Developpez.com
Club des développeurs

Introduction à Windows Communication Foundation

par Lainé Vincent (autres articles)

Date de publication : 30/03/2007

Dernière mise à jour : 30/03/2007

Cet article vous présente le principe de la communication entre applications et en particulier Windows Communication Foundation pour l'implémentation

I - Pourquoi faire des applications qui communiquent ?
II - Windows Communication Foundation ? C'est quoi ?
III - Comment fonctionne la communication entre applications ?
III-A - Définir les méthodes exposées par le serveur
III-B - Définir les types de données transmissibles entre applications
III-C - Définir l'abc de la communication
III-C-1 - Définir l'adresse du service
III-C-2 - Définir le binding du service
III-C-3 - Définir le contract du service
III-D - Conclusion
IV - Etude de cas : Service de déploiement à distance
IV-A - Structure du projet
IV-B - SoftwareInstallerLibrary
IV-C - SoftwareInstallerServerService
IV-C-1 - Création de la base de WCF
IV-C-2 - L'implémentation du service WCF
IV-D - SoftwareInstallerClientService
V - Conclusion
VI - Téléchargements
VII - Remerciements

Copyright © 09/03/2007 - Vincent Lainé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

I - Pourquoi faire des applications qui communiquent ?

C'est la base de tout : Pourquoi faire une application qui permet à une autre de communiquer avec elle ?

Assurément toute les applications n'ont pas besoin de communiquer avec les autres. Alors quand mettre en oeuvre une communication entre applications ?

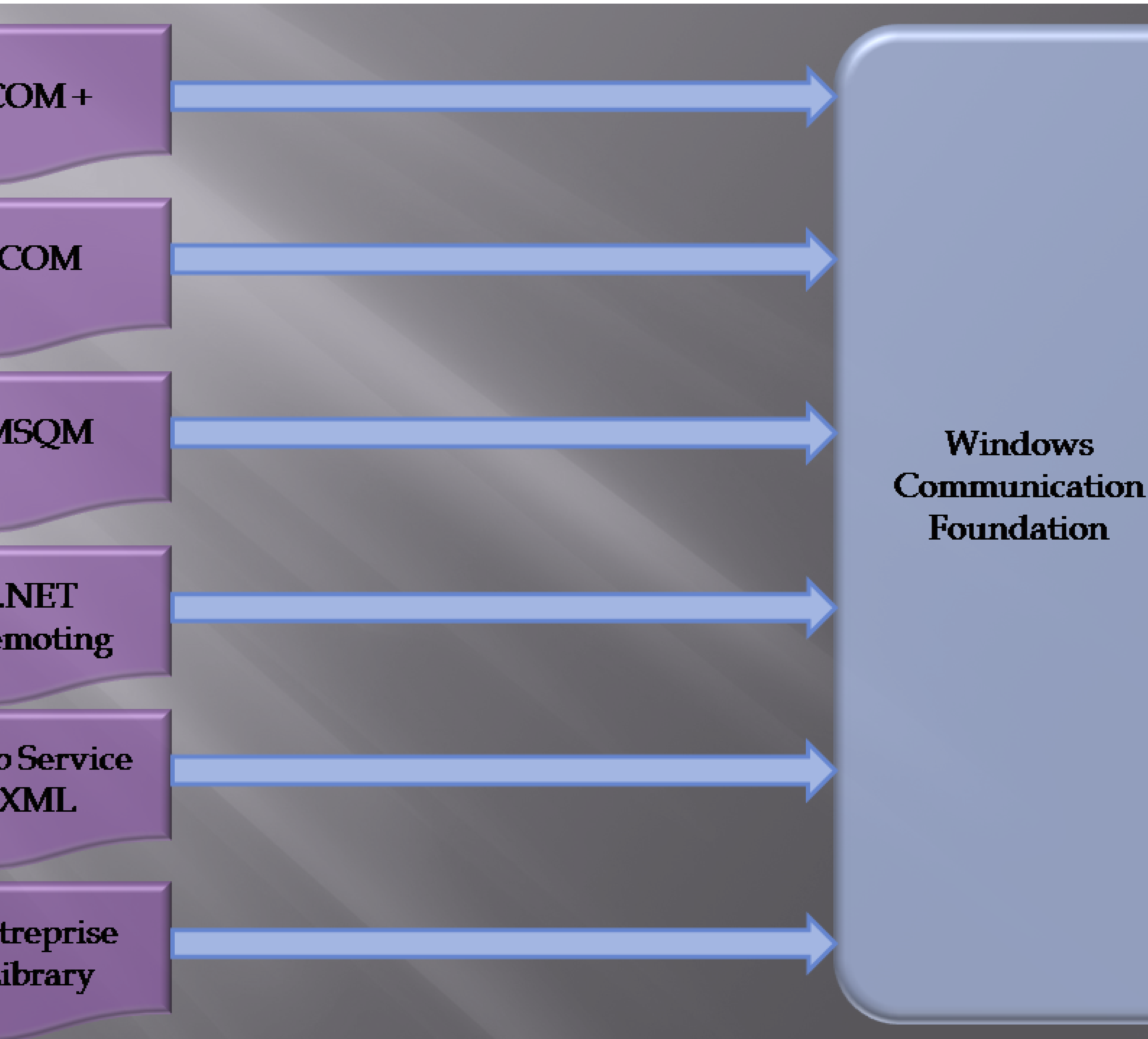
- Si vous avez par exemple une application qui doit accèder à des ressources spécifiques à un poste.
- A des fins de centralisation de données
- A des fins de répartitions de charges sur plusieurs postes
- Afin de ne pas réécrire un traitement qui existe dans une autre application
- Pour pouvoir contrôler une application

Ce ne sont que des exemples mais ils résument bien les différents cas qui sont le plus souvent rencontrés

Copyright © 09/03/2007 - Vincent Lainé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

II - Windows Communication Foundation ? C'est quoi ?

Windows Communication Foundation (WCF) est la nouvelle couche de communication du framework 3.0. Cette couche a été créée afin d'unifier les différents modèles d'écritures d'applications "communicantes".



Unification des modèles de développement de communications

WCF fait partie du framework 3.0 qui est livré d'office avec Vista et est disponible sous Windows XP et 2003 serveur.

Copyright © 09/03/2007 - Vincent Lainé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Pour pouvoir utiliser WCF dans vos applications vous devez ajouter la référence à **System.ServiceModel** à vos projets. Ensuite vous devez faire un **using System.ServiceModel;**

Copyright © 09/03/2007 - Vincent Lainé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

III - Comment fonctionne la communication entre applications ?

Afin que les applications puissent communiquer il faut qu'un certain nombres de contraintes soient respectées.

- Définir les méthodes exposées par le serveur
- Définir les types de données transmissibles entre applications
- Définir l'ABC de la communication

III-A - Définir les méthodes exposées par le serveur

Pour que le client puisse appeler des méthodes sur le serveur encore faut-il que ce premier connaisse les méthodes disponibles sur le serveur. WCF est très strict sur la façon d'implémenter la communication entre applications. En effet les méthodes exposées par WCF doivent toutes être définies dans une interface "décorée" par l'attribut ServiceContract. De plus les fonctions que l'on veut exposer par WCF doivent être "décorées" par l'attribut OperationContract.

```
using System.ServiceModel;

[ServiceContract]
public interface IMonPremierServiceWCF
{
 [OperationContract]
 void EcrireUnePhrase(string phrase);
}
```

Cette interface définie donc une méthode qu'un client peut appeler à distance. Au passage vous remarquerez que nous passons un paramètre sans aucun problème. Cette interface définit donc le *contrat* de notre service. En effet les types primaires tel que **string**, **int**, **float**, etc peuvent être utilisés sans aucun problème ni modification. Mais comment permettre à des types personnalisés d'être transmit par WCF d'un serveur à un client ?

III-B - Définir les types de données transmissibles entre applications

Afin de marquer une classe comme transmissible par WCF il suffit d'utiliser l'attribut **[DataContract]**. Cet attribut est défini dans l'assembly **System.Runtime.Serialization** qu'il faut donc ajouter à son projet. Ce choix d'implémentation est très intéressant car il permet de ne pas bousculer toute la modélisation d'un projet en obligeant l'héritage par une classe comme MarshalByRefObject en Remoting. Cela vous permet donc de migrer facilement et avec peu de modifications un projet qui n'as pas été initialement prévu pour fonctionner en WCF.

Une fois que votre classe est "décorée" avec l'attribut [DataContract] il faut ensuite identifier les méthodes que nous allons exposer aux futurs clients de notre application.

```
[DataContract]
public class MyDatas
{
 private string data1

 [DataMember]
 public string Data1
 {
  get
  {
   return this.data1;
```


Copyright © 09/03/2007 - Vincent Lainé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

```
    }

    set
    {
     this.data1 = value;
    }
   }
  }
```

III-C - Définir l'abc de la communication

Qu'est ce que l'abc de la communication ? C'est sous cet acronyme que Microsoft définit les 3 étapes essentielles du déploiement d'un service WCF.

- A pour Address : Définit l'adresse du serveur qui expose le service
- B pour Binding : Définit la façon dont le service sera exposé
- C pour Contract : Définit le contrat que le service remplit

La grande force de WCF est sans aucun doute la façon dont il a été pensé. En effet tout ce qui touche au déploiement et à la consommation d'un service WCF est défini dans le fichier de configuration de l'application. Cela permet de s'affranchir entièrement des contraintes de déploiement au moment du développement. En unifiant ainsi les différentes façons de programmer une application distribuer, WCF permet une souplesse et une facilité d'utilisation extrèmement agréable.

III-C-1 - Définir l'adresse du service

Afin de pouvoir exposer un service vous devez définir l'adresse à partir de laquelle il sera accèssible. Pour cela il faut définir les sections correspondantes dans le fichier de configuration de l'application. Ces sections doivent toutes se trouver dans <system.serviceModel> de votre fichier de configuration.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
 <system.serviceModel>
  <services>
   <service name="DvpDemoWCF.IMonPremierServiceWCF">
    <endpoint address="net.tcp://localhost:5000/DvpDemoWCF" binding="netTcpBinding"
 contract="DvpDemoWCF.IMonPremierServiceWCF"/>
   </service>
  </services>
 </system.serviceModel>
</configuration>
```

l'attribut *address* vous permet de définir un point de terminaison c'est à dire une URI où le service sera accessible. De plus cela vous permet de configurer la totalité de l'abc de la communication.

Ici l'adresse expose le service sur le poste local sur le port 5000 et à l'adresse **DvpDemoWCF**

III-C-2 - Définir le binding du service

Le binding est la façon dont le service sera exposé. C'est à dire qu'il vous permet de définir le protocole utilisé pour transporter vos objets sur le réseaux ou en local. Chaque binding à ses avantages et ses inconvénients mais au niveau du développeur ce choix ne doit pas être bloquant. En effet lors de votre développement vous ne savez pas toujours dans quel environnement votre application va être déployée. L'administrateur réseau ne veut peut être pas

Copyright © 09/03/2007 - Vincent Lainé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

ouvrir un port sur un serveur pour des raisons de sécurité, ou alors il voudrait pouvoir définir quel port. De même il préfère peut être vu la topologie de son réseau que les communications se fassent en tcp afin de réduire la charge réseau. Toutes ces questions ne sont donc pas à la charge du développeur mais bien de l'administrateur réseaux qui va s'occuper de l'installation de votre application. C'est la que l'on comprend une autre des choses merveilleuses de WCF. En .NET Remoting ce choix était fait par l'équipe de développement et l'administrateur réseaux n'avait aucun moyen d'influer dessus. WCF de part l'uniformisation des techniques d'utilisation des protocoles permet de rendre ce choix à la personne concernée : l'administrateur réseaux. Par défaut le framework vous permet d'utiliser 4 méthodes d'exposition de service:

Binding	Protocol	Encodage	Description
basicHttpBinding	HTTP	XML 1.0	Interopérable
wsHttpBinding	HTTP	XML 1.0	Sécurité,
wsDualHttpBinding	HTTP	XML 1.0	Sécurité, communication double sens
netTcpBinding	TCP	Binaire	Sécurité, communication double sens
netNamedPipeBinding	tubes nommés	Binaire	Local seulement, communication double sens
netMsmqBinding	Message windows	Binaire	Windows seulement

Le binding est donc défini par l'attribut "binding" de notre fichier de configuration.

III-C-3 - Définir le contract du service

Le "contract" (contrat en français) est tout simplement la liste des méthodes appellables par les clients. Rapellez vous, nous avons défini une interface que nous avons décoré avec les attributs [ServiceContract] et [OperationContract]. C'est cette interface que nous allons définir en tant que contrat pour notre client.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
 <system.serviceModel>
  <services>
   <service name="DvpDemoWCF.IMonPremierServiceWCF">
    <endpoint address="net.tcp://localhost:5000/DvpDemoWCF" binding="netTcpBinding"
 contract="DvpDemoWCF.IMonPremierServiceWCF"/>
   </service>
  </services>
 </system.serviceModel>
</configuration>
```

A noter qu'un contrat est **obligatoirement** une interface. Vous ne pouvez pas définir une classe en tant que contrat car le framework vous impose une séparation forte entre ce que les clients peuvent voir et ce que le serveur implémente. Ceci est une bonne chose qui vous permettra à la longue de ne pas vous mélanger entre des classes qui sont appelable par le client, des classes qui ne le sont pas et des interfaces de définition de services.

III-D - Conclusion

A travers cette présentation générale de WCF, vous avez pu constater que le modèle de programmation est bien plus simple que celui de .NET Remoting.

Copyright © 09/03/2007 - Vincent Lainé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Vous avez également découvert en quoi WCF peut vous être utile dans vos développements personnels.

Nous allons maintenant étudier 1 démonstration en profondeur afin d'appliquer les fondamentaux que nous venons de voir.

Copyright © 09/03/2007 - Vincent Lainé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

IV - Etude de cas : Service de déploiement à distance

Voyons les spécifications de notre service d'installation de logiciel à distance.

Ce logiciel doit tourner en tant que service Windows, il doit être transparent pour l'utilisateur. Il doit pouvoir interagir avec l'utilisateur en cas de besoin. Il doit envoyer le rapport de l'installation au serveur afin de maintenir la liste à jour.

Cependant il ne s'agit ici que d'un cas d'étude, nous n'allons pas mettre en oeuvre le service windows mais écrire des applications consoles.

IV-A - Structure du projet

Comme nous l'avons vu, WCF nous impose une certaine structure afin de fonctionner.

Notre projet se divise donc en trois projets au sens Visual Studio.

- SoftwareInstallerServerService : Programme console destiné à être installé sur un serveur afin de fournir aux clients les informations nécessaires à l'installation des logiciels.
- SoftwareInstallerClientService : Programme winform destiné aux clients. Il doit se connecter à intervalles réguliers au serveur afin de vérifier si des logiciels doivent être installés.
- SoftwareInstallerLibrary : Librairie contenant les classes communes et l'interface de définition du serveur.

Ces trois projets sont une base de travail mais l'on pourrait tout à fait imaginer que le service serveur soit implémenté en ASP.NET par exemple.

IV-B - SoftwareInstallerLibrary

Nous allons commencer par implémenter la librairie de définition du serveur. Comme nous l'avons vu cette définition intervient sous forme d'une interface décorée par **[ServiceContract]**.

```
/// <summary>
/// Interface de définition du service distant
/// </summary>
[ServiceContract]
public interface RemoteService
{

    /// <summary>
    /// Identifie le client sur le serveur.
    /// </summary>
    /// <param name="clientInformation">Les informations d'identification du client</param>
    [OperationContract]
 void IdentifyClient(ClientInformation clientInformation);

    /// <summary>
    /// Obtient la liste des logiciels à installer sur le client
    /// </summary>
    /// <returns></returns>
    [OperationContract]
 List<SoftwareData> GetSoftwareList();

    /// <summary>
    /// Renvois le rapport d'installation au serveur afin de centraliser l'information
    /// </summary>
```


Copyright © 09/03/2007 - Vincent Lainé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

```
    /// <param name="clientReport">le rapport du client sur l'installation des logiciels</param>
    [OperationContract]
 void UploadReport(ClientReport clientReport);

}
```

Nous avons donc défini notre interface. Nous avons aussi fait appel à deux classes personnelles que nous allons détailler.

La première est SoftwareData :

```
/// <summary>
/// Classe représentant un logiciel devant être installé sur le poste client
/// </summary>
[DataContract]
public class SoftwareData
{

    private string name;
    private string installerAddress;
    private Version version;

    /// <summary>
    /// Constructeur par défaut
    /// </summary>
    public SoftwareData()
    {

    }

    #region Proprietes

    /// <summary>
    /// Obtient ou défini le nom du logiciel à installer
    /// </summary>
    [DataMember]
    public string Name
    {
        get
        {
            return this.name;
        }

        set
        {
            this.name = value;
        }
    }

    /// <summary>
    /// Obtient ou défini l'adresse de l'installeur
    /// </summary>
    [DataMember]
    public string InstallerAddress
    {
        get
        {
            return this.installerAddress;
        }

        set
        {
            this.installerAddress = value;
        }
    }
```


Copyright © 09/03/2007 - Vincent Lainé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

```
        /// <summary>
        /// Obtient ou défini la version du logiciel à installer
        /// </summary>
        [DataMember]
        public Version Version
        {
            get
            {
                return this.version;
            }

            set
            {
                this.version = value;
            }
        }

        #endregion

    }
```

et la classe ClientReport :

```
    /// <summary>
    /// Classe définissant un rapport suite à l'installation de logiciel sur le client
    /// </summary>
    [DataContract]
    public class ClientReport
    {

        private DateTime date;

        [DataMember]
        public DateTime Date
        {
            get { return date; }
            set { date = value; }
        }

        private string clientName;

        public string ClientName
        {
            get { return clientName; }
            set { clientName = value; }
        }

        private List<SoftwareInstallReport> installReports;

        public List<SoftwareInstallReport> InstallReports
        {
            get { return installReports; }
            set { installReports = value; }
        }


    }
```

Vous remarquerez qu'il n'y a aucun code spécifique à WCF à part les attributs sur les classes et les membres que l'on souhaite exposer au client. Notre librairie partagée entre le client et le serveur est maintenant finie.

Nous passons donc à la création du serveur.

Copyright © 09/03/2007 - Vincent Lainé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

IV-C - SoftwareInstallerServerService

Ce serveur est un peu particulier. En effet il tourne en tant que programme console sur le PC serveur où il est installé. Il va sans dire qu'un service windows serait plus adapté mais ce n'est pas le sujet dans cet article. Une bonne partie de l'initialisation du serveur WCF va donc se faire directement dans la méthode Main du programme, mais cela n'est pas une bonne architecture ;-). De plus notre programme n'est pas "security friendly". En effet il demande les droits administrateurs pour fonctionner.

IV-C-1 - Création de la base de WCF

Nous allons écrire le code necéssaire à l'initialisation et la configuration du serveur WCF.

Pour commencer voyons le code d'initialisation du service serveur.

```
public void Main(string[] args)
{
    remoteService = new RemoteService();

    //Publication de la classe en singleton
    host = new ServiceHost(remoteService);

    //Ouverture du canal de communication
 host.Open();
}
```

Et le code d'arrêt du serveur :

```
//Fermeture du canal de communication
host.Close();
```

Ces quelques lignes de code suffisent à créer un canal de communication et à se mettre en attente d'une connexion d'un client. Il faut maintenant écrire le code du service proprement dit :

IV-C-2 - L'implémentation du service WCF

```
[ServiceBehavior(InstanceContextMode = InstanceContextMode.Single, ConcurrencyMode =
 ConcurrencyMode.Single)]
public class RemoteService : IRemoteService
{

    private Dictionary<string, ClientInformation> clientInformations;
    private List<string> clientAuthentified;

    /// <summary>
    /// Constructeur par défaut
    /// </summary>
    public RemoteService()
    {
        this.clientInformations = new Dictionary<string, ClientInformation>();
        this.clientAuthentified = new List<string>();

        //DEBUG
        ClientInformation ci = new ClientInformation();
        ci.ClientVersion = new Version("1.0.0.0");
```


Copyright © 09/03/2007 - Vincent Lainé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

```
            ci.Name = "dev01";

            this.clientInformations.Add(ci.Name, ci);
        }

        #region IRemoteService Members

        public void IdentifyClient(ClientInformation clientInformation)
        {
            if ( !clientInformations.ContainsKey(clientInformation.Name))
            {
                throw new Exception("Client inconnu !");
            }

            if (clientAuthentified.Contains(clientInformation.Name))
                throw new Exception("Client déjà authentifié");

            this.clientAuthentified.Add(clientInformation.Name);
        }

        public List<SoftwareData> GetSoftwareList(ClientInformation clientInformation)
        {
            if (this.clientAuthentified.Contains(clientInformation.Name))
            {
                //DEBUG
                SoftwareData soft = new SoftwareData();
                soft.InstallerAddress = "\\myserver\partage\setup.exe";
                soft.Name = "Test";
                soft.Version = new Version("0.0.0.1");

                List<SoftwareData> lists = new List<SoftwareData>();
                lists.Add(soft);

                return lists;
            }
            else
            {
                throw new Exception("Client non authentifié");
            }
        }

        public void UploadReport(ClientReport clientReport)
        {

        }

        #endregion
    }
```

La seule chose importante dans cette classe c'est l'attribut **[ServiceBehavior(InstanceContextMode = InstanceContextMode.Single, ConcurrencyMode = ConcurrencyMode.Single)]**. Cet attribut permet de définir que notre classe est prévue pour être publiée en tant que singleton et sur un seul thread. Le paramètre ConcurrencyMode permet de définir si un thread doit être créé a chaque fois qu'un client fait un appel ou si les appels doivent attendre que le précédent soit fini. Il existe une troisième option mais tout cela va plus loin que cette introduction.

IV-D - SoftwareInstallerClientService

Il ne reste plus qu'à écrire le programme client de notre projet.

```
    /// <summary>
    /// Service client récupérant la liste des softs à installer sur le client
```


Copyright © 09/03/2007 - Vincent Lainé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

```
    /// </summary>
    public partial class SoftwareInstallerClientService
    {

        private ChannelFactory<IRemoteService> channelFactory;
        private IRemoteService remoteSerice;
        private System.Timers.Timer timer;
        private ClientInformation clientInformation;

        public event UpdateNeededProgramEventHandler UpdateNeededProgram;

        public SoftwareInstallerClientService()
        {

        }

        public void Start()
        {
            channelFactory = new ChannelFactory<IRemoteService>("SI");
            channelFactory.Open();

            remoteSerice = channelFactory.CreateChannel();

            clientInformation = new ClientInformation();
            clientInformation.ClientVersion = new Version("1.0.0.0");
            clientInformation.Name = "dev01";
            clientInformation.UserName = Environment.UserName;

            //Envoi de l'authentification au serveur
            remoteSerice.IdentifyClient(clientInformation);

            //Timer toute les 5 minutes
            timer = new System.Timers.Timer(6000);
            timer.Elapsed += new ElapsedEventHandler(timer_Elapsed);
            timer.Enabled = true;

        }

        void timer_Elapsed(object sender, ElapsedEventArgs e)
        {
            List<SoftwareData> softs = remoteSerice.GetSoftwareList(clientInformation);

            if (this.UpdateNeededProgram != null)
                this.UpdateNeededProgram(softs);
        }

        public void Stop()
        {
            timer.Enabled = false;

            channelFactory.Close();
     }
    }
```

Le point important de notre client est la façon dont on va récupérer l'instance du service. En effet il existe plusieurs manières, la plus courante étant de passer par un générateur d'interface, outil également utilisé pour la consomation de web service. Mais je trouve cette façon de faire extrêmement non pratique. Je préfére largement déployer l'interface du service sur le client que regénérer des classes avec un outil (ici le SoapSuds fourni par le SDK du framework). Nous utilisons une classe du framework 3.0 qui s'appelle ChannelFactory qui vous permez de récupérer une instance correspondant à l'interface du contrat. Cette classe peut prendre un certain nombre de paramètres pour son contructeur mais le plus utile est de passer le nom du **endpoint** a utiliser pour créer cette instance. Ensuite un simple appel à **CreateChannel** vous récupérera tout ce qu'il faut pour utiliser de manière transparente votre service WCF.

Copyright © 09/03/2007 - Vincent Lainé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

V - Conclusion

Pour finir cet article d'introduction à Windows Communication Foundation je dirais que l'arrivée de cette nouvelle technologie est une très très bonne chose. Elle rationnalise et permet de fédérer les différentes méthodes de communications inter applications qu'elles soient locales ou distantes. De plus la simplicité est au rendez-vous alors pourquoi s'en priver ?

Par contre, il reste encore beaucoup de choses à connaître de ce fabuleux outil ;-).

Copyright © 09/03/2007 - Vincent Lainé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

VI - Téléchargements

Le runtime du framework 3.0 pour WinXP et Win 2003

Le SDK du framework 3.0 pour Win 2003, Win XP, Win Vista

Les extensions WPF et WCF pour Visual Studio 2005

Source **Les sources du projet de démo**

Copyright © 09/03/2007 - Vincent Lainé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

VII - Remerciements

Je tiens remercier Cardi pour la relecture orthographique et toute la rubrique dotnet pour son travail.

Copyright © 09/03/2007 - Vincent Lainé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

Copyright © 09/03/2007 - Vincent Lainé. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.