L'imagerie et Active Directory

par MORAND Louis-Guillaume

Date de publication : 18/10/2004

Dernière mise à jour : 27/10/2004

Comment profiter pleinement du schéma d'Active Directory, notamment la récupération et mise en ligne d'images

Avant-propos

Tout le monde sait qu'Active Directory permet de maintenir une base de données complète des utilisateurs du domaine. Certains n'utilisent ce dernier qu'à des fins d'authentification. Mais il est possible d'utiliser l'annuaire Active Directory comme une base de données contenant des informations complètes sur les utilisateurs du domaine (nom, prénom, informations de bureau, etc.). De plus, peu de personnes savent qu'il est possible de stocker une photo pour chaque utilisateur car la console MMC de base ne propose pas d'onglet ou de bouton permettant l'utilisation d'image. Pour l'instant, il est nécessaire de passer par une application tierce (dameware) pour y stocker l'image. Nous allons voir comment utiliser cette option cachée d'Active Directory.

I - Interaction avec Active Directory

Dans ce premier chapitre, nous ne ferons que survoler la connexion avec Active Directory, un article complet existant déjà. Pour des informations complémentaires sur la récupération de données, création des utilisateurs, etc, je vous encourage vivement à lire l'article de Thomas LEBRUN: http://morpheus.developpez.com/ADDotnet

Passons au code.

Pour pouvoir interagir avec Active Directory, il nous faut ajouter le namespace correspondant:

```
using System.DirectoryServices;
```

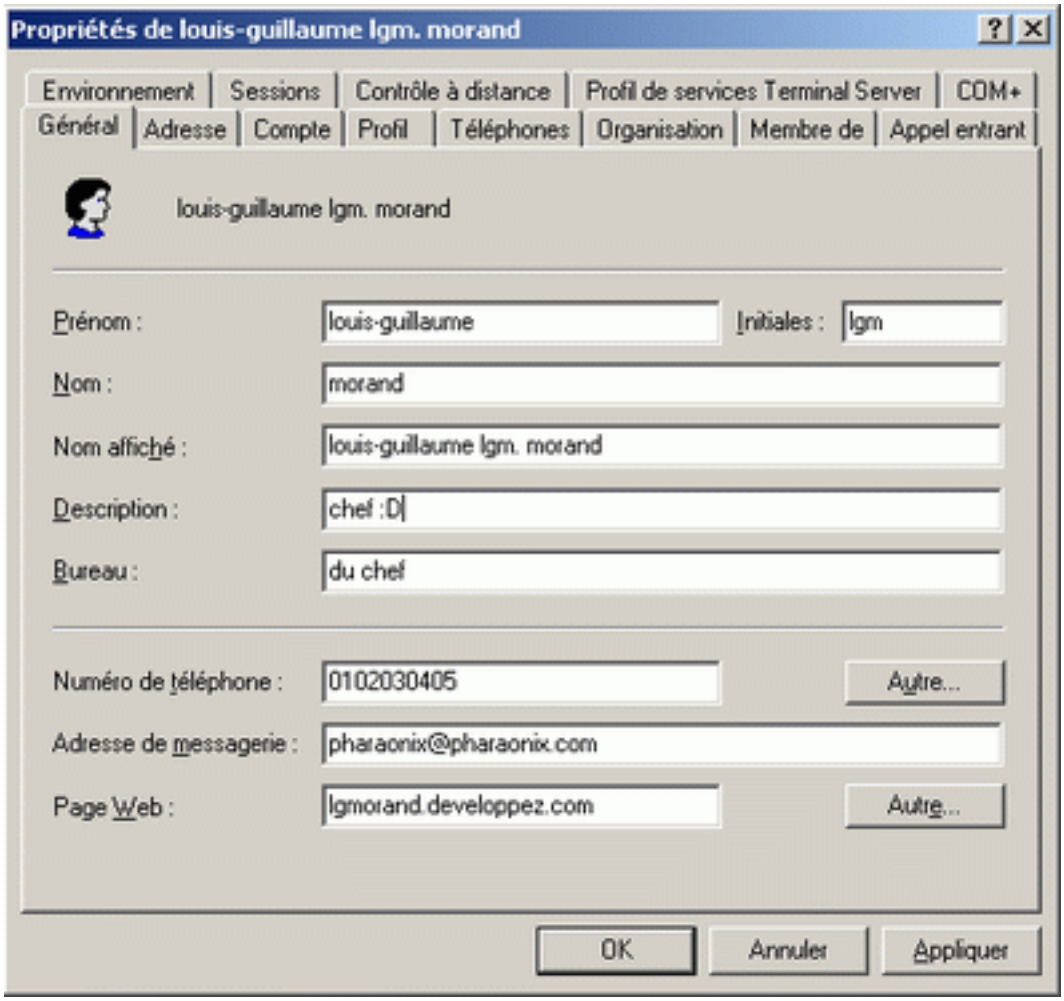
Voilà très rapidement, comment récupérer les informations d'un utilisateur. (Tout ceci n'est qu'un résumé de l'article cité plus haut)

```
try
{
   DirectoryEntry Ldap = new DirectoryEntry ( "LDAP://votre-nom-AD" , "Login" , "Password" );
   DirectorySearcher searcher = new DirectorySearcher (Ldap);
   searcher.Filter = "(objectClass=user)" ;
   foreach ( SearchResult result in searcher.FindAll() )
   {
      // On récupère l'entrée trouvée lors de la recherche
                  DirectoryEntry DirEntry = result.GetDirectoryEntry();

   //On peut maintenant afficher les informations désirées
   string _login = "Login : "  DirEntry.Properties[ "SAMAccountName" ].Value;
   string _login = "Nom : "  DirEntry.Properties[ "sn" ].Value;
   string _login = "Prénom : "  DirEntry.Properties[ "givenName" ].Value;
   string _login = "Email : "  DirEntry.Properties[ "mail" ].Value;
   string _login = "Tél : "  DirEntry.Properties[ "TelephoneNumber" ].Value;
   string _login = "Description : "  DirEntry.Properties[ "description" ].Value;
   }
catch (Exception Ex)
{
      MessageBox.Show(Ex.Message);
}
```

II - Récupération de l'image

Pour ceux qui n'ont jamais administré un domaine ou approché Active Directory, voici des captures d'écran de la console MMC et de la fenêtre de propriété des utilisateurs. Toutes les modifications des informations/propriétés d'un utilisateur se vont via cette fenêtre.



Sachez qu'il n'existe pas d'onglet pour une image/photo dans cette fenêtre. Néanmoins, le schéma d'Active Directory contient une propriété de base qui permet de stocker une image: la propriété *thumbnailPhoto* L'image est alors stockée sous forme binaire; pour la récupérer, nous devons alors récupérer son flux d'octets puis en faire un stream (=flux de données) qui nous permettra de créer l'image à la volée.

```
byte[ ] mesBytes = (byte[ ]) de.Properties[ "thumbnailPhoto" ][ 0 ];        // récupération du flux d'octets
System.IO.MemoryStream monStream= new System.IO.MemoryStream ( mesBytes );  // je crée un stream à partir
                                                                            // de mon tableau d'octets
System.Drawing.Image image = System.Drawing.Image.FromStream ( monStream ); // je crée un objet image
```

Une fois l'objet Image crée, vous pouvez l'afficher directement dans une pictureBox:

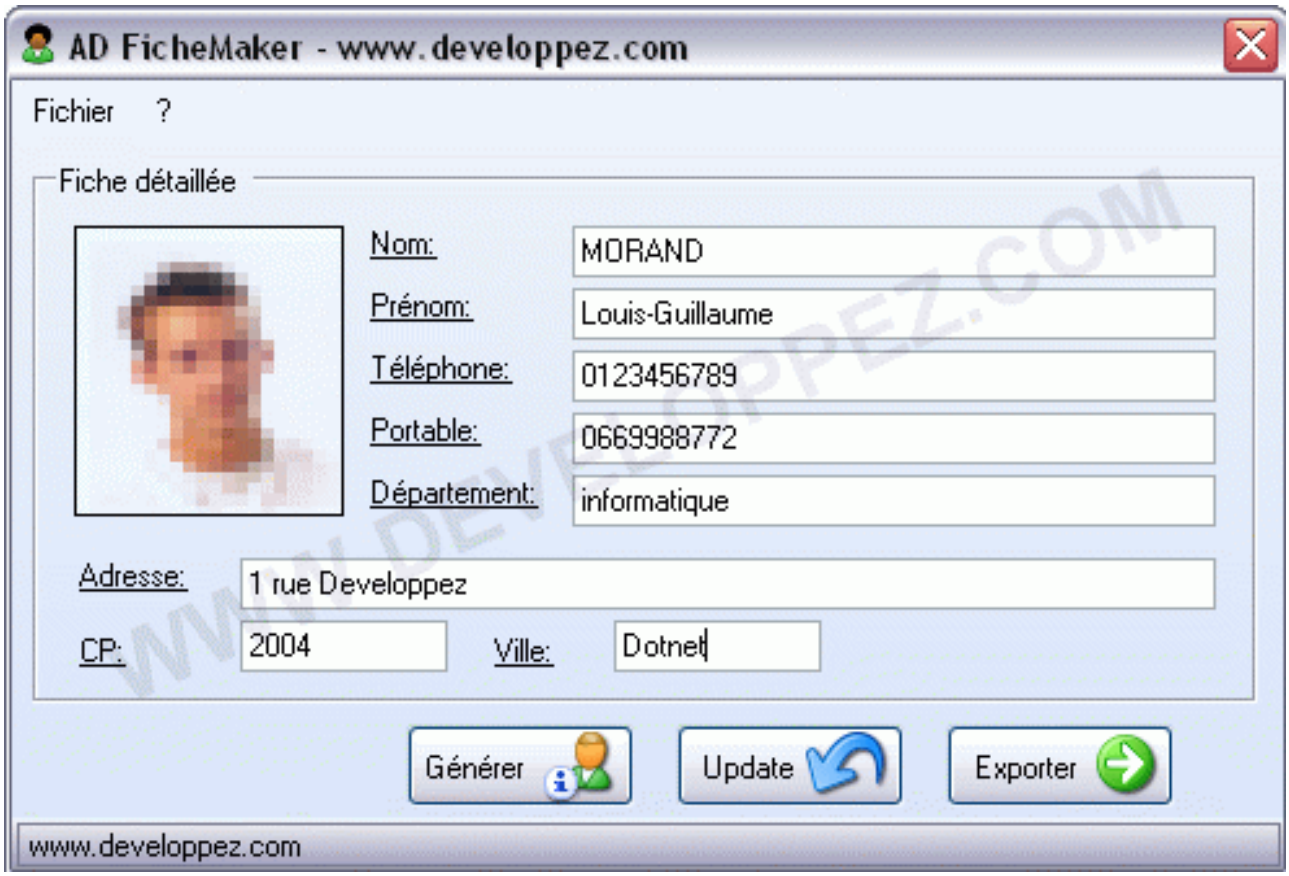
```
maPictureBox.Image = image;
```

Ou la sauvegarder sur le disque dur:

```
image.Save (@"c:\image.jpg");
```

III - Optimisation de l'image

Au moment où il développe son application, le programmeur ne sait pas de quelle taille est l'image stockée et si elle n'engendrera pas des problèmes dans son application (dépassement, déformation). Il convient donc de retravailler l'image avant de l'utiliser dans notre application. Prenons l'exemple d'une application générant une fiche d'identité. Comme l'exemple d'application ci-dessous:



La première "optimisation" consiste à en obtenir une miniature: chose facile à faire en mettant l'option stretch sur une PictureBox. Néanmoins, cette méthode peut avoir un gros effet indésirable: la déformation. En effet, vous risquez d'avoir une photo totalement disproportionnée. Voici un bout de code qui, en fonction de votre PictureBox, récupère une miniature directement d'Active Directory et affiche celle-ci dans notre miniature.

```
   #region Fonction pour afficher l'image au bon format
   private void ResizeImage(string path)
   {
      // réglages des valeurs servant au calcul
      int Lmax = IMG.Width;                                  // IMG est le nom de ma pictureBox
      int Hmax = IMG.Height;

      Image i = Image.FromFile(path);                        // objet image à partir de l'image choisie
      double ratio = (double) Lmax / Hmax;                   // ratio de base à obtenir pour rentrer
   correctement dans la picturebox
      double ratioImage = (double) i.Width / i.Height;       // ratio de l'image d'origine
      double Flng = i.Width;                                 // largeur de l'image d'origine
      double Fht = i.Height;                                 // hauteur de l'image d'origine
      if(Flng > Lmax || Fht > Hmax)                          // si l'image est plus grande d'une
   quelconque longueur
```

```
    {
       if (Flng > Lmax)                                    // si la longueur est plus longue
       {
         if (1 > ratioImage)                                // et si la largueur est plus longue
         {
           Fht = Hmax;                                     // la hauteur prend la hauteur maximale
           if (Flng > i.Height)Flng = Fht / ratioImage;    // calcul de la longueur
           else Flng = Fht * ratioImage;                   // calcul de la longueur (bis)
         }
         else                                              // seule la largeur est plus longue
         {
             Flng = Lmax;                                  // la largeur prend la largeur maximale
             if (Fht > i.Width) Fht = Flng / ratioImage;// calcul de la hauteur
             else Fht = Flng / ratioImage;
         }
       }
       else                                                // seule la largeur est plus longue
       {
         Fht = Hmax;
         Flng = Fht * ratioImage;
       }
       // j'en tire une miniature
       IMG.Image =
Image.FromFile(path).GetThumbnailImage(Convert.ToInt32(Flng),Convert.ToInt32(Fht),null,
IntPtr.Zero);
    }
    else IMG.Image=Image.FromFile(path);                   // sinon j'affiche l'image de base
}
#endregion
```

La deuxième optimisation possible peut se faire si vous souhaitez d'enregistrer votre image en local. Comme vous le savez déjà, pour une vidéo, il est possible d'utiliser un codec, et bien il en est de même pour les images. De plus, il est même possible d'utiliser une compression plus ou moins importante. Commençons tout d'abord par écrire la fonction qui nous permettra de récupérer le codec souhaité. On aurait pu coller celle-ci directement à la suite du code, mais en la mettant à part, cela nous permet d'avoir une vraie fonction réutilisable en lui donnant différents paramètres.

```
#region Retourne le codec souhaité
private ImageCodecInfo GetEncoderInfo(String mimeType)
{
    int j;
    ImageCodecInfo[] encoders;
    encoders = ImageCodecInfo.GetImageEncoders(); // on récupère tous les codecs d'image
    for(j=0;j<encoders.Length; j++ )
    {
       if ( encoders[j].MimeType == mimeType)
       return encoders[j]; // on retourne le codec souhaité
    }
    return null; // aucun codec correspondant n'a été trouvé
}
#endregion
```

Voyons maintenant la partie de code, permettant de sauvegarder une image en gagnant la place sur le disque dur:

```
if ( pictureBox1.Image != null)
{
    //***********************************************
    // Sauvegarde de base sans recompression (sans optimisation)
    //***********************************************/
    // pictureBox1.Image.Save(path"\\photo.jpg");

    //***********************************************
    //Sauvegarde au format jpeg avec compression
    //***********************************************/
    ImageCodecInfo CodecInfo = GetEncoderInfo("image/jpeg"); // je récupère le codec pour enregistrer
```

```
   au format jpeg
      EncoderParameters encoderParameters = new EncoderParameters(1);
      encoderParameters.Param[0] = new EncoderParameter(Encoder.Quality ,80L); // je choisi la qualité
   de compression
      // je sauvegarde en précisant le codec et les parametres de codecs
      pictureBox1.Image.Save(path"\\"  TB_Nom.Text  ".jpg", CodecInfo, encoderParameters);
   }
```

Ici, nous récupérons donc les informations pour encoder au format jpeg (car ce n'est pas qu'une extension!) puis, pour cet encodage, nous lui passons un paramètre de qualité (Encoder.Quality) égal à 80L (80%). Plus ce chiffre est grand (100 maximum) plus la qualité et la taille du fichier image seront grandes.

IV - Stockage de l'image dans Active Directory

Le stockage de l'image dans Active Directory se fait de la même manière d'une propriété standard (nom, prénom, etc). Cette fois-ci, nous devons récupérer un tableau d'octets à partir de notre image.

```
   // mise a jour de l'image
   // je n'ai pas trouvé comment récuperer un tableau d'octets directement depuis une image d'une
   PictureBox
   // donc je la 'save()' en local avant d'en recuperer le stream
   if(pictureBox1.Image != null)
   {
      pictureBox1.Image.Save("monImage");
      System.IO.FileStream monStream = new
   FileStream("monImage",System.IO.FileMode.Open,System.IO.FileAccess.Read);
      byte[] mesBytes = new byte[monStream.Length];       // tableau d'octets contenant l'image à
   uploader
      monStream.Read(mesBytes, 0, (int)monStream.Length); // je met le contenu du stream dans mon
   buffer (mémoire tampon)
      FileInfo fi =new FileInfo("monImage");              // je crée un FileInfo sur mon fichier
      fi.Delete();                                        // je supprime la photo
      de.Properties["thumbnailPhoto"].Value = mesBytes;   // je remplace par ma nouvelle image
   }
```

N'oubliez pas de valider les changements sur le DirectoryEntry, sans quoi, aucune modification ne sera faite.

```
                    de.CommitChanges();
```

Conclusion

Nous savons maintenant qu'il est possible d'utiliser Active Directory comme une base de données ou plus précisément un annuaire complet. Nous savons aussi qu'il est possible d'y stocker et y récupérer des images. Rien de vous empêche dorénavant d'utiliser cet annuaire pour générer des fichiers pour un intranet ou même des fiches pour les ressources humaines d'une société. Je vous fais confiance pour savoir trouver le bon usage.

Pour toute question supplémentaire sur cet article ou Active Directory, vous pouvez contacter pharaonix ou morpheus.

Téléchargements

Pour un exemple concret, voici un petit programme qui permet:

- de se connecter à Active Directory

- lister les utilisateurs

- lister les informations d'un utilisateur choisi (infos image)

- mettre à jour les informations d'un utilisateur (infos image)

- exporter les informations pour la création d'une fiche

Exécutable (371 ko) Miroir

Sources (564 ko) Sources

Un remerciement particulier à morpheus et freegreg pour leurs corrections et améliorations de cet article