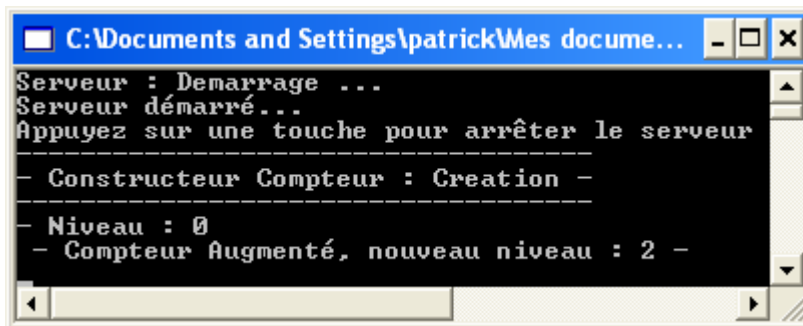


TD Objets distribués : Introduction à .NET Remoting

Vous mettez en œuvre votre première application en .NET remoting. Les sources sont disponibles sur : http://dept-info.labri.fr/~felix/Annee2006-07/LicencePro/TD1_Remoting/

Dans ce TD, on crée un serveur qui distribue sur le réseau un objet Compteur avec des possibilités de consultation et de modification à partir de clients répartis sur le réseau. Voici ce que donne la console associée à l'objet Compteur lors de son exécution :



```
C:\Documents and Settings\patrick\Mes docume...
Serveur : Demarrage ...
Serveur démarré...
Appuyez sur une touche pour arrêter le serveur
-----
Constructeur Compteur : Creation -
-----
Niveau : 0
Compteur Augmenté, nouveau niveau : 2 -
```

Dans ce scénario, l'objet Compteur a démarré avec une valeur égale à 0 et une augmentation de valeurs 2 a été demandée par un client.

Environnement

1. Lancer Visual Studio .NET 2003 et créez une nouvelle solution vide "Remoting1".
2. Ajoutez, dans votre solution, un nouveau projet Visual C# de type "Application Console" que vous nommerez "Serveur". Renommer Class1.cs en Main.cs.
3. Ajoutez, dans votre solution, un nouveau projet Visual C# de type "Application Console" que vous nommerez "Client". Renommer Class1.cs en Main.cs.
4. 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 int iID;
    public string strNom;
```

```

    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 une référence à la dll .NET qui s'appelle 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. Il faut spécifier plusieurs choses dans cette méthode :

- le canal de transmission, et quel protocole il utilise (HTTP, TCP etc...)
- le port du canal (éviter certains ports...)
- le mode de transmission des données (il y a deux choix possible : Singleton ou SingleCall)
 - 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). Après 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 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);

        //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 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, écrivons dans le fichier Main.cs un 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.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
        {
            //Enregistrer le canal TCP port 1234
            TcpChannel chnl = new TcpChannel();
            ChannelServices.RegisterChannel(chnl);
            //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 référence 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 Client une référence à la dll .NET qui s'appelle 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 lancer 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. Ecrire la classique application 'Hello World' en version distribuée.