TD Objets distribués : Introduction à .NET Remoting

Vous mettrez en œuvre votre première application en .NET remoting. Les sources sont disponibles sur :

`\\anis\Exemples\DOTNET Remoting\SourcesTD1_Remoting.zip`

Dans ce TD, on crée un serveur qui distribue sur le réseau un objet Compteur avec des possibilités de consultation et de modification.

Environnement

1. Lancer Visual Studio .NET et créez une nouvelle solution vide "Remoting".
2. Ajoutez, dans votre solution, deux nouveaux projets Visual C# de type "Application Console" que vous nommerez "Serveur" et "Client".
3. Ajoutez, dans votre solution, un nouveau projet Visual C# de type "Bibliothèque de classes" que vous nommerez "IRemote". Il s'agit là de l'interface de l'objet Compteur.

Interface

Définissons l'interface de notre objet Compteur : ajouter dans le projet IRemote le code C# suivant :

```
using System;
namespace IRemote
{
	public interface ICompteur
	{
		void Augmenter(double dValeur);
	}
}
```

Quel est le rôle de ces lignes de code ?

Objet Compteur 'Côté serveur'

Créons la classe Compteur. Cette classe sera définie dans le fichier Compteur.cs au sein du projet Serveur.

Voici le code Compteur.cs :

```
	public class Compteur
	{
		public double dNiveau;
		public void Augmenter(double dValeur)
		{
			this.dNiveau+=dValeur;
			Console.WriteLine(" - Compteur Augmenté, nouveau niveau : " +
this.dNiveau.ToString() + " -");
		}
	}
```

Le serveur.

Ajouter le code Serveur.cs suivant :

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

namespace Serveur
{
```

```
	public class CompteurManager:MarshalByRefObject,IRemote.ICompteur
	{
		//C'est cet objet que l'on distribue sur le réseau
		//Voir plus loin son code
	}
	public class RemotingServer
	{
		//Ce code va publier notre objet CompteurManager
		//Voir plus loin son code
	}
}
```

Ajouter au projet Serveur la référence à la dll .NET System.Runtime.Remoting. Cette dll contient toutes les classes nécessaires au remoting.
Il faudra aussi ajouter une référence à l'interface de l'objet Compteur.
La classe qui sera servie s'appellera CompteurManager. Cette classe implémente l'interface que l'on a définie plus haut. Elle doit donc implémenter les méthodes qui sont listées dans l'interface.

```
	public class CompteurManager:MarshalByRefObject,IRemote.ICompteur
	{
		DemarreServeur myclass = new DemarreServeur();
		//C'est cette classe que l'on distribue sur le reseau
		public override object InitializeLifetimeService()
		{
			//Durée de vie de l'objet : pour "toujours"
			return null;
		}
		public void Augmenter(double dMontant)
		{
			myclass.monCompteur.Augmenter(dMontant);
		}
	}
```

La classe RemotingServer aura une méthode pour "démarrer" le serveur. Dans cette méthode il faut décrire le canal de transmission en spécifiant :
• le protocole utilisé au sein de ce canal,
• le port utilisé par ce canal (éviter certains ports…)
• le mode de transmission des données (il y a deux choix possible : Singleton ou SingleCall)

- Singleton : à chaque instant nous avons qu'une seule instance de l'objet. On peut associer à cette instance une durée de vie (pouvant être l'infini) Après cette durée de vie écoulée, le serveur détruit l'instance courante de l'objet et crée une nouvelle instance.
- SingleCall : une nouvelle instance de l'objet sera créée à chaque appel et détruite juste après.

```
public class RemotingServer
	{
		public void Start()
		{
			int canal;
			//Ouvre un canal en mode TCP sur le port '1234'
			canal=1234;
			TcpChannel chnl_1 = new TcpChannel(canal);
			ChannelServices.RegisterChannel(chnl_1,true);

			//Publier notre objet CompteurManager

	RemotingConfiguration.RegisterWellKnownServiceType(typeof(CompteurManager),
"CompteurManager", WellKnownObjectMode.Singleton);
		}
```

```
	}
```

On spécifie dans la méthode RegisterWellKnownServiceType que c'est la classe CompteurManager que l'on sert. Si vous voulez que votre objet dure pour "toujours", il vous suffit de surcharger la méthode InitializeLifetimeService. La durée de vie peut avoir une valeur spécifique ou être infinie (valeur "null")est ainsi infinie.
Finalement, écrivons dans le fichier Main.cs quelques lignes de code qui démarre un thread pour lancer le serveur.

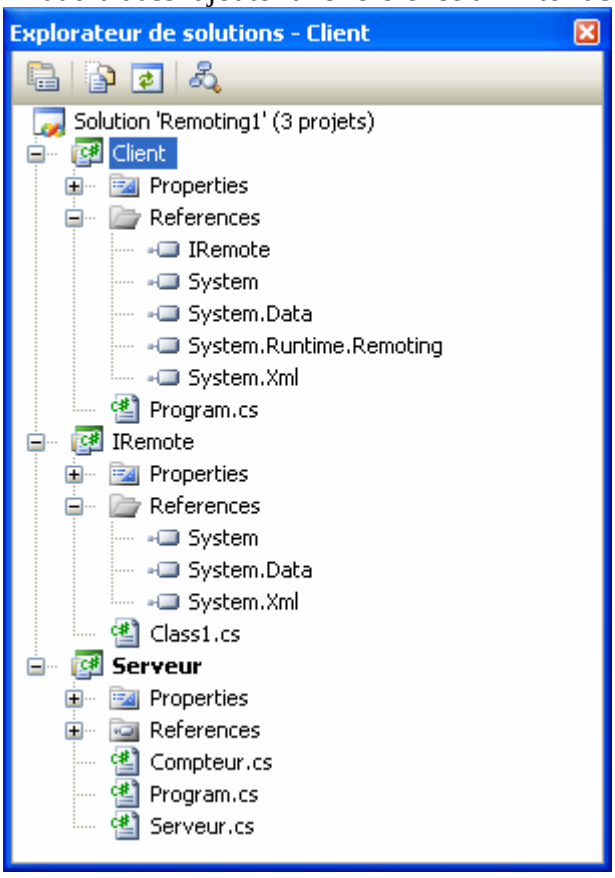
```
	class DemarreServeur
	{
		public Compteur monCompteur = new Compteur();
		[STAThread]
		static void Main(string[] args)
		{
			Serveur.RemotingServer myServer = new Serveur.RemotingServer();
			try
			{
				Console.WriteLine(" Serveur : Demarrage ...");
				myServer.Start();
				Console.WriteLine("Serveur démarré...");
				Console.ReadLine();
			}
			catch (Exception ex)
			{
				Console.WriteLine("Serveur : Erreur d'initialisation !!!
" + ex.Message);
			}
		}
```

Compiler et exécuter le serveur.
Le client qui utilise le service :
Ouvrez le même canal que le serveur, et prenez une référence à l'objet servi. Ensuite il vous suffit d'appeler les méthodes par cette référence :

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
namespace Client
{
	class Client
	{
		/// <summary>
		/// The main entry point for the application.
		/// </summary>
		[STAThread]
		static void Main(string[] args)
		{
			try
			{
	//Ouvre un canal TCP côté client
				TcpChannel chnl = new TcpChannel();
				//Pour VS2003
				//ChannelServices.RegisterChannel(chnl_1);
				//Pour VS2005
				ChannelServices.RegisterChannel(chnl_1,true);
	//Prendre la référence sur le serveur (tcp://nommachine:port/CompteurManager)
				IRemote.ICompteur CompteurDistant =
				(IRemote.ICompteur)Activator.
				GetObject(typeof(IRemote.ICompteur),
				"tcp://localhost:1234/CompteurManager");
	//Verifier que la reférence au serveur a été acquise
```

```
                        if (CompteurDistant == null)
                        {
                                Console.WriteLine("Référence non acquise");
                        }
                        else
                        {
                                Console.WriteLine("Référence au serveur acquise ");
        //On peut à présent appeler les méthodes CompteurDistant
                                CompteurDistant.Augmenter(2);
        //Attendre les instructions sur la console...
                                Console.ReadLine();
                }
                        }
                        catch (Exception ex)
                        {
                                Console.WriteLine("ERREUR :" + ex.Message);
                        }
                }
        }
}
```

Ajouter au projet Serveur la référence à la dll .NET System.Runtime.Remoting. Cette dll contient toutes les classes nécessaires au remoting.
Il faudra aussi ajouter une référence à l'interface de l'objet Compteur.



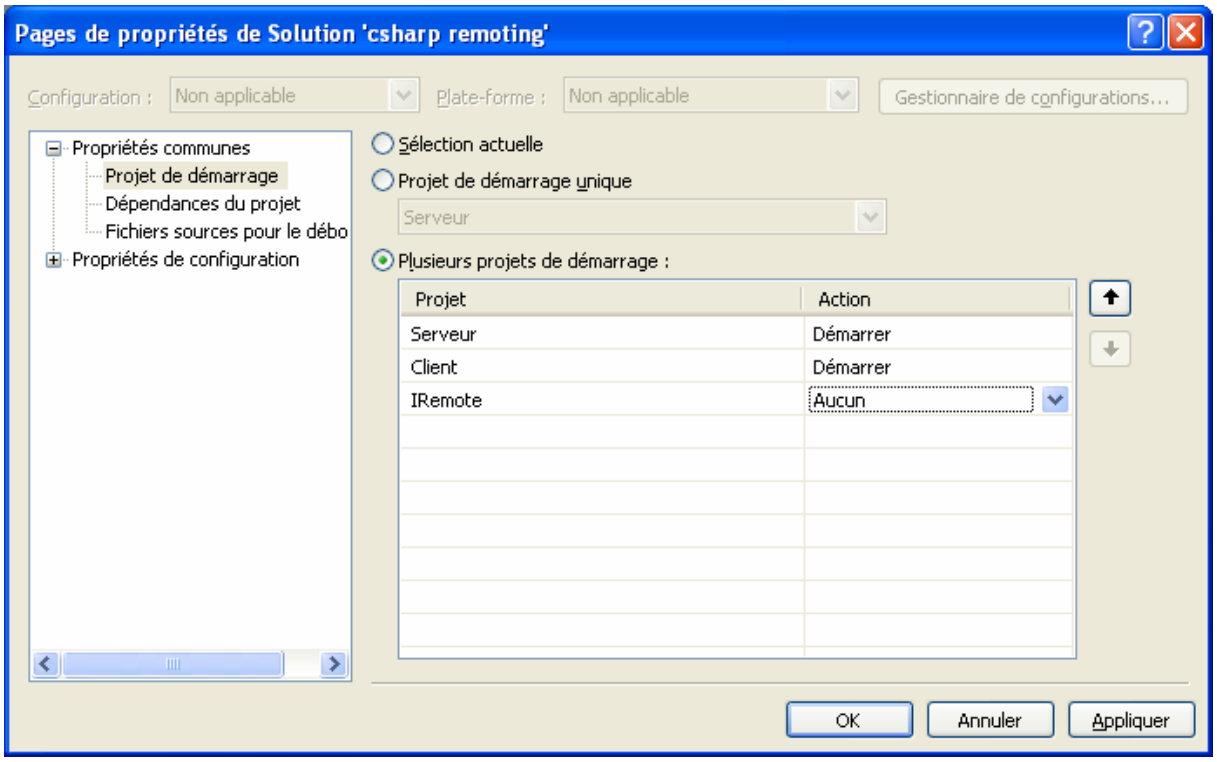
1. Compiler et exécuter le client. Vérifier que :

- le serveur crée une fois l'instance Compteur,
- les appels aux méthodes ont été reçus par l'objet Compteur (sur la console du serveur),
- si vous lancez un autre client sans arrêter le serveur, vous verrez qu'il s'agit du même objet Compteur sur le serveur qui est utilisé depuis le début et qui continue d'être augmenté (effet du mode Singleton),

2. Modifier votre client afin de le connecter sur le serveur de votre voisin (client et serveur sur 2 machines séparées).
3. Rajouter et utiliser des méthodes pour :

- diminuer la valeur du compteur,
- afficher la valeur du compteur chez le client,

4. Modifier le client de telle sorte que l'utilisateur puisse saisir les modifications à faire sur le compteur (par exemple : ajouter 10, enlever 13…).
5. Utiliser le mode Singleton en SingleCall : que constatez-vous ?
6. Ecrire la classique application 'Hello World' en version distribuée.


Pages de propriétés de Solution 'csharp remoting'
Configuration :
Non applicable
Plate-forme :
Non applicable
Gestionnaire de configurations...
Propriétés communes
Projet de démarrage
Dépendances du projet
Fichiers sources pour le débo
Propriétés de configuration
Sélection actuelle
Projet de démarrage unique
Serveur
Plusieurs projets de démarrage :
Projet
Action
Serveur
Démarrer
Client
Démarrer
IRemote
Aucun
OK
Annuler
Appliquer