TD1 Objets distribués : .NET Remoting

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

http://dept-info.labri.fr/~felix/Annee2007-08/LicenceProAlt/ObjetsDistristribues/TD1/RemotingJuin08.zip

Dans ce TD, on crée un serveur qui distribue sur le réseau un objet Compteur avec des possibilités de consultation et de modification. Cet objet Compteur sera par la suite utilisé par un client.

Environnement

1. Lancez Visual Studio .NET et créez une nouvelle '*Solution Visual Studio*' que vous nommerez "Remoting".
2. Ajoutez, dans votre solution, deux nouveaux projets Visual C# de type "Application Console" que vous nommerez "Serveur" et "Client". Les sources Program.cs créés automatique seront renommés Serveur.cs (projet Serveur) et Client.cs (projet 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

Définissez 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'

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

Voici le code Compteur.cs :

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

Le serveur.

Ajoutez le fichier '*Serveur.cs*' au projet Serveur avec le code 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
	}
}
```

Ajoutez au projet Serveur une référence à la dll System.Runtime.Remoting (dll .NET contenant toutes les classes nécessaires au .NET remoting).
Ajoutez au projet Serveur 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. Voici le code à intégrer dans la source Serveur.cs.

```
	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. Il faut spécifier plusieurs choses dans cette méthode :
• le protocole utilisé (TCP)
• le port du canal utilisé (1234),
• le mode de transmission des données :
  • Singleton : Il n'y a uniquement qu'une seule instance de l'objet au même moment. On peut leur associer une durée de vie (quelques secondes ou éternel). Apres la durée de vie écoulée, le serveur recrée une instance de l'objet à nouveau et la servira aux clients jusqu'à la fin de durée de vie.
  • SingleCall : une nouvelle instance de l'objet sera crée 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);

                //Publie 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 une méthode qui s'appelle InitializeLifetimeService. En retournant "null" dans cette méthode, la durée de vie est ainsi infinie. Mais vous pouvez aussi mettre une durée spécifique.
Finalement, définissez le point d'entrée de Program.cs (la classe ci-dessous remplacera la classe Program définie par défaut) :

```
    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);
            }
        }
```

Le code ci-dessus démarre un thread qui va lancer l'exécution du serveur.

Compilez et exécutez 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 :
Au sein du projet Client, définissez le point d'entrée du Program.cs (la classe ci-dessous remplacera la classe Program définie par défaut) :

```
namespace Client
{
    class Client
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main(string[] args)
        {
            try
            {
    //Enregistrer le canal TCP port 1234
                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);
                        }
                }
        }
}
```

Rajoutez les clauses ‘using’ suivantes au début de la source Program.cs :

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

Ajoutez au projet Client une référence à la dll System.Runtime.Remoting (dll .NET contenant toutes les classes nécessaires au .NET remoting).

Ajoutez au projet Client une référence à l’interface de l’objet ‘*Compteur*’.

1. Compilez le client.
2. Après avoir lancé l’exécution du serveur, exécutez 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),
3. Modifier votre client afin de le connecter sur le serveur de votre voisin (client et serveur sur 2 machines séparées).
4. Rajouter et utiliser des méthodes pour :
   - diminuer la valeur du compteur,
   - afficher la valeur du compteur chez le client,
5. Modifier le client de telle sorte que l’utilisateur puisse saisir les modifications à faire sur le compteur (par exemple : ajouter 10, enlever 13…).
6. Ecrire la classique application ‘Hello World’ en version distribuée.