Exemples Cours .Net Remoting

Extraits du livre « Advanced .Net Remoting, Ingo Rammer & Mario Szpuszta »

Un premier Exemple : Singleton

This sample remoting application exposes a server-side MarshalByRefObject in Singleton mode. You will call this object *CustomerManager*, and it will provide a method to load a *Customer* object (which is a ByValue object) from a fictitious database. The resulting object will then be passed as a copy to the client.

Defining the Remote Interface

As a first step, you have to define the interface ICustomerManager, which will be implemented by the server. In this interface, you'll define a single method, GetCustomer(), that returns a Customer object to the caller.

```
public interface ICustomerManager
{
Customer GetCustomer(int id);
}
```

Defining the Data Object

Because you want to provide access to customer data, you first need to create a Customer class that will hold this information. This object needs to be passed as a copy (by value), so you have to mark it with the [Serializable] attribute.
In addition to the three properties FirstName, LastName, and DateOfBirth, you will also add a method called GetAge() that will calculate a customer's age.

```
[Serializable]
public class Customer
{
public String FirstName;
public String LastName;
public DateTime DateOfBirth;
public Customer()
{
Console.WriteLine(Customer.constructor: Object created);
}
public int GetAge()
{
Console.WriteLine("Customer.GetAge(): Calculating age of {0}, " +
"born on {1}.",
FirstName,
DateOfBirth.ToShortDateString());
TimeSpan tmp = DateTime.Today.Subtract(DateOfBirth);
return tmp.Days / 365; // rough estimation
}
}
```

Implementing the Server

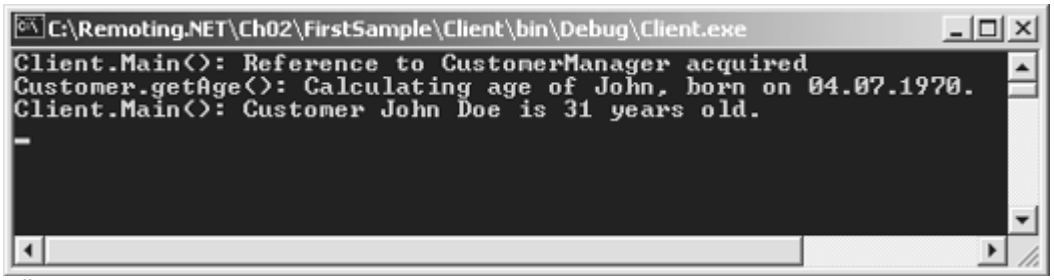
On the server you need to provide an implementation of ICustomerManager that will allow you to load a customer from a fictitious database; in the current example, this implementation will only fill the Customer object with static data.

```
using System.Runtime.Remoting;
using General;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Http;
As described previously, you will have to implement ICustomerManager in an object derived
from MarshalByRefObject. The method GetCustomer()will just return a dummy Customer object:
class CustomerManager: MarshalByRefObject, ICustomerManager
{
public CustomerManager()
{
Console.WriteLine("CustomerManager.constructor: Object created");
}
public Customer GetCustomer(int id)
{
Console.WriteLine("CustomerManager.GetCustomer(): Called");
Customer tmp = new Customer();
tmp.FirstName = "John";
tmp.LastName = "Doe";
tmp.DateOfBirth = new DateTime(1970,7,4);
Console.WriteLine("CustomerManager.GetCustomer(): Returning " +
"Customer-Object");
return tmp;
}
}

class ServerStartup
{
static void Main(string[] args)
{
HttpChannel chnl = new HttpChannel(1234);
ChannelServices.RegisterChannel(chnl);
RemotingConfiguration.RegisterWellKnownServiceType(
typeof(CustomerManager),
"CustomerManager.soap",
WellKnownObjectMode.Singleton);
// the server will keep running until keypress.
Console.ReadLine();
}
}
```

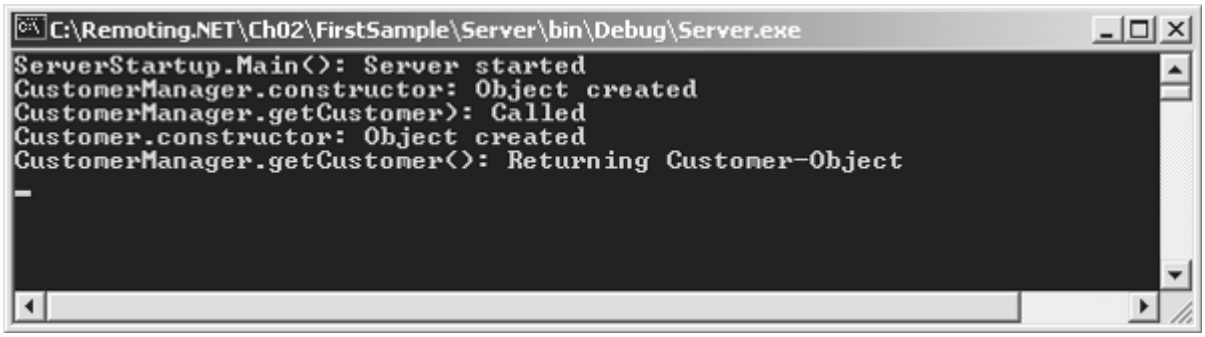
Implementing the Client

```
using System.Runtime.Remoting;
using General;
using System;
using System.Runtime.Remoting.Channels.Http;
using System.Runtime.Remoting.Channels;

class Client
{
static void Main(string[] args)
{
HttpChannel channel = new HttpChannel();
ChannelServices.RegisterChannel(channel);
ICustomerManager mgr = (ICustomerManager) Activator.GetObject(
typeof(ICustomerManager),
"http://localhost:1234/CustomerManager.soap");
Console.WriteLine("Client.Main(): Reference to CustomerManager acquired");
Customer cust = mgr.GetCustomer(4711);
int age = cust.GetAge();
Console.WriteLine("Client.Main(): Customer {0} {1} is {2} years old.",
cust.FirstName,
cust.LastName,
age);
Console.ReadLine();
}
}
```

Résultats



Client output



Server output

Un deuxième exemple : utilisation des objets sérialisables

Quite commonly, data has to be validated against several business rules. It's very convenient and maintainable to place this validation code on a central server. To allow validation of Customer data, you will extend the ICustomerManager interface to include a validate() method. This method will take a Customer object as a parameter and return another object by value. This returned object contains the status of the validation and explanatory text. As a sample business rule, you will check if the customer has been assigned a first name and last name and is between 0 and 120 years old.

General Assembly

In the General assembly extend the interface ICustomerManager to include the method Validate().

```
public interface ICustomerManager
{
Customer GetCustomer(int id);
ValidationResult Validate (Customer cust);
}
The ValidationResult is defined as follows. It will be a serializable (transfer by value) object
with a constructor to set the necessary values.
[Serializable]
public class ValidationResult
{
public ValidationResult (bool ok, String msg)
{
Console.WriteLine("ValidationResult.ctor: Object created");
this.Ok = ok;
this.ValidationMessage = msg;
}
public bool Ok;
public String ValidationMessage;
}
```

Server

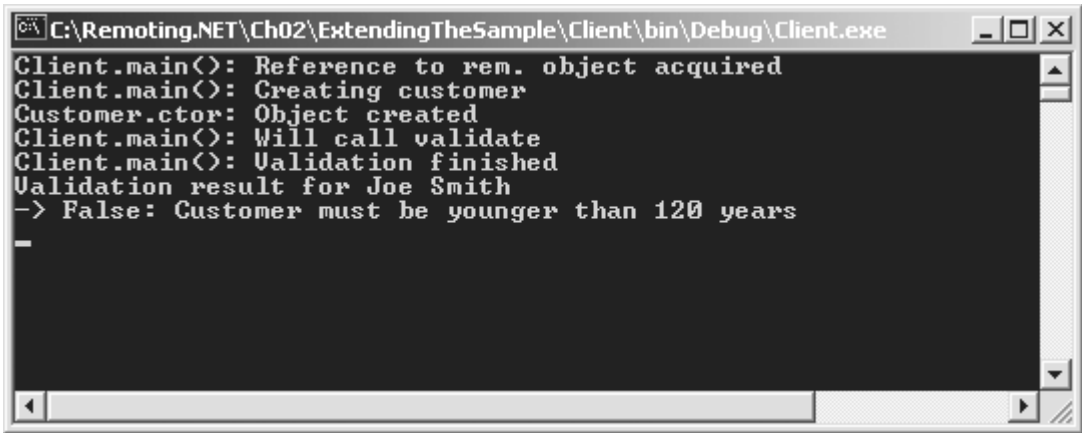
On the server, you have to provide an implementation of the mentioned business rule:

```
public ValidationResult Validate(Customer cust)
{
int age = cust.GetAge();
Console.WriteLine("CustomerManager.Validate() for {0} aged {1}",
cust.FirstName, age);
if ((cust.FirstName == null) || (cust.FirstName.Length == 0))
{
return new ValidationResult(false,"Firstname missing");
}
if ((cust.LastName == null) || (cust.LastName.Length == 0))
{
return new ValidationResult(false, "Lastname missing");
}
if (age < 0 || age > 120)
{
return new ValidationResult(false,"Customer must be " +
"younger than 120 years");
}
return new ValidationResult(true,"Validation succeeded");
}
```

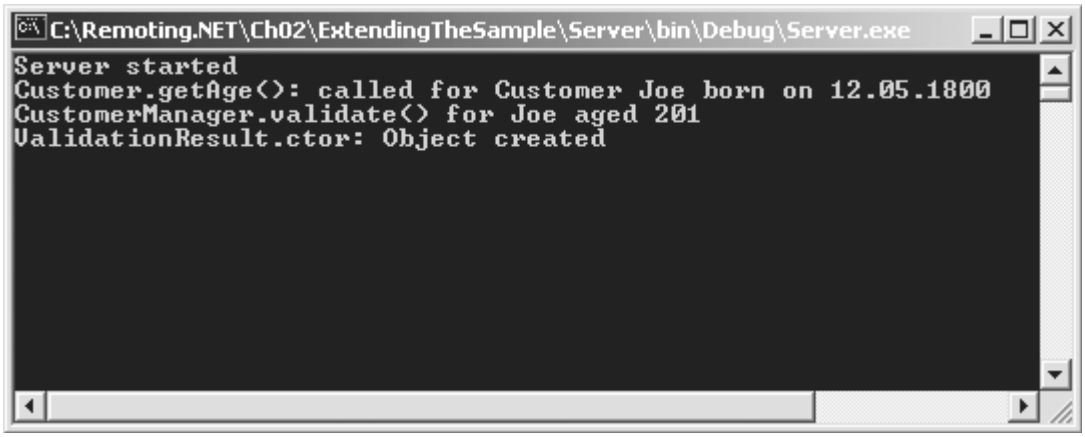
Client

```
static void Main(string[] args)
{
HttpChannel channel = new HttpChannel();
ChannelServices.RegisterChannel(channel);

ICustomerManager mgr = (ICustomerManager) Activator.GetObject(
typeof(ICustomerManager),
"http://localhost:1234/CustomerManager.soap");
Console.WriteLine("Client.main(): Reference to rem. object acquired");
Console.WriteLine("Client.main(): Creating customer");
Customer cust = new Customer();
cust.FirstName = "Joe";
cust.LastName = "Smith";
cust.DateOfBirth = new DateTime(1800,5,12);
Console.WriteLine("Client.main(): Will call validate");
ValidationResult res = mgr.validate (cust);
Console.WriteLine("Client.main(): Validation finished");
Console.WriteLine("Validation result for {0} {1}\n-> {2}: {3}",
cust.FirstName, cust.LastName,res.Ok.ToString(),
res.ValidationMessage);
Console.ReadLine();
}
```

Résultats



Client's output when validating a customer



Server's output while validating a customer

Un troisième Exemple : SingleCall

The shared assembly *General.*dll will contain the interface to a very simple remote object that allows the storage and retrieval of stateful information in the form of an int value.

```
using System;
namespace General
{
public interface IMyRemoteObject
{
CHAPTER 3 ■ .NET REMOTING IN ACTION 27
void SetValue (int newval);
int GetValue();
}
}
```

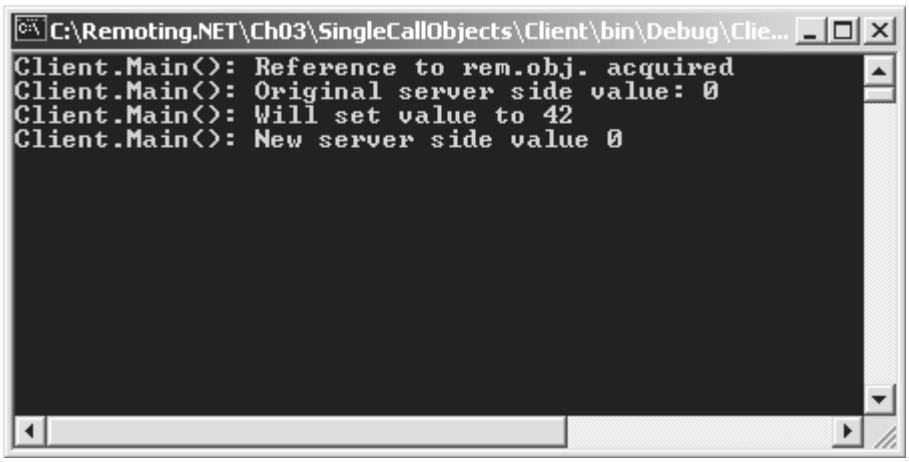
Client

```
using System;
using System.Runtime.Remoting;
using General;
using System.Runtime.Remoting.Channels.Http;
using System.Runtime.Remoting.Channels;
namespace Client
{
class Client
{
static void Main(string[] args)
{
HttpChannel channel = new HttpChannel();
ChannelServices.RegisterChannel(channel);
IMyRemoteObject obj = (IMyRemoteObject) Activator.GetObject(
typeof(IMyRemoteObject),
"http://localhost:1234/MyRemoteObject.soap");
Console.WriteLine("Client.Main(): Reference to rem. obj acquired");
int tmp = obj.GetValue();
Console.WriteLine("Client.Main(): Original server side value: {0}",tmp);
Console.WriteLine("Client.Main(): Will set value to 42");
obj.SetValue(42);
tmp = obj.GetValue();
Console.WriteLine("Client.Main(): New server side value {0}", tmp);
Console.ReadLine();
}
}
}
```

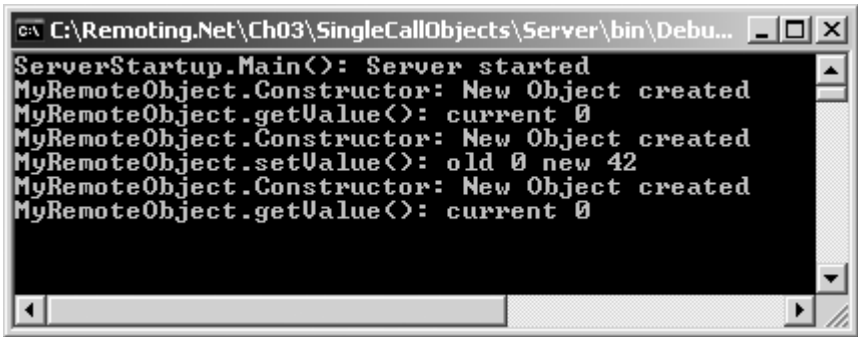
Server

```
using System;
using System.Runtime.Remoting;
using General;
using System.Runtime.Remoting.Channels.Http;
using System.Runtime.Remoting.Channels;
namespace Server
{
class MyRemoteObject: MarshalByRefObject, IMyRemoteObject
{
int myvalue;
public MyRemoteObject()
{
Console.WriteLine("MyRemoteObject.Constructor: New Object created");
}
public MyRemoteObject(int startvalue)
{
Console.WriteLine("MyRemoteObject.Constructor: .ctor called with {0}",
startvalue);
myvalue = startvalue;
}
public void SetValue(int newval)
{
Console.WriteLine("MyRemoteObject.setValue(): old {0} new {1}",
myvalue,newval);
myvalue = newval;
}

public int GetValue()
{
Console.WriteLine("MyRemoteObject.getValue(): current {0}",myvalue);
return myvalue;
}
}
class ServerStartup
{
static void Main(string[] args)
{
Console.WriteLine ("ServerStartup.Main(): Server started");
HttpChannel chnl = new HttpChannel(1234);
ChannelServices.RegisterChannel(chnl);

RemotingConfiguration.RegisterWellKnownServiceType(
typeof(MyRemoteObject),
"MyRemoteObject.soap",
WellKnownObjectMode.SingleCall);

// the server will keep running until keypress.
Console.ReadLine();
}
}
}
```

Résultats



Client's output for a SingleCall object



Server's output for a SingleCall object

Un quatrième exemple : Synchronous Calls

The Shared Assembly's Source Code

```
using System;
using System.Runtime.Remoting.Messaging;
namespace General
{
public interface IMyRemoteObject
{
void SetValue(int newval);
int GetValue();
String GetName();
}
}
```

Creating the Server

The server implements the defined methods with the addition of making the SetValue() and GetName() functions long-running code. In both methods, a five-second delay is introduced so you can see the effects of long-lasting execution in the different invocation contexts.

```
using System;
using System.Runtime.Remoting;
using General;
using System.Runtime.Remoting.Channels.Http;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Messaging;
using System.Collections;
using System.Threading;
namespace Server
{



class MyRemoteObject: MarshalByRefObject, IMyRemoteObject
{
int myvalue;
public MyRemoteObject()
{
Console.WriteLine("MyRemoteObject.Constructor: New Object created");
}
public void SetValue(int newval)
{
Console.WriteLine("MyRemoteObject.setValue(): old {0} new {1}",
myvalue,newval);
// we simulate a long running action
Console.WriteLine(" .setValue() -> waiting 5 sec before setting" +
" value");
Thread.Sleep(5000);
myvalue = newval;
Console.WriteLine(" .SetValue() -> value is now set");
}
public int GetValue()
{
Console.WriteLine("MyRemoteObject.GetValue(): current {0}",myvalue);
return myvalue;
}
public String GetName()
{
Console.WriteLine("MyRemoteObject.getName(): called");
```

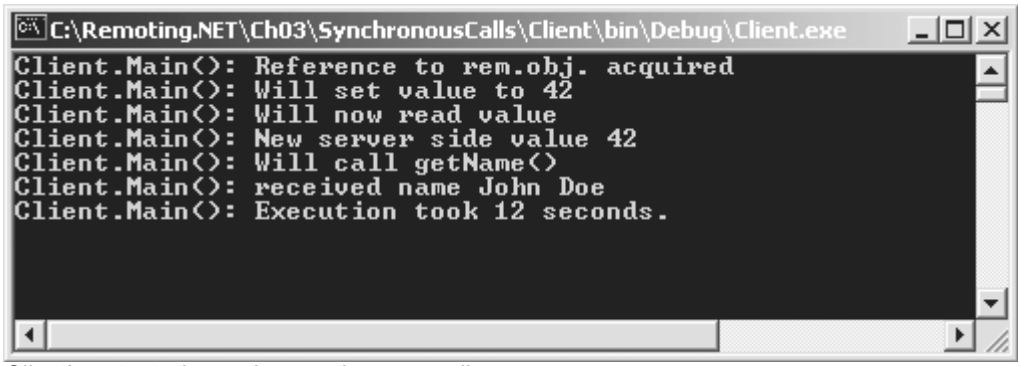
```
// we simulate a long running action
Console.WriteLine(" .GetName() -> waiting 5 sec before continuing");
Thread.Sleep(5000);
Console.WriteLine(" .GetName() -> returning name");
return "John Doe";
}
}



class ServerStartup
{
static void Main(string[] args)
{
Console.WriteLine ("ServerStartup.Main(): Server started");
HttpChannel chnl = new HttpChannel(1234);
ChannelServices.RegisterChannel(chnl);
RemotingConfiguration.RegisterWellKnownServiceType(
typeof(MyRemoteObject),
"MyRemoteObject.soap",
WellKnownObjectMode.Singleton);
// the server will keep running until keypress.
Console.ReadLine();
}
}
}
```

Creating the Client

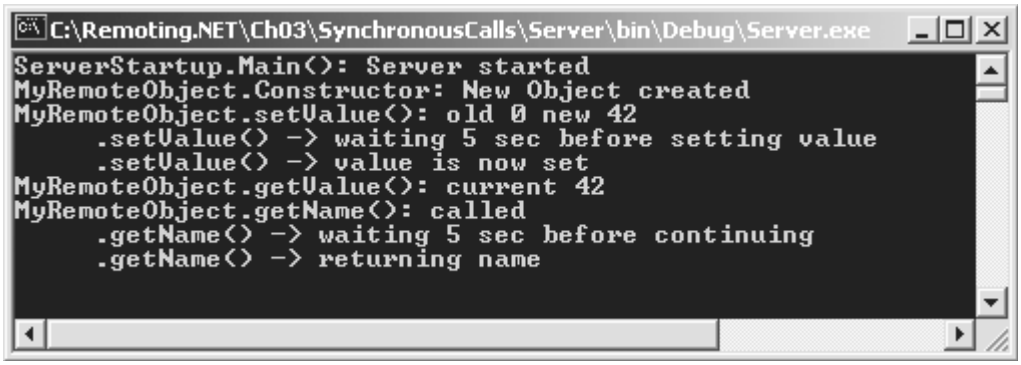
```
using System;
using System.Runtime.Remoting;
using General;
using System.Runtime.Remoting.Channels.Http;
using System.Runtime.Remoting.Channels.Tcp;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Proxies;
using System.Threading;
using System.Reflection;
namespace Client
{
class Client
{
static void Main(string[] args)
{
DateTime start = System.DateTime.Now;
HttpChannel channel = new HttpChannel();
ChannelServices.RegisterChannel(channel);

IMyRemoteObject obj = (IMyRemoteObject) Activator.GetObject(
typeof(IMyRemoteObject),
"http://localhost:1234/MyRemoteObject.soap");

Console.WriteLine("Client.Main(): Reference to rem.obj. acquired");
Console.WriteLine("Client.Main(): Will set value to 42");
obj.SetValue(42);
Console.WriteLine("Client.Main(): Will now read value");
int tmp = obj.GetValue();
Console.WriteLine("Client.Main(): New server side value {0}", tmp);
Console.WriteLine("Client.Main(): Will call getName()");
String name = obj.GetName();
Console.WriteLine("Client.Main(): received name {0}",name);
DateTime end = System.DateTime.Now;
TimeSpan duration = end.Subtract(start);
Console.WriteLine("Client.Main(): Execution took {0} seconds.",
```

```
duration.Seconds);
Console.ReadLine();
}
}
}
```

Résultats



Client's output when using synchronous calls

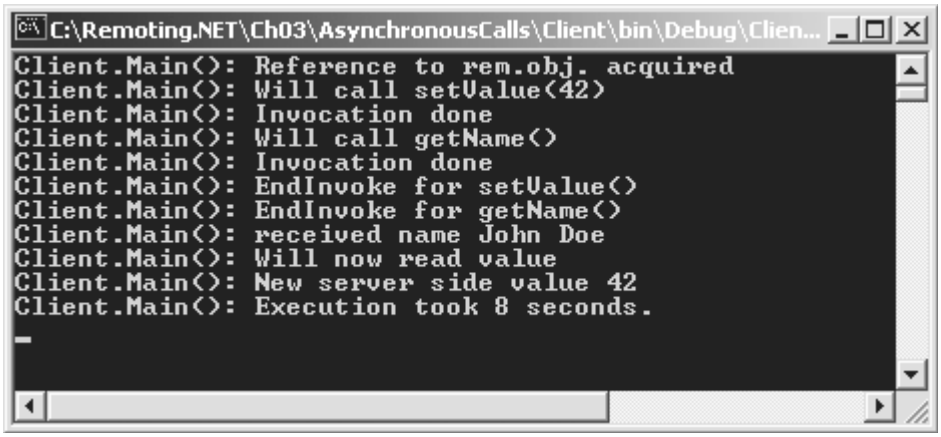


Server's output when called synchronously

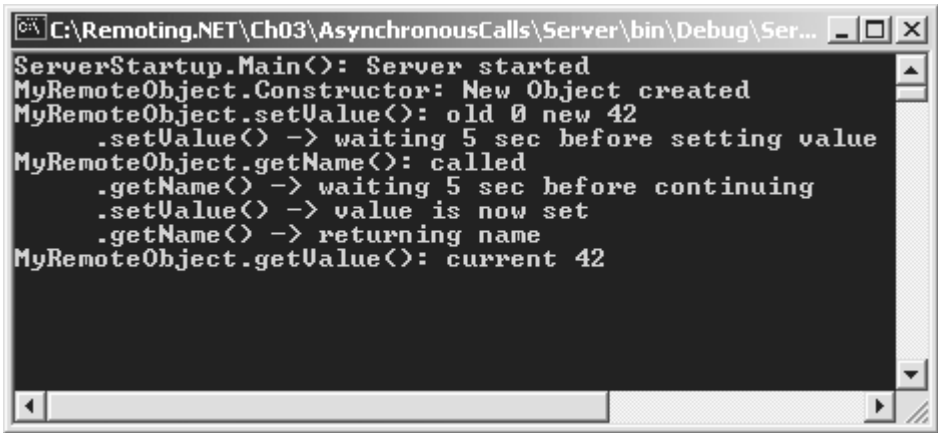
Cinquième Exemple : Asynchronous Calls

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

using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Proxies;
using System.Threading;
namespace Client
{
class Client
{
delegate void SetValueDelegate(int value);
delegate String GetNameDelegate();
static void Main(string[] args)
{
DateTime start = System.DateTime.Now;
HttpChannel channel = new HttpChannel();
ChannelServices.RegisterChannel(channel);
IMyRemoteObject obj = (IMyRemoteObject) Activator.GetObject(
typeof(IMyRemoteObject),
"http://localhost:1234/MyRemoteObject.soap");
Console.WriteLine("Client.Main(): Reference to rem.obj. acquired");
Console.WriteLine("Client.Main(): Will call setValue(42)");

SetValueDelegate svDelegate = new SetValueDelegate(obj.SetValue);
IAsyncResult svAsyncres = svDelegate.BeginInvoke(42,null,null);

Console.WriteLine("Client.Main(): Invocation done");
Console.WriteLine("Client.Main(): Will call GetName()");

GetNameDelegate gnDelegate = new GetNameDelegate(obj.GetName);
IAsyncResult gnAsyncres = gnDelegate.BeginInvoke(null,null);

Console.WriteLine("Client.Main(): Invocation done");
Console.WriteLine("Client.Main(): EndInvoke for SetValue()");

svDelegate.EndInvoke(svAsyncres);
Console.WriteLine("Client.Main(): EndInvoke for SetName()");

String name = gnDelegate.EndInvoke(gnAsyncres);
Console.WriteLine("Client.Main(): received name {0}",name);
Console.WriteLine("Client.Main(): Will now read value");
int tmp = obj.GetValue();
Console.WriteLine("Client.Main(): New server side value {0}", tmp);
DateTime end = System.DateTime.Now;
TimeSpan duration = end.Subtract(start);
CHAPTER 3 ■ .NET REMOTING IN ACTION 55
Console.WriteLine("Client.Main(): Execution took {0} seconds.",
duration.Seconds);
Console.ReadLine();
}
}
}
```

Résultats



Client output when using asynchronous calls



Server's output when called asynchronously