer Message

- Interface: IMessage
  - Intern: Dictionary

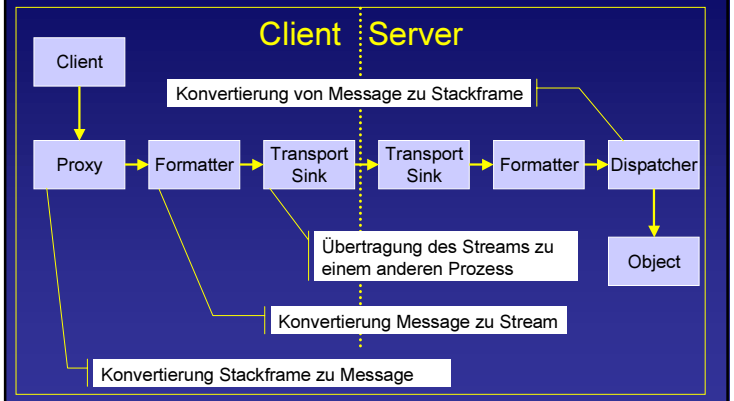
```
bool ret = mv.GetData("Hallo Welt", 4711);
```

__Uri	*Adresse des Zielobjektes*
__MethodName	GetData
__MethodSignature	{typeof(String), typeof(int)}
__TypeName	MagicValue, 01_LocalProxy
__Args	{"Hallo Welt", 4711}
__CallContext	*LogicalCallContext*

Elemente

- Message:
  - Was?
- Channel:
  - Womit?
  - Welches Protokoll?
- Formatter:
  - Wie?
  - Welches Format?

Remoting Architektur



.NET Remoting

- Verteilte Applikationen auf Basis der .NET Plattform
- Unterstützung aller CLR-Typen
  - Objekte, Strukturen, Vererbungshierarchie, Interfaces, ...
- Arbeitet unabhängig von Transfermedium und Kodierung
  - Transfer
    - HTTP
    - TCP
  - Kodierung
    - SOAP
    - Binary (proprietär)

Features von .NET Remoting

- Übergabe von Objektreferenzen und Objektkopien
  - `ObjRef`
- Distributed Object Lifetime Management
- Callbacks/Events
- Publikation vorhandener Objektinstanzen
- Encryption, Authentication & Authorization (in Kombination mit IIS)

Abgrenzung zu anderen Systemen

- Proxygenerierung
  - Transparent oder statisch?
- Datentypen
  - Automatische Serialisierung oder manuell?
  - Nur native Datentypen?
- Transportprotokoll und Wire-Format
  - Festgelegt oder frei?
- "Binding" zur Plattform
  - Retrofitted oder Bestandteil der Plattform?

Abgrenzung zu SOAP Web Services

- http://www.w3.org/TR/SOAP Sect. 1.1

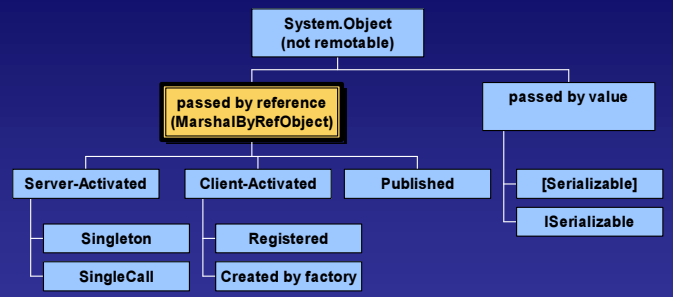
`[...] not part of the core SOAP specification [...]:`

- `Distributed garbage collection`
- `Objects-by-reference (which requires distributed garbage collection)`
- `Activation (which requires objects-by-reference)`

Zurück zum Thema

- Zurück zu Remoting ...

Object-type Roadmap



Server Activated Objects (SAO)

- Verhalten sich wie Web Services
  - Definierter SOAP-Endpunkt URL pro Service
- Entweder Singleton oder SingleCall
- Objektinstanzen werden bei Bedarf vom Server erzeugt
- Zugriff mittels `Activator.GetObject()`
- Multithreading!

Remote Objects

- Basisklasse `System.MarshalByRefObject`
- Sind an den Ursprungsort gebunden
- Werden *"by Reference"* übergeben
- Client arbeitet mit einem Proxy Objekt
- Jeder Methodenaufruf wird zum Server übertragen
  - Ausnahme: GetType() und GetHashCode()
  - Statische Methoden

Client Activated Objects (CAO)

- Verhalten sich wie "klassische" Objekte
- Können entweder per Factory-Pattern von einem SAO oder direkt (per transparenter Factory) erzeugt werden
- Objektreferenzen können über Remoting-Grenzen übergeben werden
- Lifetime-Managed!
- Transparente Aktivierung mittels `Activator.CreateInstance()`

Objektpublikation

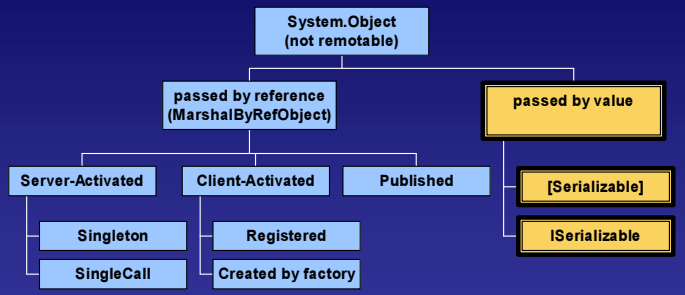
```
HttpChannel channel =
      new HttpChannel(1234);

ChannelServices.RegisterChannel(
      channel);

SomeObject obj = new SomeObject();

RemotingServices.Marshal(
       obj,"myobject.rem");
```

Object-type Roadmap



Implementation

- DEMO

Mobile Objekte

- Klassen, die mit [Serializable] markiert sind
- Bei Übergabe wird eine Kopie erstellt (byValue-Übergabe)
- Objektgraphen: Zirkuläre Referenzen werden erkannt und korrekt serialisiert
- Implementation muss beim Client verfügbar sein, oder mit `Assembly.LoadFrom()` explizit geladen werden!

[Serializable] vs Remote

- DEMO

Interface based Remoting

- DEMO

Metadata-Sharing

- Client benötigt Zugriff auf Metadaten
  - Klassen, Methodensignaturen, ...
- Möglichkeiten:
  - Shared Interfaces
  - Shared Base Classes
  - Shared Implementation (Client und Server verfügen über die Implementationsklassen)
  - SoapSuds.exe (Extraktion von Proxy-Klassen aus serverseitigen Assemblies oder direkt vom laufenden Server)

Lifetime Management

- Lease
  - Erstellung, sobald das Objekt angesprochen oder veröffentlicht wird
  - Eigenschaften
    - CurrentLeaseTime: Lebenszeit des Objektes
    - InitialLeaseTime: default 5 Minuten
    - RenewOnCallTime: default 2 Minuten
    - SponsorShipTimeout: default 2 Minuten
  - Time-To-Live wird standardmässig alle 10 Sekunden dekrementiert

Geänderte Lebenszeiten

- Überschreiben der Methode InitializeLifetimeService()

```
public override object InitializeLifetimeService()
{
  ILease lease =
      (ILease)base.InitializeLifetimeService();

  lease.InitialLeaseTime =
      TimeSpan.FromSeconds(10);
  lease.SponsorshipTimeout =
      TimeSpan.FromSeconds(10);
  lease.RenewOnCallTime =
      TimeSpan.FromSeconds(10);

  return lease;
}
```

Deployment

- jede „managed Application"
  - Console
  - Windows Forms
  - Windows Service
- Internet Information Server (IIS)
  - Authentication (HTTP Basic oder Windows integrated)
  - Encryption & Signing (SSL)
- Remote-Komponenten: COM+
  - erben von `ServicedComponent`

Sponsorship

- Für jede Lease können Sponsor-Objekte registriert werden (implementieren `ISponsor`)
- Sobald TTL abgelaufen ist → Framework kontaktiert Sponsoren
- Sponsor kann eine neue TTL bestimmen

```
public TimeSpan Renewal(ILease lease)
{
   return TimeSpan.FromMinutes(5);
}
```

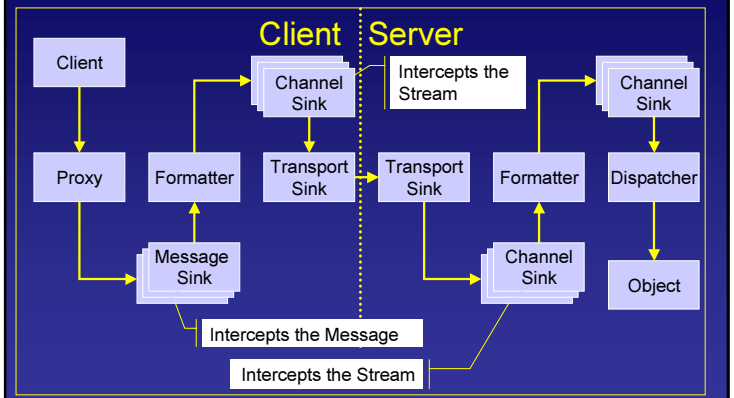
Konfigurationsdateien

- Standard .NET Konfigurationsdateien
- Section <System.Runtime.Remoting>
- Konfiguriert alle Aspekte von Remoting
  - Channels
  - Custom Sink Chains
  - Objekte
- Client- und serverseitig

Verwendung der Konfiguration

- DEMO

Erweiterbarkeit



Verwendung der Konfiguration

- **Server**

```
public static void Main(String[] args)
{
  RemotingConfiguration.Configure("<datei>");
  Console.ReadLine();
}
```

- **Client**

```
public static void Main(String[] args)
{
  RemotingConfiguration.Configure("<datei>");
  Foo obj = new Foo();
  int value = obj.GetValue();
}
```

Verhalten des new Operators wird geändert!

Erweiterbarkeit

- Debugging
- Logging
- Accounting
- Übergabe von "out-of-band" Informationen (Laufzeitparameter, z.B. Thread-Priorität)
- Security (IP-Blocking, Verschlüsselung)
- Komprimierung des Streams
- Caching

Erweiterbarkeit

- Neue Transferkanäle
  - Named Pipes (IPC)
  - MSMQ (Message Queuing)
  - SMTP/POP3 (Internet Email)
  - Jabber (instant messaging)
  - Mainframe-Integration (dateibasierend)
- Neue Channel/Formatter Kombinationen
  - IIOP Transport mit CDR Format. Anyone?

Downloads

- SMTPChannel, JabberChannel, BidirectionalTcpChannel
  http://www.dotnetremoting.cc/Projects
- NamedPipeChannel, SSLChannel
  http://www.gotdotnet.com + search
- MSMQ Channel
  http://www.codeproject.com/csharp/msmqchannel.asp

Summary

- .NET Remoting bietet eine Plattform für objektorientierte, verteilte Anwendungen auf Basis des .NET Frameworks
- Erlaubt Übergabe von Referenzen und Kopien
- .NET Remoting basiert auf Messages, die mittels beliebiger Kanäle übertragen werden
- Flexibel durch Mehrschicht-Architektur

Danke für Ihre Aufmerksamkeit!

Fragen?

http://www.DotNetRemoting.cc