Active Directory & .Net 2.0

par Ronald VASSEUR

Date de publication : 26/10/2005

Dernière mise à jour :

Au cours de cet article, nous allons voir comment interagir avec l'annuaire Active Directory de Microsoft grâce au Framework .Net, en effet, ce dernier présente une base solide en la matière depuis sa version 1.1, mais offre de grandes nouveautés dans sa version 2.0, notamment grâce à son namespace System.DirectoryServices.ActiveDirectory. Nous nous attacherons, au cours de cet article, à effectuer des requêtes dans l'annuaire mais aussi à gérer les objets qu'il contient en procédant à des accès aussi bien en lecture qu'en écriture. L'autre point principal de cet article concernera la gestion d'une infrastructure Active Directory depuis du code .Net 2.0, nous parlerons alors contrôleurs de domaine, forêts, unités d'organisation, catalogues global ou encore rôles FSMO... et tout cela dans une application .Net, alors accrochez-vous ! Microsoft nous a particulièrement gaté !!!

Introduction

Avant d'entrer dans le vif du sujet, il convient de définir un certain nombre de termes comme par exemple, annuaire, Active Directory ou encore LDAP. Voyons de suite à quoi correspondent ces termes un peu "barbares".

Définition de LDAP

LDAP pour Lightweight Directory Access Protocol est un protocole d'annuaires reposant sur TCP/IP. Il a pour de nombreuses vocations mais les deux principales, selon moi, sont de standardiser et de régir les communications entre les clients et serveurs d'annuaires en offrant un protocole réseau d'accès, et de définir un espace de nommage qui permet d'identifier de manière unique chaque ressource. Ce protocole, dans sa version 3.0, est implémenté dans Active Directory, (si vous êtes motivés et surtout intéressés, je vous recommande de lire la RFC 3377 qui contient les spécifications de ce protocole, vous pouvez la retrouver à cette adresse http://www.ietf.org/rfc/rfc3377.txt). Je ne rentrerai pas plus dans les détails pour le moment mais je pense qu'il est important de bien connaître ce protocole, au même titre qu'il est important de connaître SQL lorsque l'on souhaite utiliser des SGBD.

Définition d'Active Directory

Active Directory est un ensemble de services d'annuaire se basant sur le protocole LDAP en version 3.0. Il sert à recenser les ressources présentes sur un réseau. Ces services d'annuaire sont intégrés aux versions Server de Windows depuis la version 2000. Ils ont pour fonction principale de centraliser l'administration des ressources des réseaux, même les plus grands, tout en apportant une sécurité élevée et une souplesse de déploiement et d'utilisation.

Active Directory est avant tout une base de données, qui contient la liste de l'ensemble des ressources (utilisateurs, ordinateurs, serveurs, partages réseau, imprimantes, etc.) proposées sur un réseau donné, typiquement une forêt composée de domaines (pour plus de détails sur ce qu'est un domaine ou encore une forêt, je vous renvoie à la documentation Technet dont vous trouverez les adresses en fin d'article dans la section "Ressources"). Par le biais de cet annuaire, il va nous être possible d'obtenir des informations détaillées sur chaque objet contenu dans la base, de les localiser, et également d'en avoir une vue centralisée et hiérarchique.

La fonction primaire d'Active Directory est de permettre aux utilisateurs, aux administrateurs mais également aux autres ressources elles-mêmes de récupérer des objets et leurs attributs au sein de cet annuaire (par exemple une imprimante possédant un module un recto-verso). La force d'un tel annuaire ne tient pas tellement dans sa capacité à stocker des objets représentant des ressources (bien que cela soit un point extrêmement important) mais plutôt dans ses capacités d'indexation et de recherche, d'ailleurs Active Directory, comme les autres annuaires de ce type, présente des performances en écriture relativement faibles, mais élevées en lecture, mais attention aux raccourcis trop faciles, si nous pouvons apparenter Active Directory à une base de données, il ne s'agît en aucun cas d'un SGBD tel Oracle ou SQL Server. L'annuaire de Microsoft, utilise le protocole LDAP pour effectuer des recherches ; mais, en plus de celui-ci, il repose sur d'autres protocoles réseaux comme par exemple TCP/IP, DNS ou encore Kerberos en version 5 pour l'authentification sécurisée des services et des utilisateurs.

Dernier aspect important d'Active Directory : la réplication. En effet, pour permettre une haute disponibilité, une sécurisation et de bonnes performances d'accès à son contenu, les informations sont répliquées sur tous les contrôleurs de domaine de l'étendue du réseau. C'est-à-dire qu'il existera plusieurs copies de l'annuaire sur un même réseau et qu'elles seront synchronisées pour répliquer toute modification faite au niveau de l'une de ses copies. L'organisation des domaines Active Directory repose sur une distribution précise des rôles, et la gestion de l'annuaire demande une configuration rigoureuse.

Voilà, nous en avons terminé pour cette rapide présentation d'Active Directory, mais attention ne vous méprenez

pas, il s'agit d'un outil très puissant et d'une grande complexité, le réduire à une définition basique serait une erreur, ses fonctionnalités sont innombrables et mériteraient plusieurs ouvrages, mon objectif était ici de vous définir globalement ce qu'est cet outil, qui à mon grand regret est trop souvent mal connu des développeurs. Il faut être conscient, que plus que connaître les classes du Framework que nous allons utiliser dans cet article, il faut connaître l'organisation et le fonctionnement d'Active Directory. Passons maintenant au sujet principal de cet article à savoir les interactions entre Active Directory et la technologie .Net dans sa version 2.0.

> ***Avertissement*** *: accéder au contenu de l'annuaire d'Active Directory et modifier la structure même de celui-ci est tout sauf anodin et vous devrez impérativement prendre les précautions qui s'imposent. De plus, il est encore plus aisé par le code que par l'interface graphique de mettre à terre toute votre architecture suite à une manipulation mal effectuée quand vous intervenez sur celle-ci avec des privilèges d'administratiion, intervenez donc de manière réfléchie, et sur un environnement de test avant toute application sur un environnement de production. Tous les exemples qui sont donnés dans cet article ont été testés sous Windows Server 2003 Enterprise Edition et sont fonctionnels, mais en aucun cas je ne pourrais être tenu responsable de quelque incident que ce soit provoqué par une mauvaise utilisation de ces exemples. Donc soyez prudents, mais je ne me fais pas de soucis pour cela, je sais que les personnes travaillant avec Active Directory le sont !*

1 - Etendue des possibilités du tandem Active Directory / .Net 2.0

1.1 - Présentation des namespaces du Framework

Le Framework .Net en version 1.0 offrait déjà un namespace (System.DirectoryServices) pour interagir avec les services d'annuaire reposant sur le protocole standardisé LDAP, ce n'est donc pas en soit une révolution du Framework 2.0 mais vous verrez au cours de cet article qu'il y a de notables améliorations qui vont faciliter la vie des développeurs que nous sommes. Voyons immédiatement les deux namespaces principaux que nous utiliserons tout au long de cet article.

1.1.1 - Namespace System.DirectoryServices

Ce namespace était déjà présent dans le Framework 1.1, mais il à évolué dans la version 2.0. Il permet d'accéder au service d'annuaire Active Directory par l'intermédiaire des ADSI (Active Directory Service Interfaces). Ce namespace contient une classe DirectoryEntry et DirectorySearcher qui vont respectivement vous permettre d'accéder à des objets dans l'annuaire et d'y effectuer des recherches. Pour pouvoir utiliser correctement ces différentes classes vous devez connaître au minimum les rudiments de la syntaxe des espaces de noms LDAP, sans quoi vous aurez du mal à accéder à vos différents objets, mais pas de panique vous allez voir que c'est assez simple et très loin d'être insurmontable.

Pour simplifier, ce namespace vous permet de gérer (lecture, ajout, suppression, modification...) les ressources de votre réseau et de les localiser dans l'annuaire.

1.1.2 - Namespace System.DirectoryServices.ActiveDirectory

Ce namespace mérite que l'on s'y attarde un peu plus et cela pour deux raisons : c'est une nouveauté du Framework .Net 2.0 et surtout il fait le bonheur des administrateurs systèmes souhaitant des applications de gestion de leur infrastructure Active Directory. Tout d'abord, comme son nom l'indique, cet espace de nom est dédié aux interactions de haut niveau avec le célèbre service d'annuaire de Microsoft. Grâce à un nombre très important de classes vous pouvez accéder à Active Directory au plan physique et logique. Vous pouvez accéder directement aux concepts de forêt, des domaines, des contrôleurs de domaine, des partitions, du schéma, du catalogue global, des sous réseaux, des sites... et je pourrais encore continuer la liste, ce namespace est simplement impressionnant, les spécialistes d'Active Directory, au travers de ces quelques lignes doivent déjà entrevoir les possibilités faramineuses (non, non je n'exagère pas) que nous laisse entrevoir cette nouveauté de la version 2.0 du Framework .Net.

Peut-être que tous ces concepts et ces explications vous semblent un peu théoriques, voyons donc maintenant quelles peuvent en être les applications concrètes.

1.2 - Quelles applications peut t'on envisager ?

Il existe de nombreuses applications différentes de ces fonctionnalités intégrées au Framework .Net, nous allons seulement en citer deux pour voir concrètement comment nous pouvons les utiliser.

1.2.1 - Sécurisation des accès à une application directement dans Active Directory

Le service d'annuaire d'Active Directory gérant entre autre les objets comptes d'utilisateurs, c'est à lui qu'il faut s'adresser pour vérifier si l'authentification demandée par un utilisateur (ou un service) est valide, c'est-à-dire en vérifiant notamment la validité du couple identifiant/mot de passe est exact, après cela dépend aussi (évidemment) de l'ACL (Access Control List) de l'objet pour lequel l'accès est demandé. Tout cela pour vous dire qu'une utilisation évidente des fonctionnalités d'interaction avec l'Active Directory est tout simplement l'authentification des utilisateurs d'une application (Web ou Windows) directement dans l'annuaire AD. Les avantages d'une telle solution sont multiples, surtout dans le cadre d'une utilisation au sein d'une entreprise :

- Une seule base de comptes utilisateurs est nécessaire, pas la peine d'avoir une base pour les comptes utilisateurs que l'on utilise pour accéder à son poste de travail et une autre pour les applications métiers que vous avez développez.
- Vous bénéficiez de la sécurité et de la disponibilité élevée offertes par le service d'annuaire de Microsoft.
- Les utilisateurs n'utilisent qu'un seul et unique compte, au-delà de l'aspect pratique de la chose, c'est surtout un gros avantage au plan de la sécurité, en effet, lorsque les utilisateurs ont trop de comptes utilisateurs, avec évidemment un mot de passe différent à chaque fois, ils ont la fâcheuse tendance à les noter par exemple sur un post-it, le plus généralement collé sur l'écran... bref pas la peine de vous faire un dessin, ce scénario fait bondir les DSI de toutes les entreprises.

1.2.2 - Gestion d'une infrastructure Active Directory

Je dois tout d'abord préciser ce que j'entends par gestion d'une infrastructure Active Directory, il s'agît en fait de la maintenance des services d'annuaire. Avec le Framework .Net 1.1 toutes ces opérations étaient possibles mais relativement complexes. Le Framework .Net 2.0 apporte quand à lui une grande nouveauté en la matière, je veux bien sûr parler du namespace System.DirectoryServices.ActiveDirectory. Il permet une gestion de haut niveau des composants d'une infrastructure Active Directory que sont par exemples : les domaines, les contrôleurs de domaine, le schéma, les rôles, les partitions... Ce namespace va donc permettre de développer des applications de maintenance, sur mesure, d'Active Directory.

De plus, la gestion d'une infrastructure Active Directory inclut bien évidemment la gestion des objets que sont les comptes utilisateurs, les comptes d'ordinateurs, les groupes, les stratégies de groupes, les unités d'organisation...

Les présentations sont maintenant terminées, passons à table :) !! Enfin presque...

2 - Ce que vous devez savoir avant de coder

J'ai volontairement placé cette partie avant de commencer à entrer dans le détail du code pour que vous connaissiez les choses importantes à savoir pour une compréhension plus aisée du code et du principe de fonctionnement de ce que nous allons voir au cours de cet article.

> ***Remarque*** *: il ne faut pas d'oublier d'ajouter une référence à l'assembly "System.DirectoryServices" dans votre projet, et également ajouter deux "Imports" en entête de votre classe : "Imports System.DirectoryServices" et "Imports System.DirectoryServices.ActiveDirectory". Sans cela vous ne pourriez pas utiliser les classes que nous allons employer tout au long de cet article.*

2.1 - Etablir une connexion avec l'annuaire

Pour nous connecter à Active Directory, nous allons utiliser la classe DirectoryEntry qui permet d'accéder à un objet dans l'annuaire. Pour nous connecter à un objet dans l'annuaire, il suffit de fournir son nom unique LDAP.

> ***Remarque*** *: un nom unique LDAP est un dérivé de la convention de nommage X500. Voilà un exemple de nom unique du compte utilisateur Administrateur du domaine Developpez.local : CN=Administrateur,CN=Users,DC=Developpez,DC=local. Attention cependant à ne pas confondre l'identifiant unique qui est attribué à un objet par Active Directory : le GUID (Globally Unique Identifier).*

Etablir une connexion

```
Dim monEntry As New DirectoryEntry("LDAP://" & cheminLdapConteneur, monUsername, monPassword,
AuthenticationTypes.Secure)
```

Comme vous pouvez le voir le constructeur de DirectoryEntry prends en paramètres le nom unique LDAP de l'objet, le nom d'ouverture de session et le mot de passe du compte que vous utilisez pour accéder à l'annuaire. Il prends en paramètre le type d'authentification à employer, au cours de cet article nous utiliserons systématiquement "secure" qui permet d'établir une connexion sécurisée à l'annuaire grâce au protocole Kerberos (car nous utilisons le "provider LDAP", et non "WINNT" par exemple). Si aucun identifiant ou mot de passe n'est donné, alors c'est le contexte de sécurité dans lequel s'exécute votre application qui est utilisé. Les opérations que nous allons effectuer nécessitent des droits de base quand il s'agît d'accéder en lecture seulement, et des droits administratifs lorsqu'il s'agît de modifier quoi que ce soit dans l'annuaire, mais il faut adapter cela à chaque situation, le niveau d'autorisation pourra varier, comme par exemple un utilisateur souhaitant modifier l'adresse inscrite dans son propre compte Active Directory.

2.2 - Membres principaux de la classe DirectorySearcher

- **ClientTimeOut** : spécifie la durée maximale durant laquelle le client effectuant la requête attend la réponse du serveur, au-delà de ce délai la recherche est annulée.
- **Filter** : cette propriété est une chaîne de caractères " au format LDAP " qui permet d'établir un filtre sur la recherche, c'est-à-dire que par exemple, nous allons pouvoir spécifier que nous recherchons par exemple des objets de type " user " (voici ce que cela donne dans ce cas précis : "(objectClass=user)"). Il existe de très nombreux types d'objets, je vous renvoie donc à la documentation officielle, idem pour la syntaxe à employer, mais rassurez vous, je vous donne tous les liens nécessaires dans la section ressources de cet article.
- **FindAll** : cette méthode exécute la recherché que vous avez paramétré, et renvoie l'ensemble des réponses qu'elle aura trouvé dans l'annuaire.
- **FindOne** : cette méthode comme la précédente exécute la recherche, mais à la différence prés, qu'elle ne renvoie que le premier résultat qu'elle aura pu trouver, et se stop immédiatement après.
- **PropertiesToLoad** : par le biais de cette propriété, il est possible de spécifier quelle est le type de propriétés qu'il est nécessaire de récupérer pour les objets recherchés, par exemple, si vous ne recherchez que la

propriété adresse des objets comptes d'utilisateurs, il n'est pas nécessaire et encore moins recommandé de récupérer toutes les propriétés, ainsi grâce à PropertiesToLoad, vous ne récupérez que les propriétés qui vous sont utiles.

- **SearchRoot** : cette propriété permet de spécifier ou de récupérer le noeud dans la hiérarchie Active Directory à partir duquel commence la recherche.
- **SearchScope** : la valeur que prend cette propriété indique si la recherche s'effectue uniquement dans le noeud de l'objet concerné par la recherche ou également dans tous les enfants de celui-ci. Par défaut la recherche est effectuée dans tous les noeuds "enfants".
- **ServerTimeLimit** : spécifie la durée maximale que doit prendre la recherche, dès que ce délai est atteint, elle est stoppée et seuls les objets jusqu'alors trouvés sont inclus dans le résultat renvoyés au client.
- **SizeLimit** : permet de spécifier le nombre maximal d'objets que peut retourner une recherche dans l'annuaire. La valeur par défaut est 0, ce qui signifie que le serveur retournera un nombre maximal de 1000 entrées si aucun paramètre contraire ne lui est précisé.

3 - Gestion des objets utilisateurs dans Active Directory

Dans cette partie, nous allons voir ce que nous offre le Framework .Net 2.0 dans la gestion des objets utilisateurs dans Active Directory, ces explications seront agrémentées de codes d'exemple. Commençons d'abord sur comment se connecter aux services d'annuaires Active Directory.

> ***Remarque*** *: tout au long de cet article, je vais parler d'objets comptes d'utilisateur, de compte d'utilisateur ou même d'utilisateur, sachez que ces trois types d'appellation recoupent la même notion, à savoir un objet de type compte d'utilisateur se trouvant dans l'annuaire Active Directory. J'ai simplement utilisé trois appellations différentes pour ne pas avoir des paragraphes trop indigestes.*

3.1 - Lecture de propriétés des comptes d'utilisateurs

Dans un premier temps, nous allons récupérer des attributs d'un objet utilisateur. Chaque objet possède de très nombreux attributs qui, dans le cas d'un utilisateur, peuvent être sa description, ou encore prénom. Je vous renvoie à la documentation du schéma d'Active Directory pour connaître la liste de tous les attributs de chaque objet, il en existe des centaines !

3.1.1 - Récupération de la valeur d'un paramètre

Voici la démarche à suivre pour effectuer une telle opération.

- Instancier un objet DirectoryEntry qui contiendra notre objet compte utilisateur, pour cela on fournit :

- Le chemin LDAP de l'objet utilisateur concerné
- Le compte AD utilisé pour se connecter

- Lecture et récupération du paramètre souhaité.

- Fermeture de la "connexion" établie par l'objet DirectoryEntry.

Ci-dessous, l'exemple d'une fonction permettant de récupérer le paramètre "displayName", qui correspond tout simplement au nom affiché dans l'annuaire pour ce compte utilisateur.

Récupération de l'attribut 'displayName' :

```
    Public Function getDisplayName(ByVal monCheminLdapUser As String, _
                                   ByVal monUsername As String, _
                                   ByVal monpassword As String) As String

            ' Exemple de chemin LDAP : "CN=User 1,CN=Users,DC=monDomaine,DC=local"

        Dim monDisplayName As String = Nothing

        Try
            ' Connexion à l'objet compte utilisateur souhaité
            Dim monUser As DirectoryEntry = New DirectoryEntry("LDAP://" & _
                                   monCheminLdapUser, monUsername, monpassword)

            ' Récupération de la valeur de la propriété
            monDisplayName = monUser.Properties("displayName").Value.ToString
            ' Fermeture de la "connexion"
            monUser.Close()

        Catch ex As Exception
```

Récupération de l'attribut 'displayName' :

```
                monDisplayName = ex.Message

            End Try

            Return monDisplayName

        End Function
```

3.1.2 - Récupération des membres d'un groupe

Comme vous le savez déjà certainement, Active Directory permet de créer des groupes (de sécurité ou de distribution) qui contiennent d'autres objets comme des comptes utilisateurs ou des comptes d'ordinateurs par exemple. L'avantage principal de l'utilisation des groupes est qu'ils permettent de centraliser et de regrouper des objets afin d'en faciliter l'administration, mais là n'est pas le thème de cet article donc je n'entrerai pas plus dans les détails. Voyons comment récupérer par le code la liste des utilisateurs contenus dans un groupe donné.

La démarche va être la suivante :

- Connexion à l'annuaire Active Directory
- Récupération de la propriété " members " d'un groupe
- Boucle pour parcourir la propriété " members "
- Renvoi du résultat

Exemple montrant comment lister les utilisateurs du groupe "Utilisateurs du domaine" :

Lister les membres d'un groupe :

```
        Public Function listeMembresGroupe(ByVal cheminLdapGroupe As String, _
                                           ByVal monUsername As String, _
                                           ByVal monPassword As String) As ArrayList

            ' Instanciation de la liste qui va contenir le résulat
            Dim maListeMembres As New ArrayList

            Try

                ' Groupe dont les membres sont à lister
                Dim monGroupe As New DirectoryEntry("LDAP://" & cheminLdapGroupe, _
                                                    monUsername, monPassword, AuthenticationTypes.Secure)

                Dim unMembre As New Object()
                ' Ajoute chaque membre trouvé à notre ArrayList à retourner
                For Each unMembre In monGroupe.Properties("member")
                    maListeMembres.Add(unMembre.ToString)
                Next

                monGroupe.Close()

            Catch ex As Exception

                MessageBox.Show(ex.Message)

            End Try

            ' Retourne la liste des membres du groupe
            Return maListeMembres

        End Function
```

3.1.3 - Liste des groupes dont un utilisateur est membre

Après avoir récupéré la liste des membres d'un groupe, intéressons nous aux utilisateurs, ainsi nous allons lister les groupes dont un utilisateur donné est membre. Pas la peine de faire plus de commentaires à ce sujet, le code suivant "parle" de lui-même :

Liste les groupes dont un utilisateur est membre

```
      Public Function isMemberOf(ByVal cheminLdapUser As String, _
                                 ByVal monUsername As String, _
                                 ByVal monPassword As String) As ArrayList

                ' Exemple de chemin LDAP : "CN=Administrateurs de
  l'entreprise,CN=Users,DC=monDomaine,DC=local"

          ' Instanciation la liste qui va contenir les groupes
          Dim maListeGroupes As New ArrayList

          Dim monUser As New DirectoryEntry("LDAP://" & cheminLdapUser, _
                                 monUsername, monPassword)

          ' Appel de la méthode "Groups" d'ADSI
          Dim mesGroupes As Object = monUser.Invoke("Groups")

          ' Récupération de la liste des groupes
          For Each unGroupe As Object In CType(mesGroupes, IEnumerable)
              Dim groupEntry As DirectoryEntry = New DirectoryEntry(unGroupe)
              maListeGroupes.Add(groupEntry.Name)
              MessageBox.Show(groupEntry.Name)
          Next

          Return maListeGroupes

      End Function
```

3.1.4 - Récupération des objets utilisateurs contenus dans une OU

Il s'agît ici de récupérer tous les utilisateurs contenus dans une OU, le chemin LDAP passé en paramètre pourrait être par exemple *"CN=MesUtilisateurs,DC=contoso,DC=local"*.

Récupérer les comptes utilisateurs contenus dans une OU :

```
      Public Function getUsersOU(ByVal monCheminLdapRecherche As String, _
                                 ByVal monUsername As String, _
                                 ByVal monpassword As String) As ArrayList

          ' ArrayList qui va contenir le résultat retourné par la recherche
          Dim maListeUsers As New ArrayList

          Try

              ' Instanciation d'un objet DirectorySearcher

              ' Définition de l'emplacement de recherche
              Dim monEmplacementRecherche As New DirectoryEntry("LDAP://" & _
                              monCheminLdapRecherche, monUsername, monpassword,
  AuthenticationTypes.Secure)

              Dim maRecherche As New DirectorySearcher(monEmplacementRecherche)

              ' dureeMax initisalisée à 25 secondes
              Dim dureeMax As New TimeSpan(0, 0, 25)

              ' Emplacement où la recherche doit être effectuée
              ' dans la hiérarchie Active Directory
              maRecherche.SearchRoot = monEmplacementRecherche

              ' Définition du Scope de la recherche, ici le conteneur
              ' seulement et tous ses "sous conteneur"
              maRecherche.SearchScope = SearchScope.Subtree

              ' Filtre uniquement les objets de type "user"
              maRecherche.Filter = "(objectClass=user)"

              ' Détermination de la propriété à récupérer lors de la recherche
```

Récupérer les comptes utilisateurs contenus dans une OU :

```
            maRecherche.PropertiesToLoad.Add("sAMAccountName")

            ' Durée maximum de la recherche
            maRecherche.ServerTimeLimit = dureeMax

            ' Fixe le nombre maximum d'objets retournés
            maRecherche.SizeLimit = 1500

            Dim unUtilisateur As DirectoryServices.SearchResult

            ' Récupération du 'sAMAccountName' des utilisateurs récupérés
            For Each unUtilisateur In maRecherche.FindAll()
                maListeUsers.Add(unUtilisateur.GetDirectoryEntry.Properties.Item("sAMAccountName").Value.ToStrin
            Next

            monEmplacementRecherche.Close()

        Catch ex As Exception
            MessageBox.Show(ex.Message)
        End Try

        Return maListeUsers

    End Function
```

***Attention** : ici, nous récupérons la propriété "sAMAccountName" donc celle-ci sera forcément toujours renseignée, puisqu'elle est obligatoire, mais pour les propriétés facultatives, il faudra faire attention, en effet, si l'une d'elles est nulle une exception sera alors levée. Il faut donc penser à gérer cela correctement dans son code, une simple vérification avec le mot clé "Nothing" suffira à éviter un crash assez ridicule de votre méthode !*

3.1.5 - Récupération du SID d'un objet compte d'utilisateur

SID signifie simplement Security Identifier, il s'agit en fait d'une chaîne de caractères alphanumériques unique qui permet d'identifier chaque objet (compte d'utilisateurs, groupes, ordinateurs...) dans une forêt Active Directory. Cette chaîne contient une partie fixe qui correspond au domaine d'appartenance et une partie " aléatoire et unique " qui est en fait le RID (Relative ID), et qui, elle, identifie directement l'objet au sein du domaine. Le rôle du SID est donc de permettre d'identifier de manière unique et formelle un objet dans une forêt, mais ce SID peut être modifié, dans le cas où, par exemple, l'objet auquel il se rattache est transféré d'un domaine à un autre. Il ne faut de plus pas confondre SID et GUID, ce dernier permet également d'identifier de manière unique un objet au sein d'une forêt Active Directory, mais le GUID quoi qu'il arrive ne sera jamais modifié, il est associé à un objet tout au long de son existence. Toutes les autorisations et les droits associés aux objets dans l'annuaire reposent sur le SID de chaque objet et non sur son nom comme on le croit trop souvent.

Voyons, dans un premier temps, comment récupérer ce SID dans l'annuaire, puis par la suite, nous verrons comment le formater pour l'afficher sous la forme que le connaisse tous les administrateurs.

Récupérer le SID d'un compte utilisateur :

```
    Public Function getUserSid(ByVal cheminLdapUser As String, _
                               ByVal monUsername As String, _
                               ByVal monPassword As String) As Byte()

        ' Tableau d'octets qui va contenir le SID
        Dim sidUser As Byte() = Nothing

        Try

            ' Récupération du SID de l'objet souhaité
            Dim monEntry As New DirectoryEntry("LDAP://" & cheminLdapUser, _
                                               monUsername, monPassword)

            sidUser = CType(monEntry.Properties("objectSid").Value, Byte())
```

Récupérer le SID d'un compte utilisateur :

```
            monEntry.Close()

        Catch ex As Exception

            MessageBox.Show(ex.Message)

        End Try

        ' Retour du SID dans un tableau d'octets sous sa forme primaire
        Return sidUser

    End Function
```

Vous avez vu qu'il est très simple de récupérer ce SID, par contre pour le formater correctement, vous allez voir que ce n'est pas aussi simple ! Enfin, rien d'insurmontable pour autant, rassurez-vous.

Formater un SID :

```
    Public Function formateSidToString(ByVal sidEnOctets As Byte()) As String

        Dim nombreSubauthority As Short = 0
        Dim monSid As StringBuilder = New StringBuilder

        monSid.Append("S-")

        Try
            monSid.Append(sidEnOctets(0).ToString)
            nombreSubauthority = Convert.ToInt16(sidEnOctets(1))


            If Not (sidEnOctets(2) = 0) OrElse Not (sidEnOctets(3) = 0) Then

                Dim strAuth As String = String.Format("0x{0:2x}{1:2x}{2:2x}{3:2x}{4:2x}{5:2x}", _
                                        CType(sidEnOctets(2), Int16), _
                                        CType(sidEnOctets(3), Int16), _
                                        CType(sidEnOctets(4), Int16), _
                                        CType(sidEnOctets(5), Int16), _
                                        CType(sidEnOctets(6), Int16), _
                                        CType(sidEnOctets(7), Int16))

                monSid.Append("-")
                monSid.Append(strAuth)

            Else

                Dim iVal As Int64 = CLng(CType((sidEnOctets(7)), Int32) & _
                                    CType((sidEnOctets(6) < 8), Int32) & _
                                    CType((sidEnOctets(5) < 16), Int32) & _
                                    CType((sidEnOctets(4) < 24), Int32))

                monSid.Append("-")
                monSid.Append(iVal.ToString)

            End If


            Dim idxAuth As Integer = 0
            Dim i As Integer = 0

            While i < nombreSubauthority

                idxAuth = 8 + i * 4

                Dim iSubAuth As UInt32 = BitConverter.ToUInt32(sidEnOctets, idxAuth)

                monSid.Append("-")
                monSid.Append(iSubAuth.ToString)

                i += 1

            End While


        Catch ex As Exception
```

Formater un SID :

```
                Return "Une erreur est survenue : " & ex.Message
            End Try
            Return monSid.ToString
        End Function
```

Ouf ! Voilà ce SID enfin correctement formaté !

Jusqu'à maintenant, nous avons vu comment lire des paramètres et des propriétés dans l'annuaire, maintenant voyons l'accès en écriture dans l'annuaire.

3.2 - Modification des objets compte utilisateur

Désormais nous savons lire des informations dans l'annuaire Active Directory, montons d'un cran sur l'échelle de l'intérêt que présentent ces fonctionnalités, et voyons maintenant comment écrire dans l'annuaire. Vous allez voir, qu'à condition de disposer des autorisations suffisantes, cela ne présente aucune difficulté particulière.

> ***Remarque*** *: pour pouvoir écrire dans l'annuaire Active Directory, vous devez être membre du groupe "Administrateur du domaine", ou alors avoir reçu les droits nécessaires de la part d'un administrateur.*

3.2.1 - Fixer la valeur de propriété d'un objet compte d'utilisateur

Nous allons voir ici comment modifier facilement la propriété "description" d'un objet compte utilisateur, mais le fonctionnement sera le même pour l'immense majorité des autres propriétés.

Modifier la description d'un utilisateur :

```
    Public Function modifDescriptionUser(ByVal monCheminLdapUser As String, _
                                         ByVal monUsername As String, _
                                         ByVal monPassword As String, _
                                         ByVal maValeur As String) As Boolean

        Try

            ' Connexion à l'objet utilisateur souhaité
            Dim monUser As DirectoryEntry = New DirectoryEntry("LDAP://" & _
                                                    monCheminLdapUser, monUsername, monPassword)
            ' Modification de la valeur
            monUser.Properties("description").Value = maValeur
            ' Validation des modifications
            monUser.CommitChanges()
            monUser.Close()

        Catch ex As Exception

            MessageBox.Show(ex.Message)
            Return False

        End Try

        Return True

    End Function
```

3.2.2 - Vérification de l'existence d'un objet utilisateur dans l'annuaire

Nous allons voir ici comment grâce à la classe DirectorySearch, il est possible de tester facilement l'existence d'un utilisateur. La fonction que vous allez voir ci-dessous, retourne "True" si l'utilisateur existe, et "False" évidemment s'il n'existe pas. Précédemment, nous avons suffisamment étudié le fonctionnement des recherches dans l'annuaire, je ne rentrerai donc pas plus dans les détails, les commentaires du code seront suffisants.

Vérification de l'existence d'un utilisateur :

```
    Public Function userExist(ByVal userAverifier As String, _
                              ByVal cheminLdapAexaminer As String, _
                              ByVal monUsername As String, _
                              ByVal monPassword As String) As Boolean

        Try
            ' Connexion à l'annuaire
            Dim monEntry As New DirectoryEntry(cheminLdapAexaminer, monUsername, _
                                               monPassword, AuthenticationTypes.Secure)
            Dim maRecherche As DirectorySearcher = New DirectorySearcher

            ' Paramétrage de la requête
            maRecherche.SearchRoot = monEntry
            maRecherche.Filter = "(&(objectClass=user) (cn=" + userAverifier + "))"

            ' Récupération du résultat de la requête
            Dim results As SearchResultCollection = maRecherche.FindAll()
            monEntry.Close()

            ' Analyse du résultat
            If results.Count = 0 Then
                Return False
            Else
                Return True
            End If

        Catch ex As Exception

            MessageBox.Show(ex.Message)

        End Try

    End Function
```

3.2.3 - Ajout ou suppression d'un utilisateur dans un groupe

Ajouter un utilisateur à un groupe

```
    Public Function addUserToGroup(ByVal cheminLdapGroup As String, _
                                   ByVal cheminLdapUser As String, _
                                   ByVal monUsername As String, _
                                   ByVal monPassword As String) As Boolean

        Dim monGroupe As New DirectoryEntry("LDAP://" & cheminLdapGroup, monUsername, _
                                            monPassword, AuthenticationTypes.Secure)
        Dim monUtilisateur As New DirectoryEntry("LDAP://" & cheminLdapUser, monUsername, _
                                            monPassword, AuthenticationTypes.Secure)

        Try
            ' Vérifie si l'utilisateur n'est pas déjà membre du groupe
            Dim estDejaMembre As Boolean = Convert.ToBoolean(monGroupe.Invoke("IsMember", _
                                                    New Object() {monUtilisateur.Path}))

            If Not estDejaMembre Then
                ' Utilisation de la méthode ADSI "Add" pour ajouter un utilisateur à un groupe
                monGroupe.Invoke("Add", New Object() {monUtilisateur.Path})
            Else
                MessageBox.Show("L'utilisateur " & monUtilisateur.Properties("cn")._
                                                Value.ToString() & " est déjà membre de ce groupe
!")
            End If

            Return True

        Catch ex As Exception

            MessageBox.Show(ex.Message)
```

Ajouter un utilisateur à un groupe

```
                Return False

            Finally

                ' Libération des ressources inutiles encore utilisées
                If Not IsNothing(monUtilisateur) Then
                    monUtilisateur.Dispose()
                End If

                If Not IsNothing(monGroupe) Then
                    monGroupe.Dispose()
                End If

            End Try

        End Function
```

Entre l'ajout et la suppression d'un objet compte utilisateur dans l'annuaire il n'y a finalement que des différences infimes, vous pouvez d'ailleurs le constater dans les deux codes exemples que je vous donne. Seule la méthode ADSI qui est appelée n'est pas la même (" remove ", au lieu de " add ", ainsi que l'ordre dans lequel est vérifiée l'appartenance au groupe par l'intermédiaire d'un booléen. Evidemment, pour supprimer un compte, on vérifie auparavant qu'il existe, et inversement pour ajouter un compte, on vérifie qu'il n'existe pas déjà avant de lancer l'opération.

Retirer un utilisateur d'un groupe :

```
        Public Function removeUserToGroup(ByVal cheminLdapGroup As String, _
                                          ByVal cheminLdapUser As String, _
                                          ByVal monUsername As String, _
                                          ByVal monPassword As String) As Boolean

            Dim monGroupe As New DirectoryEntry("LDAP://" & cheminLdapGroup, monUsername, _
                                monPassword, AuthenticationTypes.Secure)
            Dim monUtilisateur As New DirectoryEntry("LDAP://" & cheminLdapUser, monUsername, _
                                monPassword, AuthenticationTypes.Secure)

            Try
                ' Vérifie si l'utilisateur n'est pas déjà membre du groupe
                Dim estMembre As Boolean = Convert.ToBoolean(monGroupe.Invoke("IsMember", _
                                                    New Object() {monUtilisateur.Path}))

                If estMembre Then
                    ' Utilisation de la méthode ADSI "Add" pour ajouter un utilisateur à un groupe
                    monGroupe.Invoke("Remove", New Object() {monUtilisateur.Path})
                Else
                    MessageBox.Show("L'utilisateur " & monUtilisateur.Properties("cn")_
                                                    .Value.ToString() & " n'est pas membre de ce groupe
    !")
                End If

                Return True

            Catch ex As Exception

                MessageBox.Show(ex.Message)
                Return False

            Finally

                ' Libération des ressources inutiles encore utilisées
                If Not IsNothing(monUtilisateur) Then
                    monUtilisateur.Dispose()
                End If

                If Not IsNothing(monGroupe) Then
                    monGroupe.Dispose()
                End If

            End Try

        End Function
```

3.2.4 - Activation ou désactivation d'un compte utilisateur

Une pratique courante, et surtout indispensable au plan de la sécurité, est de désactiver chaque compte inutilisé pour une période suffisamment longue. Dans Windows Server, par le biais de la console "Utilisateurs et ordinateurs Active Directory", il ne suffit que d'un clic droit et d'un clic gauche pour désactiver ou activer un compte, vous allez pouvoir constater que le faire grâce à du code .Net est à peine plus long.

Activer un compte utilisateur :

```
    Public Function activeCompteUser(ByVal cheminLdapUser As String, _
                                     ByVal monUsername As String, _
                                     ByVal monPassword As String) As Boolean

        Try
            ' Connexion au compte utilisateur donné
            Dim monUtilisateur As DirectoryEntry = New DirectoryEntry("LDAP://" & cheminLdapUser, _
                                                   monUsername, monPassword, AuthenticationTypes.Secure)
            ' Récupération du flag indiquant l'état d'activation du compte
            Dim maValeur As Integer = CType(monUtilisateur.Properties("userAccountControl").Value,
Integer)
            ' Modification du flag pour activer le compte
            monUtilisateur.Properties("userAccountControl").Value = maValeur And Not 2
            ' Validation des modifications
            monUtilisateur.CommitChanges()
            Return True

        Catch ex As Exception

            MessageBox.Show(ex.Message)
            Return False

        End Try


    End Function
```

Après l'activation d'un compte utilisateur, voilà tout naturellement et sans surprise, la désactivation :

Désactiver un compte utilisateur :

```
    Public Function desactiveCompteUser(ByVal cheminLdapUser As String, _
                                        ByVal monUsername As String, _
                                        ByVal monPassword As String) As Boolean

        Try
            ' Connexion au compte utilisateur donné
            Dim monUtilisateur As DirectoryEntry = New DirectoryEntry("LDAP://" & cheminLdapUser)
            ' Récupération du flag indiquant l'état d'activation du compte
            Dim maValeur As Integer = CType(monUtilisateur.Properties("userAccountControl").Value,
Integer)
            ' Modification du flag pour désactiver le compte
            monUtilisateur.Properties("userAccountControl").Value = maValeur Or 2
            ' Validation des modifications
            monUtilisateur.CommitChanges()
            Return True

        Catch ex As Exception

            MessageBox.Show(ex.Message)
            Return False

        End Try


    End Function
```

3.2.5 - Déplacer un objet dans l'annuaire

Cet exemple vous montre comment déplacer un objet dans l'annuaire, vous pouvez ainsi, par exemple, déplacer

un utilisateur d'une OU à une autre en quelques lignes, bref, une fonctionnalité tout à fait indispensable pout n'importe quel administrateur système, surtout si on la couple à d'autres comme par exemple une recherche, imaginez plutôt : vous récupérez tous les utilisateurs d'un groupe donné dans une OU et vous les déplacez dans une autre en quelques lignes de code ! De quoi vous faire une application .Net de gestion d'annuaire très performante, et cela pour un très faible coût de développement.

Déplacer un objet compte d'utilisateur :

```
    Public Function moveUserFromTo(ByVal cheminLdapOrigine As String, _
                                   ByVal cheminLdapDestination As String, _
                                   ByVal monUsername As String, _
                                   ByVal monPassword As String) As Boolean

        Try

            ' Connexion à l'utilisateur à déplacer
            Dim monUserAdeplacer As New DirectoryEntry("LDAP://" & cheminLdapOrigine, _
                                                       monUsername, monPassword,
AuthenticationTypes.Secure)
            ' Déplacement de l'utilisareur vers son nouvel emplacement dans l'annuaire
            monUserAdeplacer.MoveTo(New DirectoryEntry("LDAP://" & cheminLdapDestination, _
                                                       monUsername, monPassword,
AuthenticationTypes.Secure))

            monUserAdeplacer.Close()

            Return True

        Catch ex As Exception

            MessageBox.Show(ex.Message)
            Return False
        End Try

    End Function
```

3.3 - Suppression d'objets comptes d'utilisateurs dans l'annuaire

Une autre règle de base en sécurité, après la désactivation des objets momentanément inutiles, est de supprimer ce dont on est sûr qu'ils ne serviront plus par la suite. Et comme pour tout ce que l'on a vu jusqu'ici, le namespace System.DirectoryServices et ADSI vont nous venir en aide. Ce code s'applique aux comptes utilisateurs mais également à n'importe quel autre objet de l'annuaire, à la condition que l'objet à supprimer n'ai pas d'objets enfants, comme cela peut être le cas par exemple pour une OU, l'objet ne doit donc contenir aucun autre objet, si ce n'est pas le cas, il faut utiliser une autre méthode pour supprimer cet objet.

Supprimer un objet compte d'utilisateur :

```
    Public Function deleteUser(ByVal cheminLdapParent As String, _
                               ByVal cheminLdapObjetAsupprimer As String, _
                               ByVal monUsername As String, _
                               ByVal monPassword As String) As Boolean

        ' Noeud parent de l'objet à supprimer
        Dim entryParent As New DirectoryServices.DirectoryEntry("LDAP://" & cheminLdapParent, _
                                   monUsername, monPassword, AuthenticationTypes.Secure)

        ' Objet à supprimer
        Dim entryAsupprimer As New DirectoryServices.DirectoryEntry("LDAP://" &
cheminLdapObjetAsupprimer, _
                                   monUsername, monPassword, AuthenticationTypes.Secure)

        Try

            ' Suppression de l'objet
            entryParent.Children.Remove(entryAsupprimer)
            Return True

        Catch ex As Exception

            MessageBox.Show(ex.Message)
```

Supprimer un objet compte d'utilisateur :

```
                Return False

            Finally

                ' Fermeture des DirectoryEntry
                entryParent.Close()
                entryAsupprimer.Close()

            End Try

        End Function
```

3.4 - Création d'objets comptes d'utilisateurs dans l'annuaire

Enfin ne manquait plus qu'une fonctionnalité principale en ce qui concerne la gestion des comptes utilisateurs depuis .Net, il s'agît bien évidemment de la création de comptes. En la matière, comme d'ailleurs pour toute la gestion de l'Active Directory que ce soit en mode graphique ou ligne de commande, Microsoft à particulièrement gaté les administrateurs systèmes, mais .Net est en quelque sorte la troisième voie, et, à mon avis, la plus prometteuse pour les années à venir, surtout pour les plus grosses infrastructures Active Directory, mais tout ceci n'engage que moi. Sans plus tarder passons au code commenté qui nous montre comment réaliser cette opération :

Créer un compte utilisateur :

```
        Public Function createUser(ByVal monCheminLdap As String, _
                                   ByVal monUsername As String, _
                                   ByVal monpassword As String, _
                                   ByVal nouvelUser As String) As Boolean

            Try
                Dim monEmplacement As DirectoryEntry = New DirectoryEntry("LDAP://" & _
                                                monCheminLdap, monUsername, monpassword,
    AuthenticationTypes.Secure)

                Dim monUser As DirectoryEntry = monEmplacement.Children.Add("cn=" & nouvelUser, "user")

                ' Nom d'ouverture de session pour les systèmes antérieurs à Windows 2000
                monUser.Properties("sAMAccountName").Add("userDeveloppez")

                ' Nom affiché dans l'annuaire
                monUser.Properties("displayName").Add("Mr Developpez")

                ' Adresse de l'utilisateur titulaire de ce compte
                monUser.Properties("street").Add("1 rue Developpez.com")

                ' Nom d'ouverture de session de l'utilisateur
                monUser.Properties("UserPrincipalName").Add("userDeveloppez@contoso.local")

                ' Description du compte utilisateur
                monUser.Properties("description").Add("Utilisateur de test")

                ' Création de l'objet utilisateur et application des paramètres définis ci-dessus
                monUser.CommitChanges()

                'monUser.NativeObject.AccountDisabled = False

                ' Création du mot de passe pour ce compte utilisateur
                monUser.Invoke("SetPassword", New Object() {"moTdePassE012"})
                monUser.CommitChanges()

                ' L'utilisateur devra changer son mot de passe à sa prochaine connexion
                monUser.Properties("pwdLastSet").Value = 0
                monUser.CommitChanges()

                Return True

            Catch ex As Exception

                MessageBox.Show(ex.Message)
                Return False
```

Créer un compte utilisateur :

```
            End Try

        End Function
```

***Remarque** : une grande partie de la sécurité de vos comptes utilisateurs repose sur la robustesse et la confidentialité des mots de passe, vous pouvez constater que dans le code ci-dessus, je donne le mot de "passe en dur" ce qui n'est absoluement pas souhaitable. Je vous conseille donc de créer une méthode permettant de générer des mots de passe à la volée et surtout de manière aléatoire. La complexité ou le fait que ce mot de passe soit généré automatiquement n'est en aucun cas un problème puisque tout administrateur sensé forcera l'utilisateur à changer son mot de passe lors de la première ouverture de session. Le Framework .Net vous permettra facilement de créer des mots de passe à la volée respectant les impératifs de complexité et de longueur en vigueur sur votre réseau. Donc pour résumer : n'entrez pas vos mots de passe à la main dans votre code, mais faites le générér automatiquement !*

***Remarque** : vous trouverez tous les liens vers la documentation nécessaire dans la section "Ressources" de cet article, cela concerne entre autre la valeur, le rôle des différents "flags" ainsi que leur fonctionnement.*

Après avoir étudié comment obtenir ou modifier des propriétés d'un objet compte utilisateur dans Active Directory, voyons maintenant ce qu'il en est pour les autres objets présents dans l'annuaire.

4 - Opérations sur d'autres types objets de l'annuaire

4.1 - Pas de différence notable

Que cela concerne des objets de type comptes d'utilisateurs ou n'importe quel autre type d'objet les opérations de base ne changent pas. Pour travailler sur d'autres types d'objet il faudra surtout penser à configurer le filtre de la recherche avec la bonne classe d'objet et veiller à accéder aux bonnes propriétés des différents objets stockés dans l'annuaire, je vous recommande d'ailleurs à ce sujet d'utiliser l'utilitaire de Microsoft ADSIedit qui permet vraiment d'entrer dans le détail et la composition de chaque objet. Attention cependant, cet utilitaire est extrêmement puissant mais il faut prendre garde à ce que l'on fait avec ! En effet, en deux clics de souris vous pouvez potentiellement mettre à genoux votre infrastructure Active Directory, donc prenez garde, ce n'est pas un outil à mettre entre toutes les mains.

> *Cet utilitaire n'est pas installé par défaut, mais il se trouve sur le CD de votre Windows Server dans le répertoire "\support\tools". En installant le suptools.msi vous aurez ainsi accès à de trés nombreux outils qui vous permettront d'administrer de manière trés productive votre serveur.*

4.2 - Lister les objets ordinateurs contenus dans une OU

Pour vous montrer à quel point le fonctionnement reste le même quelque soit le type d'objet concerné, je vous propose un exemple montrant comment lister tous les comptes d'ordinateurs se trouvant dans une OU donnée.

Lister ordinateurs contenus dans une OU

```
    Public Function getListeComputerContainer(ByVal conteneurAinspecter As String, _
                                               ByVal monUsername As String, _
                                               ByVal monPassword As String) As ArrayList

            ' Exemple de chemin vers un conteneur : "OU=monOU,DC=contoso,DC=local"


        ' Liste qui sera retournée avec les noms des ordinateurs
        ' trouvés lors de la recherche
        Dim maListeComputers As New ArrayList

        Try

            ' Définition de la DirectoryEntry pour accéder à l'annuaire
            Dim monEntry As DirectoryEntry = New DirectoryEntry("LDAP://" & _
                           conteneurAinspecter, monUsername, monPassword,
AuthenticationTypes.Secure)

            Dim maRecherche As DirectorySearcher = New DirectorySearcher(monEntry)

            ' Filtre spécifiant que l'on recherche des objets de type 'computer'
            maRecherche.Filter = ("(objectClass=computer)")

            Dim unComputer As SearchResult = Nothing

            ' Boucle permettant de placer dans un liste chaque nom d'ordinateur
            ' retourné par la recherche.
            For Each unComputer In maRecherche.FindAll
                maListeComputers.Add(unComputer.GetDirectoryEntry.Name)
            Next

        Catch ex As Exception

            MessageBox.Show(ex.Message)

        End Try

        Return maListeComputers

    End Function
```

5 - Gestion d'Active Directory

Nous venons de voir ce qui concerne spécifiquement la gestion des objets de l'annuaire comme les comptes utilisateurs, tout cela était à peu prés existant dans le Framework .Net 1.1, à quelques évolutions prés. Cette nouvelle partie va maintenant concerner la nouveauté en la matière: je veux bien évidemment parler du namespace System.DirectoryServices.ActiveDirectory. Comme son nom ne l'indique absolument pas (lol) ce namespace a été spécifiquement développé pour Microsoft Active Directory à la différence de System.DirectoryServices qui est un namespace supportant tous les annuaires conforme à LDAP 3.0. Ce namespace est entièrement nouveau, et fait son apparition dans la désormais célèbre version 2.0 du Framework .Net, il va grandement faciliter le développement d'applications de gestion d'infrastructure Active Directory, mais plutôt que d'en parler voyons de quoi il est capable, concrètement, dans le code.

5.1 - Les domaines et la forêt

5.1.1 - Définition des concepts de domaines et de forêt

Un domaine est une entité logique de sécurité délimitée dans le cadre d'un réseau pouvant être situé sur plusieurs sites physiques. Il regroupe un ensemble de ressources qui seront administrées de manière centralisée par le biais des contrôleurs de domaine. Le domaine Active Directory (différent du domaine Windows NT) est basé sur le protocole DNS, ainsi un domaine sera délimité par un nom DNS commun, par exemple "mondomaine.local"; si l'on continue avec cet exemple un serveur nommé serveur01 aura un nom DNS serveur01.mondomaine.local, et cela est de même pour chaque ressource.

Une forêt est un ensemble de domaines qui sont regroupés eux-mêmes en arbres avec des noms DNS contigus, il s'agît de la plus grande entité de l'infrastructure Active Directory, depuis Windows Server 2003, il est possible d'établir des relations d'approbations entre des forêts, ce qui repousse un peu plus encore la taille maximale de votre Active Directory. Une forêt est donc une entité administrative qui est composée de domaines qui peuvent être regroupés en arbres.

5.1.2 - Les contrôleurs de domaine

L'administration des domaines Active Directory est donc centralisée et se réalise grâce aux contrôleurs de domaines. Il doit donc en avoir au minimum un par domaine, s'il y en a plusieurs alors ils se répartissent les rôles FSMO et organisent la réplication de l'annuaire sur chacun d'eux pour en avoir une copie synchronisée. Nous allons voir ici comment lister l'ensemble des contrôleurs de domaines, le nouveau namespace : System.DirectoryServices.ActiveDirectory nous facilite grandement la tâche, seules quelques lignes de code sont nécessaires à cette opération.

Lister les contrôleurs de domaine

```
    Public Function getListeDC() As DomainControllerCollection

        Dim listeDC As DomainControllerCollection = Nothing

        ' Définition du context de connexion : connexipn au domaine courant
        Dim monContext As New DirectoryContext(DirectoryContextType.Domain, _
                                    Domain.GetCurrentDomain.ToString)

        Dim monDomaine As Domain = Domain.GetDomain(monContext)

        Try

            ' Récupération de la liste des contrôleurs de domaine
            listeDC = monDomaine.FindAllDomainControllers()
```

Lister les contrôleurs de domaine
```
            Catch ex As Exception

                MessageBox.Show(ex.Message)

            End Try

            Return listeDC

        End Function
```

***Remarque** : j'ai constaté un bug qui se produisait avec les versions Fr de Windows Server 2003 lors de l'utilisation de l'objet **DirectoryContext**, en effet à chaque instanciation de cet objet une exception était levée pour une propriété dnsHostName non renseignée. Aucun problème particulier n'a, par contre, était constaté avec les versions US de Windows Server 2003. Gageons que ce problème ne sera plus présent dans les versions finales ! En effet, cet article à entièrement été réalisé avec Visual Studio 2005 Beta 2 et le Framework .Net 2.0 également en version beta 2.*

5.1.3 - Informations sur la forêt

Voyons maintenant comment récupérer des informations sur la forêt Active Directory courante. Je me contenterai du seul code commenté tant ces opérations ne présentent aucune difficulté !

- Nom de la forêt :

Récupérer le nom de la forêt :
```
        Public Function getNomForet() As String

            ' Définition d'une variable forêt
            Dim maForet As Forest = Nothing

            Try

                ' Récupération de la forêt courante
                maForet = Forest.GetCurrentForest

            Catch ex As Exception

                MessageBox.Show(ex.Message)

            End Try

            Return maForet.ToString

        End Function
```

- Liste des domaines de la forêt :

Lister les domaines de la forêt :
```
        Public Function getDomainesForet() As DomainCollection

            ' Collection qui contiendra la liste des domaine
            Dim mesDomaines As DomainCollection = Nothing

            Try

                ' Récupération de la liste des domaines de la forêt courante
                mesDomaines = Forest.GetCurrentForest.Domains

            Catch ex As Exception

                MessageBox.Show(ex.Message)

            End Try

            Return mesDomaines
```

Lister les domaines de la forêt :

```
        End Function
```

- Niveau fonctionnel de la forêt :

Le niveau fonctionnel de la forêt correspond (pour faire très simple) au niveau de fonctionnalité qu'autorise la fôret. Ainsi, en fonction de la présence dans votre fôret de contrôleurs de domaines fonctionnant sous Windows NT, 2000 Server ou Server 2003, vous n'aurez pas accès aux mêmes fonctionnalités, tout cela évidemment pour des raisons de compatibilités et en définitive de permettre la cohabitation de contrôleurs de domaines tournant avec des versions différentes de Windows. Par défaut, une fôret à un niveau fonctionnel "Windows 2000". Je vous propose ici de détecter grâce à ce code quel est le niveau fonctionnel actuel de votre forêt.

Trouver le niveau fonctionnel de la forêt :

```
        Public Function getNiveauForet() As ForestMode

            Dim monMode As ForestMode = Nothing

            Try
                ' Récupération du niveau fonctionnel de la forêt
                monMode = Forest.GetCurrentForest.ForestMode

            Catch ex As Exception

                MessageBox.Show(ex.Message)

            End Try

            Return monMode

        End Function
```

5.1.4 - Niveau fonctionnel du domaine

Un domaine, au même titre qu'une fôret, possède un niveau fonctionnel, il en existe plusieurs qui déterminent chacun des niveaux de fonctionnalités différents, vous pouvez si vous le désirez augmenter le niveau fonctionnel d'un domaine si vous souhaitez disposer de fonctionnalités et d'une sécurité encore plus avancées, mais cela impose également quelques contraintes. Le code que je vous propose ici, vous permet de détecter le niveau fonctionnel d'un domaine.

Trouver le niveau fonctionnel du domaine :

```
        Public Function getNiveauDomaine() As DomainMode

            ' Connexion au domaine courant
            Dim monContext As New DirectoryContext(DirectoryContextType.Domain, _
                                        Domain.GetCurrentDomain.ToString)

            Dim monDomaine As Domain = Domain.GetDomain(monContext)
            Dim modeDomaine As DomainMode

            Try

                ' Récupération du niveau du domaine
                modeDomaine = monDomaine.DomainMode

            Catch ex As Exception

                MessageBox.Show(ex.Message)

            End Try

            Return modeDomaine

        End Function
```

Passons maintenant au processus de réplication, processus capital d'Active Directory.

5.1.5 - Gestion de la réplication

La réplication, est le processus vital au fonctionnement d'une infrastructure Active Directory. Pour simplifier, c'est grâce à la réplication que nous pouvons avoir une copie identique et synchronisée de l'annuaire sur l'ensemble du domaine. Ce processus est d'une trés grande complexité et, selon moi, peut être ce qu'il y a de plus sensible à gérer dans Active Directory. Malgré cette grande complexité, le nouveau namespace du Framework .Net 2.0 nous permet de gérer en partie ce processus. Voyons immédiatement un exemple vous permettant de récupérer la liste des actions de réplications n'ayant pas pu être effectuées :

Actions de réplications non réalisées :

```
    Public Function operationsReplicationNonEffectuees(ByVal nomDC As String) _
                                                As ReplicationOperationCollection
        ' nomDC : nom DNS d'un contrôleur de domaine

        Dim listeOperations As ReplicationOperationCollection = Nothing

        Try

            ' Connexion au contrôleur de domaine spécifié en paramètre
            Dim monContext As New DirectoryContext(DirectoryContextType.DirectoryServer, nomDC)
            Dim monDC As DomainController = DomainController.GetDomainController(monContext)
            ' Récupération des opérations de réplication non effectuées
            listeOperations = monDC.GetReplicationOperationInformation.PendingOperations()

        Catch ex As Exception

            MessageBox.Show(ex.Message)

        End Try

        Return listeOperations

    End Function
```

Voyons maintenant une opération encore plus intéressante pour les administrateurs, initier la réplication d'une partition donnée d'un contrôleur de domaine vers un autre :

initier la réplication d'une partition :

```
    Public Function initierReplication(ByVal nomDCorigine As String, _
                                       ByVal nomDCdestination As String, _
                                       ByVal partitionArepliquer As String) As Boolean

        ' nomDCorigine : nom DNS contrôleur de domaine de
        '       contenant la partition à répliquer
        ' nomDCdestination : nom DNS du contrôleur de domaine
        '      où la partition doit être répliquée
        ' partitionArepliquer : nom distingué de la partition
        '      à répliquer

        Try

            Dim monContext As New DirectoryContext(DirectoryContextType.DirectoryServer,
nomDCdestination)
            Dim monDC As DomainController = DomainController.GetDomainController(monContext)
            ' Initiation de la réplication
            monDC.SyncReplicaFromServer(partitionArepliquer, nomDCorigine)

            Return True

        Catch ex As Exception

            MessageBox.Show(ex.Message)
            Return False

        End Try

    End Function
```

5.2 - Gestion des "Organizational Units" ou OU

Les Unités d'Organisation (pour "Organizational Units" (OU)) sont en fait des conteneurs que l'on utilise pour y placer des objets comptes d'utilisateurs, des comptes d'ordinateurs, des groupes, ou encore tout autres types de ressources... C'est sur eux que repose en très grande partie le déploiement des stratégies de groupe, pierre angulaire de l'administration centralisée d'Active Directory.

5.2.1 - Lister les OU dans un annuaire

Les OU sont gérées depuis la console "Ordinateurs et utilisateurs Active Directory", vous pouvez par le biais de celle-ci réaliser un trés grand nombre d'opérations, voyons ici un exemple commenté en détail qui vous montre comment lister l'ensemble des OU contenues dans l'annuaire.

Lister les Organizational Units (OU) :

```
      Public Function listerOU(ByVal monCheminLdapRecherche As String, _
                               ByVal monUsername As String, _
                               ByVal monPassword As String) As ArrayList

          ' ArrayList qui va contenir le résultat retourné par la recherche
          Dim maListeOU As New ArrayList

          Try


              ' Définition de l'emplacement de recherche
              Dim monEmplacementRecherche As New DirectoryEntry("LDAP://" & _
                                monCheminLdapRecherche, monUsername, monPassword,
  AuthenticationTypes.Secure)

              ' Instanciation d'un objet DirectorySearcher
              Dim maRecherche As New DirectorySearcher(monEmplacementRecherche)

              ' dureeMax initisalisée à 45 secondes
              ' Il s'agît de la durée maximale que prendra la requête
              Dim dureeMax As New TimeSpan(0, 0, 45)

              ' Emplacement où la recherche doit être effectuée
              ' dans la hierarchie Active Directory
              maRecherche.SearchRoot = monEmplacementRecherche

              ' Définition du Scope de la recherche, ici le conteneur
              ' seulement et tous ses "sous conteneur"
              maRecherche.SearchScope = SearchScope.Subtree

              ' Filtre uniquement les objets de type "organizationalUnit"
              maRecherche.Filter = "(objectClass=organizationalUnit)"

              ' Détermination de la propriété à récupérer lors de la recherche
              maRecherche.PropertiesToLoad.Add("name")

              ' Durée maximum de la recherche
              maRecherche.ServerTimeLimit = dureeMax

              ' Fixe le nombre maximum d'objets retournés
              maRecherche.SizeLimit = 1500

              Dim uneOU As DirectoryServices.SearchResult

              ' Récupération du 'sAMAccountName' des utilisateurs récupérés
              For Each uneOU In maRecherche.FindAll()
                  maListeOU.Add(uneOU.GetDirectoryEntry.Properties.Item("name").Value.ToString)
                  MessageBox.Show((uneOU.GetDirectoryEntry.Properties.Item("name").Value.ToString))
              Next

              monEmplacementRecherche.Close()

          Catch ex As Exception
              MessageBox.Show(ex.Message)
          End Try
```

Lister les Organizational Units (OU) :
```
            Return maListeOU

        End Function
```

Vous voyez qu'il s'agît d'une simple requête réalisée grâce au DirectorySearcher, j'ai volontairement utilisé un grand nombre d'option pour vous montrer que si l'utilisation de cet objet peut être basique, il est également possible de réaliser des recherches précises sous de nombreuses conditions.

5.2.2 - Créer des OU dans un annuaire

Voici maintenant un exemple commenté qui vous montre comment créer des unités d'organisation dans Active Directory :

Créer des Organizational Units (OU) :
```
        Public Function createOU(ByVal monCheminLdap As String, _
                                 ByVal monUsername As String, _
                                 ByVal monPassword As String, _
                                 ByVal nouvelleOU As String) As Boolean

            Try

                ' Connexion au noeud de l'annuaire où va être ajoutée l'OU
                Dim monEmplacement As DirectoryEntry = New DirectoryEntry("LDAP://" & _
                                monCheminLdap, monUsername, monPassword, AuthenticationTypes.Secure)

                ' Ajout de l'OU
                Dim monOU As DirectoryEntry = monEmplacement.Children.Add("OU=" & nouvelleOU, _
                                                                          "organizationalUnit")

                ' Validation des modifications
                monOU.CommitChanges()

                Return True

            Catch ex As Exception

                MessageBox.Show(ex.Message)
                Return False

            End Try

        End Function
```

5.3 - Gestion des rôles FSMO

5.3.1 - Définition des rôles FSMO

Les rôles de maîtres d'opératiion ou FSMO (Flexible Single Master Operation) correspondent à un certain nombre de rôles qui sont pris en charge par un seul et unique contôleur de domaine au sein d'un domaine ou d'une forêt, et cela, en fonction du rôle concerné. Je n'entrerai pas dans les détails, mais sachez que le but principal de la définition des rôles déténus par des contrôleurs de domaine uniques à pour objet la protection du schéma de l'annuaire et d'assurer en quelque sorte l'intégrité référencielle d'Active Directory. Voici la liste des 5 rôles FSMO que l'on trouve dans un environnement Active Directory sous Windows Server 2003.

Rôle FSMO	Présentation du rôle
Contrôleur de schéma	C'est lui qui gère et réalise toutes les modifications du schéma Active Directory, il n'en existe qu'un seul dans la fôret.
Maître d'attribution de noms de domaine	Il gère l'ajout et la suppression de domaines dans la

Rôle FSMO	Présentation du rôle
	fôret, il n'en existe qu'un seul dans la fôret.
Maître RID	Son rôle est d'allouer des séquences de RID (ID Relatifs) aux contrôleurs de domaine de son domaine, il en existe un par domaine.
Maître d'émulateur PDC	Il émule un contrôleur principal de domaine (PDC) pour les serveurs et clients qui exécutent une version de Windows antérieure à Windows 2000, on en trouve un par domaine de la fôret.
Maître d'infrastructure	Il est le garant de la cohérence des références des objets de son domaine sur les objets des autres domaines, il est une pierre angulaire du fonctionnement d'Active Directory sur des infrastructures à domaines multiples, il en existe un pour chaque domaine de la fôret.

5.3.2 - Localisation des rôles

On l'a bien compris, les rôles FSMO s'exécutent au mieux sur un conrôleur de domaine par domaine ou même par fôret suivant le rôle concerné, ces contrôleurs de domaine ont donc une importance stratégique pour le fonctionnement de votre infrastructure Active Directory, c'est pourquoi il est utile de pouvoir localiser quel serveur est titulaire de chacun de ces rôles. Je vous propose donc un petit morceau de code qui vous permettra d'indentifier quel contrôleur de domaine est titulaire de quel rôle.

Localiser les rôles FSMO dans une forêt et un domaine :

```
    Public Function getListeRolesDC() As ArrayList

        Dim maListeRoles As New ArrayList
        Dim monDomaine As Domain = Domain.GetCurrentDomain

        Dim maitreInfrastructure As String = Nothing
        Dim maitreRID As String = Nothing
        Dim emulateurPDC As String = Nothing
        Dim maitreAttributionNoms As String = Nothing
        Dim controleurSchema As String = Nothing

        Try

            ' Maître d'infrastructure
            maitreInfrastructure = monDomaine.InfrastructureRoleOwner.ToString
            maListeRoles.Add("Maître d'infrastructure : " & maitreInfrastructure)

            ' Maître  RID
            maitreRID = monDomaine.RidRoleOwner.ToString
            maListeRoles.Add("Maître  RID : " & maitreRID)

            ' Emulateur PDC
            emulateurPDC = monDomaine.PdcRoleOwner.ToString
            maListeRoles.Add("Emulateur PDC : " & emulateurPDC)

            ' Maître d'attribution de noms de domaine
            maitreAttributionNoms = monDomaine.Forest.NamingRoleOwner.ToString
            maListeRoles.Add("Maître d'attribution de noms de domaine : " & maitreAttributionNoms)

            ' Contrôleur de schéma
            controleurSchema = monDomaine.Forest.SchemaRoleOwner.ToString
            maListeRoles.Add("Contrôleur de schéma : " & controleurSchema)

        Catch ex As Exception

            MessageBox.Show(ex.Message)

        End Try

        Return maListeRoles
```

Localiser les rôles FSMO dans une forêt et un domaine :
```
        End Function
```

5.3.3 - Transfert des rôles

Les rôles FSMO sont attribués à des contrôleurs de domaine précis, mais il est possible de transférer ces rôles d'un contrôleur à un autre. Voici un exemple vous montrant comment réaliser une telle opération.

Transférer un rôle FSMO :
```
        Private Function TransfertRoleFSMO(ByVal roleAtransferer As ActiveDirectoryRole, _
                                           ByVal nomDCdestinationRole As String) As Boolean

            Try

                Dim monContext As New DirectoryContext(DirectoryContextType.DirectoryServer, _
                                                       nomDCdestinationRole)

                Dim monDC As DomainController = DomainController.GetDomainController(monContext)
                ' Transfert du rôle vers le contrôleur choisi
                monDC.TransferRoleOwnership(roleAtransferer)

            Catch ex As Exception

                MessageBox.Show(ex.Message)

            End Try

        End Function
```

5.4 - Gestion des sites

5.4.1 - Définition du concept de site dans Active Directory

Dans Active Directory, les sites vous permettent de prendre en compte la topologie de votre réseau. Les sites représentent la structure physique de votre réseau, ils sont donc différents des domaines qui eux représentent la structure logique. La prise en charge de sites est une fonctionnalité indispensable pour toute infrastructure Active Directory ayant une localisation géorgraphique éclatée, dont les différents sites sont liés par des connexions WAN. La gestion de sites dans Active Directory permet d'optimiser la réplication, les opération de maintenance et d'administration, les échanges et les flux de données entre les sites et cela à l'echelle des domaines et de la forêt. Un site est donc une réprésentation physique de votre réseau, qui est indépendante de l'organisation et du découpage des différents domaines.

5.4.2 - Informations sur les sites existants

Il peut être utile dans une vision administrative de récupérer un certain nombre d'informations à propos des sites de votre infrastructure Active Directory, voici comment récupérer par exemple la liste des sites existants dans l'annuaire.

Lister les sites présents dans une fôret Actvive Directory :
```
        Public Function getListeSitesDomaines() As ReadOnlySiteCollection

            Dim listeSites As ReadOnlySiteCollection = Nothing
            ' Connexion à la forêt courante
            Dim monContext As New DirectoryContext(DirectoryContextType.Forest, _
                                                   Forest.GetCurrentForest.ToString)

            Dim maForet As Forest = Forest.GetForest(monContext)

            Try
```

Lister les sites présents dans une fôret Actvive Directory :

```
            ' Récupération de la liste des sites
            listeSites = maForet.Sites

        Catch ex As Exception

            MessageBox.Show(ex.Message)

        End Try

        Return listeSites

    End Function
```

5.4.3 - Créer un site

Voici un exemple commenté vous montrant comment créer un nouveau site.

Créer un site dans l'Active Directory :

```
    Public Function creationSite(ByVal nomSite As String) As Boolean

        Try

            Dim monContext As New DirectoryContext(DirectoryContextType.Forest, _
                                                   Forest.GetCurrentForest.ToString)

            Dim monSite As ActiveDirectorySite
            ' Création du nouveau site dans l'annuaire
            monSite = New ActiveDirectorySite(monContext, nomSite)
            ' Validation des modifications
            monSite.Save()

            Return True

        Catch ex As Exception

            MessageBox.Show(ex.Message)
            Return False

        End Try

    End Function
```

Ressources

Téléchargements

Code source de l'article (Serveur principal)

Code source de l'article (Serveur de secours)

Article au format PDF : 33 pages, 181 Ko (Serveur principal)

Article au format PDF : 33 pages, 181 Ko (Serveur de secours)

Liens vers le MSDN

Namespace System.DirectoryServices

Namespace System.DirectoryServices.ActiveDirectory

Flags des propriétés des comptes d'utilisateurs

La syntaxe pour la propiété Filter de DirectorySearcher

Les classes d'Active Directory

La classe DirectoryEntry

La classe DirectorySearcher

La classe DirectoryContext

Schema d'Active Directory

Liens vers Developpez.com

Mes articles

Mon blog

Les articles des autres rédacteurs

La section .Net de Developpez.com

Conclusion

Tout d'abord, si vous lisez ces quelques mots : félicitations ! En effet, c'est assez long à lire, mais face à un tel sujet, il fallait au moins ca ! J'espère vous avoir montré les possibilités offertes par le Framework .Net 2.0 en matière d'interaction avec Active Directory. En effet, la puissance du Framework en la matière est tout simplement impressionnante, et encore plus depuis la version 2.0 ! Le nouveau namespace System.DirectoryServices.ActiveDirectory permet comme vous avez pu le voir une gestion de haut niveau de votre infrastructure ! Je pense que toutes ces nouveautés vont intéresser au plus haut point les administrateurs réseaux, en effet, développer une application d'administration et de maintenance d'Active Directory est désormais une chose relativement aisée !

Je pense que vous aurez compris que les capacités du Framework à interagir avec .Net m'ont totalement séduit, j'espère qu'il en sera de même pour vous, en effet il y a vraiment de quoi faire ! Merci d'avoir consacré un peu de temps à la lecture de cet article, et je vous donne rendez vous prochainement pour un nouvel article.

Un très grand merci à Freegreg pour la relecture de cet article.