

# CONSTRUIRE UNE LISTE DE CONTACTS SUR LA BASE DE L'ACTIVEDIRECTORY

Combien de fois m'a-t-on demandé *Pourquoi il n'y a pas d'annuaire sur l'intranet ?* Grrrrr. Maintenant, il y en a un ! Mais pour ceux qui souhaitent le mettre en place, il y a plusieurs questions organisationnelles à régler :

- Qui le met à jour ? Les petits gars de l'informatique ont autre chose à faire que de renseigner les numéros de portables ! Il faut trouver une personne qui ne risque pas de vous mettre en l'air l'ensemble de votre AD !!
- Combien de contacts et de requêtes par jour ? Le cas d'espèce concerne une petite centaine de noms et peu de requêtes quotidiennes. Pour des annuaires plus conséquents, il sera nécessaire de trouver une autre solution (stockage en base avec MAJ quotidienne par exemple ou optimisation du DataSet)

Ce tutoriel passe en revue plusieurs techniques :

- L'accès à l'Active Directory,
- La construction d'un DataSet,
- La publication dans un DataGridView paginé,
- La création d'un UserControl de publication et de recherche de contacts,
- L'invocation d'un XML Webservice

N'hésitez pas à télécharger et à utiliser les sources.

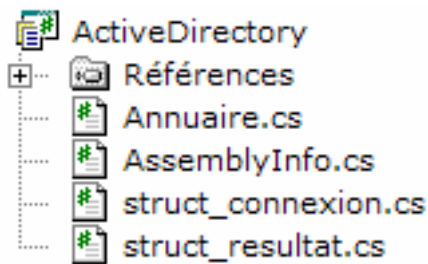
## 1. La Classe d'accès à l'ActiveDirectory

Comme d'habitude, mon projet dispose d'une classe pour les accès et de 2 structures.

struct\_connexion contient l'objet DirectoryEntry qui me permet d'accéder à mon ActiveDirectory

struct\_resultat est quand à lui intégré dans struct\_connexion.

Utilisez l'assembly System.DirectoryServices pour exploiter votre Active Directory



## 1.1 La connexion à l'Active Directory

```
public static struct_connexion Connexion(string p_serveur, string p_user, string p_pwd)
{
    // Déclare mes objets et variables
    DirectoryEntry ent;
    struct_connexion v_result = new struct_connexion();
    v_result.Retour = new struct_resultat();
    v_result.Retour.Retour = false;
    v_result.Retour.Infos = "";
    try
    {
        // Connexion au serveur LDAP
        ent = new DirectoryEntry(p_serveur,p_user,p_pwd);
        v_result.Retour.Retour = true;
        v_result.Entry = ent;
    }
    // Gestion des erreurs
    catch (System.Runtime.InteropServices.COMException)
    {
        System.Runtime.InteropServices.COMException exception = new System.Runtime.InteropServices.COMException();
        v_result.Retour.Infos = exception.ToString();
    }
    catch (InvalidOperationException)
    {
        InvalidOperationException InvOpEx = new InvalidOperationException();
        v_result.Retour.Infos = InvOpEx.Message;
    }
    catch (Exception e)
    {
        v_result.Retour.Infos = e.Message;
    }
    return v_result;
}
```

Ma méthode Connexion prend 3 arguments, l'adresse du serveur ActiveDirectory, l'utilisateur habilité à y accéder et son mot de passe.

L'objet de connexion DirectoryEntry est stocké dans ma structure ainsi que l'état de la connexion (succès ou échec). Une vieille habitude d'ASP mais les erreurs de connexion peuvent aussi être renvoyées au niveau supérieur par un vulgaire throw.

## 1.2 La construction de la table pour le DataSet

```

/// <summary>
/// Génère la structure du DataTable
/// </summary>
/// <returns></returns>
private static DataTable ConstruitTable()
{
    DataTable v_result = new DataTable();
    DataColumn myColumn;

    myColumn = new DataColumn();
    myColumn.DataType = System.Type.GetType("System.String");
    myColumn.ColumnName = "sn";
    v_result.Columns.Add(myColumn);
}

```

Rien de très compliqué ici. On crée de toute pièce un DataTable qui correspondra au niveau du nom des champs aux clés LDAP. Vous allez ici faire le tri des données que vous souhaitez récupérer. Voici une liste des principales clés disponibles dans l'ActiveDirectory.

<http://www.aspfr.com/forum.v2.aspx?ID=84360> pour une liste (trop) complète.

birthLocation	ipPhone	otherTelephone
businessCategory	location	ou
cn	mail	pager
co	mailAddress	personalTitle
company	middleName	physicalDeliveryOfficeName
countryCode	mobile	postalAddress
dc	name	postalCode
department	notes	postOfficeBox
description	o	sn
displayName	otherFacsimileTelephoneNumber	st
displayNamePrintable	otherHomePhone	street
division	otherIpPhone	streetAddress
employeeID	otherLoginWorkstations	telephoneNumber
employeeNumber	otherMailbox	teletexTerminalIdentifier
employeeType	otherMobile	telexNumber
givenName	otherPager	title
initials		

### 1.3 Accéder aux clés de l'ActiveDirectory

```
/// <summary>
/// Retourne la valeur de la propriété AD
/// </summary>
/// <param name="searchResult">Objet SearchResult</param>
/// <param name="PropertyName">Chaîne à rechercher</param>
/// <returns></returns>
private static string RetournePropriete(SearchResult p_searchResult, string p_propertyName)
{
    if (p_searchResult.Properties.Contains(p_propertyName))
        return p_searchResult.Properties[p_propertyName][0].ToString();
    else
        return string.Empty;
}
```

Une fois connecté à mon annuaire et les données récupérées, ma méthode RetournePropriete me donne la valeur de la string. Elle attend l'objet SearchResult (qui contient un enregistrement de l'annuaire) et la clé.

## 1.4 Insérer les données de l'ActiveDirectory dans mon DataSet

```
/// <summary>
/// Lance la recherche et retourne un DataTable
/// </summary>
/// <param name="p_directoryEntry">Objet DirectoryEntry</param>
/// <param name="p_all">Vrai si tous les utilisateurs, Faux si demande de recherche</param>
/// <param name="p_arg">Nom d'utilisateur à rechercher si p_all est false</param>
/// <returns></returns>
public static DataSet Recherche(DirectoryEntry p_directoryEntry, bool p_all, string p_arg)
{
    DataSet v_result = new DataSet();
    try
    {
        DirectorySearcher v_DirectorySearcher = new DirectorySearcher(p_directoryEntry);
        SearchResultCollection result;
        // Gestion du filtre dans la recherche ldap, tous ou seulement une partie des utilisateurs
        if (!p_all)
            v_DirectorySearcher.Filter = "&(objectClass=user)(anr=" + p_arg + "*)";
        else
            v_DirectorySearcher.Filter = "&(objectClass=user)";

        result = v_DirectorySearcher.FindAll();

        if (result.Count > 0)
        {
            DataTable da;
            DataRow myRow;

            // Génère la table nécessaire pour la structure LDAP
            da = ConstruitTable();
            int i = 0;
            bool v_isName = false;

```

Ma méthode Recherche va retourner le DataSet avec l'ensemble des données de mon annuaire. Elle utilise l'Objet DirectoryEntry, qui a été instancié lors de la connexion à mon annuaire, une valeur true ou false (true si je souhaite toutes les données), et une chaîne de recherche (si p\_all est à false).

Si j'ai des résultats, je construis mon DataTable

```

// Passe en revue les résultat et ne retient que les utilisateurs ayant un nom et
// un numéro de téléphone ou un email ou un portail
foreach(SearchResult resEnt1 in result)
{
    if (RetournePropriete(resEnt1,"displayname").Length > 0)
    {
        if ((RetournePropriete(resEnt1,"telephonenumber").Length > 0) ||
            (RetournePropriete(resEnt1,"mail").Length > 0) ||
            (RetournePropriete(resEnt1,"mobile").Length > 0))
        {
            v_isName = true;
        }

        // C'esy un utilisateur a récupérer
        if (v_isName)
        {
            // Ajout dans la table
            myRow = da.NewRow();
            myRow["sn"] = RetournePropriete(resEnt1,"sn");
            myRow["name"] = RetournePropriete(resEnt1,"name");
            myRow["givenname"] = RetournePropriete(resEnt1,"givenname");
            myRow["displayname"] = RetournePropriete(resEnt1,"displayname");
            myRow["title"] = RetournePropriete(resEnt1,"title");
            myRow["othertelephone"] = RetournePropriete(resEnt1,"othertelephone");
            myRow["telephonenumber"] = RetournePropriete(resEnt1,"telephonenumber");
            myRow["mobile"] = RetournePropriete(resEnt1,"mobile");
            myRow["company"] = RetournePropriete(resEnt1,"company");
            myRow["mail"] = RetournePropriete(resEnt1,"mail");
            myRow["departement"] = RetournePropriete(resEnt1,"departement");
            da.Rows.Add(myRow);
        }
        v_isName = false;
    }
}

```

Il est nécessaire ici d'opérer un tri au risque de remonter l'utilisateur ASPNET ! En effet, je ne remonte que les utilisateurs qui ont un displayName et des coordonnées (telephone, mail ou mobile).

Dès lors, je crée un DataRow et j'ajoute les valeurs dans les champs grâce à la méthode RetournePropriete décrite plus haut.

```

}
// Ajoute la table au DataSet
v_result.Tables.Add(da);

```

Ne pas oublier l'ajout de la table dans le DataSet

## 2. La construction du DataGrid et de la pagination

Pour des questions évidentes de modularité, j'ai créé un UserControl qui sera placé dans une page ASPX. Ce UserControl contient la grille et tous les modules d'accès et de recherche dans l'ActiveDirectory

### 2.1 Le UserControl et le DataGrid

```
<table cellSpacing="1" cellPadding="0" width="650" border="0">
  <tr>
    <td vAlign="middle" align="right">rechercher :&nbsp;
      <asp:textbox id="txt_recherche" MaxLength="30" runat="server"></asp:textbox>
      <asp:button id="bt_recherche" runat="server" Text="Cherche">
      </asp:button>&nbsp;<asp:button id="bt_all" runat="server" Text="Afficher Tout"></asp:button>
    </td>
  </tr>
</table>
<br>
<asp:label id="lb_erreur" runat="server"></asp:label>
<asp:datagrid id="dg" runat="server" OnSortCommand="Grid_SortCommand" OnPageIndexChanged="Grid_Change"
  CellPadding="4" BackColor="White" BorderWidth="1px" BorderStyle="None" BorderColor="#CC9966" AllowPaging="True"
  PageSize="25" AutoGenerateColumns="False">
  <SelectedItemStyle Font-Bold="True" ForeColor="#663399" BackColor="#FFCC66"></SelectedItemStyle>
  <ItemStyle ForeColor="#330099" BackColor="White"></ItemStyle>
  <HeaderStyle Font-Bold="True" ForeColor="#FFFFCC" BackColor="#990000"></HeaderStyle>
  <FooterStyle ForeColor="#330099" BackColor="#FFFFCC"></FooterStyle>
  <Columns>
    <asp:BoundColumn DataField="displayname" HeaderText="Nom" SortExpression="sn">
      <ItemStyle HorizontalAlign="Left" Width="150px"></ItemStyle>
    </asp:BoundColumn>
    <asp:BoundColumn DataField="telephonenumber" HeaderText="telephone">
      <ItemStyle HorizontalAlign="Center" Width="100px"></ItemStyle>
    </asp:BoundColumn>
    <asp:BoundColumn DataField="othertelephone" HeaderText="direct">
      <ItemStyle HorizontalAlign="Center" Width="40px"></ItemStyle>
    </asp:BoundColumn>
    <asp:BoundColumn DataField="mobile" HeaderText="portable" SortExpression="mobile">
      <ItemStyle HorizontalAlign="Center" Width="100px"></ItemStyle>
    </asp:BoundColumn>
    <asp:BoundColumn DataField="mail" HeaderText="email" SortExpression="mail">
      <ItemStyle HorizontalAlign="Left" Width="220px"></ItemStyle>
    </asp:BoundColumn>
    <asp:BoundColumn DataField="company" HeaderText="Site" SortExpression="company">
      <ItemStyle HorizontalAlign="Center" Width="40px"></ItemStyle>
    </asp:BoundColumn>
  </Columns>
  <PagerStyle HorizontalAlign="Center" ForeColor="#330099" Position="TopAndBottom" BackColor="#FFFFCC"
  Mode="NumericPages"></PagerStyle>
</asp:datagrid>
```

La partie supérieure servira pour lancer des actions. txt\_recherche permet la saisie de la chaîne de caractères de recherche, bt\_cherche, lance la sélection, bt\_all permet de revenir à une liste non filtrée. lb\_erreur affiche les éventuels messages d'échec.

Mon DataGrid, dg dispose d'une méthode pour le classement « Grid\_SortCommand » et pour la pagination « Grid\_Change ». Notez également les Expressions de tri sur les Colonnes qui correspondent aux noms des champs du DataSet.

## 2.2 Le chargement du UserControl

```
private void Page_Load(object sender, System.EventArgs e)
{
    bt_recherche.Text = "Cherche";
    lb_erreur.Text = "";
    txt_recherche.Text = "";

    // Gère la navigation dans mon DataGrid
    dg.AllowPaging = true;
    dg.AllowSorting = true;
    dg.PagerStyle.Mode = PagerMode.NumericPages;
    dg.PagerStyle.PageButtonCount = 10;
    if( !IsPostBack )
    {
        try
        {
            // Je donne des attributs a ma grille afin de gérer les tris et recherches
            dg.Attributes["sort"] = "ASC";
            dg.Attributes["nom"] = "sn";
            dg.Attributes["bFind"] = "true";
            dg.Attributes["recherche"] = "";
            BindGrid( dg.Attributes["nom"], dg.Attributes["sort"],true,null);
        }
        catch(Exception ex)
        {
            lb_erreur.Text += ex.Message.ToString();
        }
    }

    // Gestion de l'affichage du bouton afin de revenir à l'affichage complet du ldap
    if (dg.Attributes["bFind"] == "true")
        bt_all.Visible = false;
    else
        bt_all.Visible = true;
}
```

J'ajoute 4 attributs à mon DataGrid ce qui me permet de stocker différents états :

- Le mode de tri, ascendant ou descendant,
- Le champ de tri, par défaut «sn »,
- Le type d'affichage (tout ou le résultat d'une recherche –true ou false-),
- La chaîne de filtre

J'invoque alors BindGrid avec les paramètres d'affichage et de contenu).

bt\_all n'est visible que si j'ai appliqué un filtre. Ce bouton permet de revenir à l'affichage complet de l'annuaire.

## 2.3 Le Bind du DataGrid

```
private void BindGrid( string p_colonne, string p_sort, bool p_all, string p_recherche)
{
    // structure qui gère le retour et le DataSet
    DataSet v_result = new DataSet();
    v_result = Affichage(p_all,p_recherche);

    if (v_result.Tables.Count > 0)
    {
        if (v_result.Tables[0].Rows.Count > 0)
        {
            // Alimentation du DataGrid
            DataView v_dv = new DataView(v_result.Tables[0]);
            v_dv.Sort = p_colonne + " " + p_sort;

            int nbTotalEnregistrement = v_dv.Count;
            // Gère le pb de paging avec suppressions de ligne, on doit redéfinir la variable CurrentPageIndex
            if( dg.PageSize * ( dg.PageCount - 1 ) >= nbTotalEnregistrement && ( dg.PageCount - 1 ) == dg.Cu
            {
                dg.CurrentPageIndex -= 1;
            }
            else if(nbTotalEnregistrement == 0)
            {
                dg.CurrentPageIndex = 0;
            }
            dg.DataSource = v_dv;
            dg.DataBind();

        }
        else
        {
            lb_erreur.Text = "Aucun résultat";
        }
    }
    else
    {
        lb_erreur.Text = "Aucun résultat";
    }
}
```

Ma méthode Affichage me construit le DataSet. Elle est détaillée dans la section 3.2.

Si mon DataSet contient au moins un table et un enregistrement, je place la DataTable dans mon DataView que je tri en fonction des paramètres de la méthode.

On gère ensuite la page en cours et je bind mon DataView dans mon DataGrid.

## 2.4 La gestion de la navigation dans le DataGrid

```
// Gestion des demandes de navigation
public void Grid_Change(Object sender, DataGridPageChangedEventArgs e)
{
    dg.CurrentPageIndex = e.NewPageIndex;
    dg.Attributes["sort"] = "ASC";
    dg.Attributes["nom"] = "sn";
    BindGrid( dg.Attributes["nom"],
              dg.Attributes["sort"],
              Convert.ToBoolean(dg.Attributes["bFind"].ToString()),
              dg.Attributes["recherche"]);
}

// Gestion du tri ascendant et descendant
protected void Grid_SortCommand(object sender, DataGridSortCommandEventArgs e)
{
    if( dg.Attributes["sort"] == "ASC" )
        dg.Attributes["sort"] = "DESC";
    else
        dg.Attributes["sort"] = "ASC";

    // Bind la grille
    BindGrid(e.SortExpression.ToString(),
            dg.Attributes["sort"],
            Convert.ToBoolean(dg.Attributes["bFind"].ToString()),
            dg.Attributes["recherche"]);
}
```

On retrouve ici les méthodes `Grid_Change` pour le changement de page qui affecte la nouvelle page en cours et reprends des valeurs des attributs du DataGrid pour construire correctement le DataSet et le tri, et `Grid_SortCommand` qui gère le tri ascendant et descendant.

Je bind mon DataGrid à chaque fois.

## 2.5 La recherche

```
// Gestion de la demande d'affichage de tous les noms
private void bt_all_Click(object sender, System.EventArgs e)
{
    dg.Attributes["sort"] = "DESC";
    dg.Attributes["nom"] = "sn";
    dg.Attributes["bFind"] = "true";
    dg.Attributes["recherche"] = "";
    dg.CurrentPageIndex = 0;
    BindGrid(dg.Attributes["nom"], dg.Attributes["sort"], true,null);
    bt_all.Visible = false;
}

// Demande de recherche
private void bt_recherche_Click(object sender, System.EventArgs e)
{
    // Initialise boutons et DataGrid
    bt_all.Visible = true;
    dg.Attributes["sort"] = "DESC";
    dg.Attributes["nom"] = "sn";
    dg.Attributes["bFind"] = "false";
    dg.Attributes["recherche"] = txt_recherche.Text.ToString();
    dg.CurrentPageIndex = 0;

    // Bind le DataGrid
    BindGrid(dg.Attributes["nom"], dg.Attributes["sort"], false, txt_recherche.Text.ToString());
}
}
```

bt\_all\_Click permet de revenir à l'affichage global, sans filtre.

bt\_rechercher\_Client lance la recherche en fonction du mot clé saisi.

### 3. L'invocation du Webservice

J'ai choisi dans cet exemple d'accéder à l'ActiveDirectory par un XML pour 2 raisons :

- C'est un bon cas pratique, l'occasion de montrer l'utilisation des WebServices
- Dans le cas d'espèce, je souhaitais ajouter la fonctionnalité d'annuaire dans une application SharePoint. Les règles de sécurité m'empêchaient tout simplement d'accéder à l'annuaire. D'où l'idée de passer par une couche tiers, le Webservice !

### 3.1 Le XML WebService

```

/// <summary>
/// Retourne les données issues de l'ActiveDirectory
/// </summary>
/// <param name="p_serveur">Nom du serveur LDAP</param>
/// <param name="p_user">Utilisateur autorisé</param>
/// <param name="p_password">Mot de passe</param>
/// <param name="p_all">True pour toutes les données, false pour faire jouer le filtre</param>
/// <param name="p_recherche">chaîne de recherche dans l'annuaire</param>
/// <returns></returns>
[WebMethod(Description="Retourne les données de l'ActiveDirectory")]
public DataSet MonAnnuaire(string p_serveur, string p_user, string p_password, bool p_all, string p_recherche)
{
    DataSet v_result = new DataSet();

    // Connexion au serveur ldap
    ActiveDirectory.struct_connexion v_ret = ActiveDirectory.Annuaire.Connexion(p_serveur, p_user, p_password);
    // Vérifie si la connexion est correcte
    if (v_ret.Retour.Retour)
    {
        // Alimentation du DataSet
        v_result = ActiveDirectory.Annuaire.Recherche(v_ret.Entry, p_all, p_recherche);
    }
    return v_result;
}

```

Mon WebService attend les paramètres de connexion et de recherche. En cas d'erreur, le WebService retournera un DataSet vide.

N'oubliez pas la description du WebService, cela ne coûte rien.

### 3.2 L'accès au XML WebService

```

/// <summary>
/// Gere l'affichage du résultat de la requete LDAP
/// </summary>
/// <param name="p_all">true, retourne tous les éléments, false filtre selon la valeur de p_name</param>
/// <param name="p_name">Valeur du filtre</param>
private static DataSet Affichage(bool p_all, string p_name)
{
    // J'ai un WebService référencé
    wsActiveDirectory.Annuaire ws = new wsActiveDirectory.Annuaire();

    DataSet v_result = ws.MonAnnuaire("LDAP://{MONSITE}",
                                     @"{DOMAINE}\{UTILISATEUR}",
                                     "{MOTDEPASSE}",
                                     p_all, p_name);

    return v_result;
}

```

Après avoir ajouter une WebReference vers le WebService qui retourne les données de l'annuaire, on invoque la WebMethod MonAnnuaire avec les paramètres de connexion et de recherche... Tout simplement. Le WebService retourne un DataSet.

#### 4. Mon utilisateur dans l'ActiveDirectory et mon application Web

Dans l'ActiveDirectory, j'ai renseigné mon compte utilisateur de la manière suivante :

The screenshot shows the 'Propriétés de Laurent Geffroy' dialog box with the 'Compte' tab selected. The fields are as follows:

- Prénom : Laurent
- Initiales : (empty)
- Nom : Geffroy
- Nom affiché : Laurent Geffroy
- Description : (empty)
- Bureau : (empty)
- Numéro de téléphone : 01 44
- Adresse de messagerie : lgeffroy
- Page Web : (empty)

Buttons: 'Autre...' (next to phone number and web page), 'OK', 'Annuler', 'Appliquer'.

J'ajoute ma ligne directe en cliquant sur Autre

The screenshot shows the 'Numéro de téléphone (Autres)' dialog box. The 'Nouvelle valeur' field is empty. The 'Valeurs actuelles' list contains '306'. Buttons include 'Ajouter', 'Modifier', 'Supprimer', 'OK', and 'Annuler'.

Je précise mes autres numéros dans l'onglet « Téléphones »

**Propriétés de Laurent Geffroy** [?] [X]

Membre de | Appel entrant | Objet | Sécurité | Environnement | Sessions  
 Contrôle à distance | Profil de services Terminal Server | COM+  
 Général | Adresse | Compte | Profil | **Téléphones** | Organisation | Certificats publiés

Numéros de téléphone

Domicile :  [Autres...]

Radiomessagerie :  [Autres...]

Tél. mobile :  [Autres...]

Télécopie :  [Autres...]

Téléphone IP :  [Autres...]

Remarques :

[OK] [Annuler] [Appliquer]

Enfin, j'ajoute mon titre, mon service et mon site dans le champ Société.

**Propriétés de Laurent Geffroy** [?] [X]

Membre de | Appel entrant | Objet | Sécurité | Environnement | Sessions  
 Contrôle à distance | Profil de services Terminal Server | COM+  
 Général | Adresse | Compte | Profil | Téléphones | **Organisation** | Certificats publiés

Titre :

Service :

Société :

Gestionnaire

Nom :

[Modifier...] [Propriétés] [Effacer]

Collaborateurs :

[OK] [Annuler] [Appliquer]

Voici le résultat dans mon application Web ! Le tour est joué.

rechercher :

1					
Nom	telephone	direct	portable	email	Site
Laurent Geffroy	01 44	306	06 71	lgeffroy	R
1					

## 5. Pour conclure

Malgré la richesse des points abordés, le développement peut être amélioré sur plusieurs points :

- Sur l'optimisation du DataSet, notamment si les données récupérées sont importantes

- Sur le mode de récupération. Un tableau des propriétés LDAP pourrait être conçu pour construire rapidement le DataTable par une boucle.

J'espère que ce tutoriel vous a aidé à maîtriser l'accès à votre annuaire LDAP.

## 6. En Savoir plus

Objets ActiveDirectory

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon/html/vbconIntroductionToADSIObjectsInVisualStudio.asp>

Rechercher les informations Utilisateurs dans AD (VB.NET)

<http://www.123aspx.com/redir.aspx?res=30841>

Accéder à l'ActiveDirectory

<http://www.15seconds.com/issue/020730.htm>

Optimiser son DataSet

<http://www.eggheadcafe.com/articles/20031219.asp>

L'auteur : Laurent GEFROY

<http://www.laurentgeffroy.com>

lgeffroy-at-club-internet.fr