

Compte rendu de stage

Annexes

10/02/2015

Rédiger par : Romain Ledru stagiaire au CNAM du 05/01/2015 au 13/02/2015

Contenu

1. Attributs Active Directory	2
1.1. Attributs de la classe utilisateurs	2
1.2. Ajouter un attribut dans le schéma Active Directory.	4
2. PowerShell	7
2.1. Ajout du module Active Directory	7
2.1.1. Etape 1 :	7
2.1.2. Etape 2 :	7
2.1.3. Etape 3 :	8
2.2. Script PowerShell	9
2.2.1. Afficher les utilisateurs :	9
2.2.2. Modifier le mot de passe d'un utilisateur :	9
2.2.3. Cas particulier :	11
3. Visual Studio 2013 community	14
3.1. Script C# mode console	14
3.1.2 Script C# IHM, asp.net	26

1. Attributs Active Directory

1.1. Attributs de la classe utilisateurs

S'applique à l'objet	Emplacement ADU&C	Attribut	Description
Utilisateur, Contact	Onglet Général	physicalDeliveryOfficeName	Bureau
Utilisateur, Contact	Onglet Général	displayName	Nom affiché
Utilisateur, Contact	Onglet Général	givenName	Prénom
Utilisateur, Contact	Onglet Général	initials	Initiale
Utilisateur, Contact	Onglet Général	sn	Nom de famille
Utilisateur, Contact	Onglet Général	telephoneNumber	Numéro professionnel
Utilisateur, Contact	Onglet Général	otherTelephone	Numéro professionnel 2
Utilisateur, Contact	Onglet Téléphones	homePhone	Numéro personnel
Utilisateur, Contact	Onglet Téléphones	otherHomePhone	Numéro personnel 2
Utilisateur, Contact	Onglet Téléphones	pager	Numéro personnel radiomessagerie
Utilisateur, Contact	Onglet Téléphones	mobile	Numéro personnel portable
Utilisateur, Contact	Onglet Téléphones	facsimileTelephoneNumber	Numéro de fax
Utilisateur, Contact	Onglet Téléphones	info	Champ Notes
Utilisateur, Contact	Onglet Organisation	manager	Directeur
Utilisateur, Contact	Onglet Organisation	directReports	Collaborateurs
Utilisateur, Contact	Onglet Organisation	title	Titre
Utilisateur, Contact	Onglet Organisation	company	Nom de l'entreprise
Utilisateur, Contact	Onglet Organisation	department	Nom du département

Contact			
Utilisateur, Contact	Onglet Adresse	streetAddress	Adresse
Utilisateur, Contact	Onglet Adresse	postOfficeBox	Boîte postale
Utilisateur, Contact	Onglet Adresse	l	Ville
Utilisateur, Contact	Onglet Adresse	st	État
Utilisateur, Contact	Onglet Adresse	postalCode	Code postal
Utilisateur, Contact	Onglet Adresse	countryCode	Pays
Utilisateur, Contact	Non applicable	telephoneAssistant	Téléphone de l'assistant
Utilisateur, Contact	Non applicable	msExchangeAssistantName	Nom de l'assistant
Groupe	Onglet Géré par	managedBy	Propriétaire de groupe
Groupe	Onglet Général	Info	Champ Notes

Source : <https://technet.microsoft.com/fr-fr/library/aa997520%28v=exchg.65%29.aspx>

Remarque :

Les attributs telephoneAssistant et secretary doivent être entrés manuellement à l'aide de l'outil LDP (Ldp.exe) ou ADSI Edit (AdsiEdit.msc). Le composant logiciel enfichable MMC Utilisateurs et ordinateurs Active Directory n'expose pas ces champs.

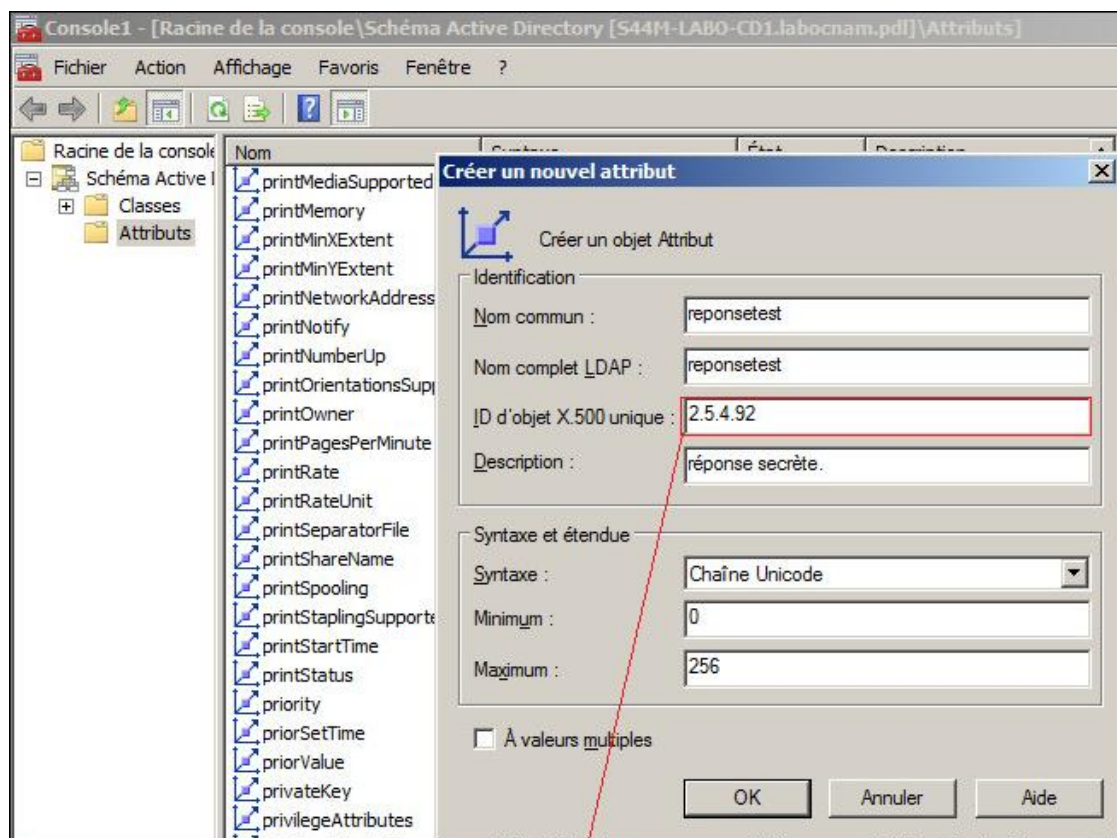
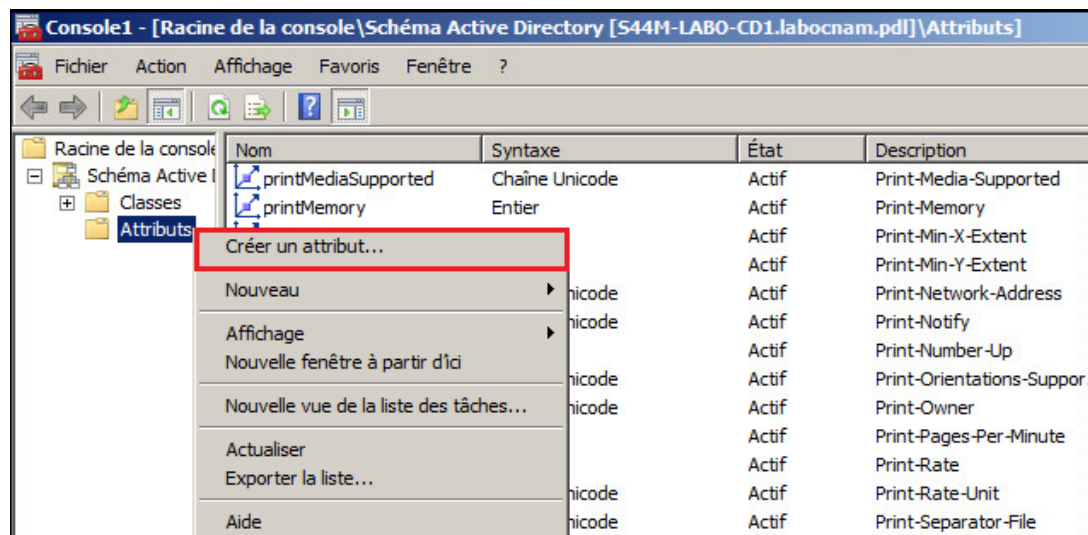
1.2. Ajouter un attribut dans le schéma Active Directory.

Mode opératoire :

Sur le poste qui virtualise l'AD :

Taper: windows + r: « mmc » enter.

Faire un clique droit sur « **attributs** » et cliquer sur « **créer un attribut** ».



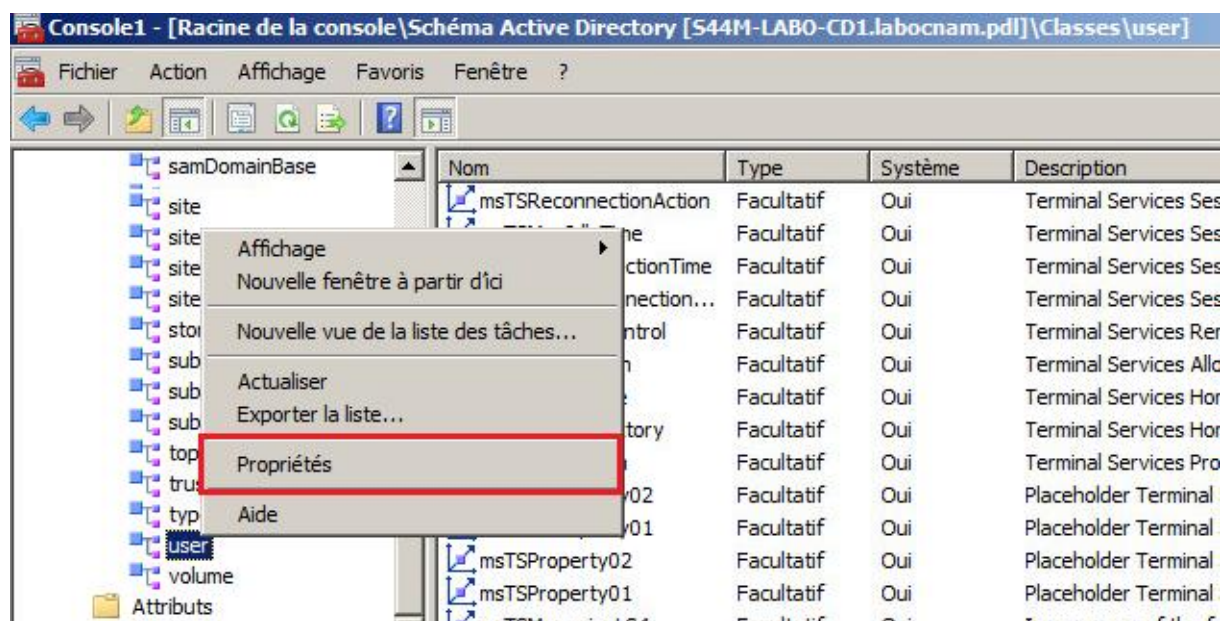
Remplir le formulaire de création d'attribut, puis cliquer sur « **ok** ».

L'attribut est maintenant créé, il faut l'ajouter à la classe désirée.

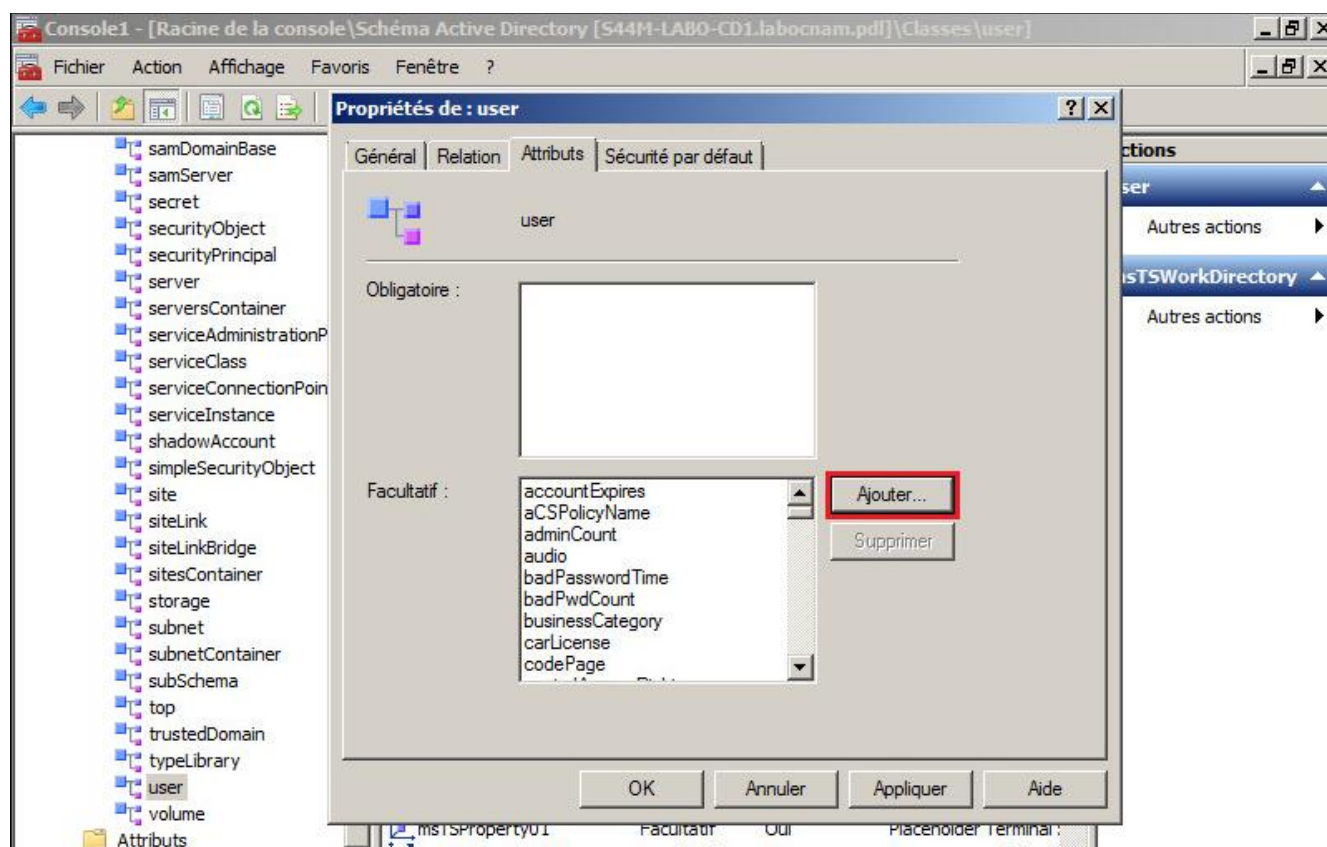
- 2.5 - fait référence au service X500
- 2.5.4 - est la définition des types d'attributs
- .92 - pour être sûr qu'il ne soit pas déjà pris

Chercher la classe « **user** » dans « **Classes** » (arborescence de gauche).

Puis clic droit sur la classe « **user** » pour accéder aux « **propriétés** ».

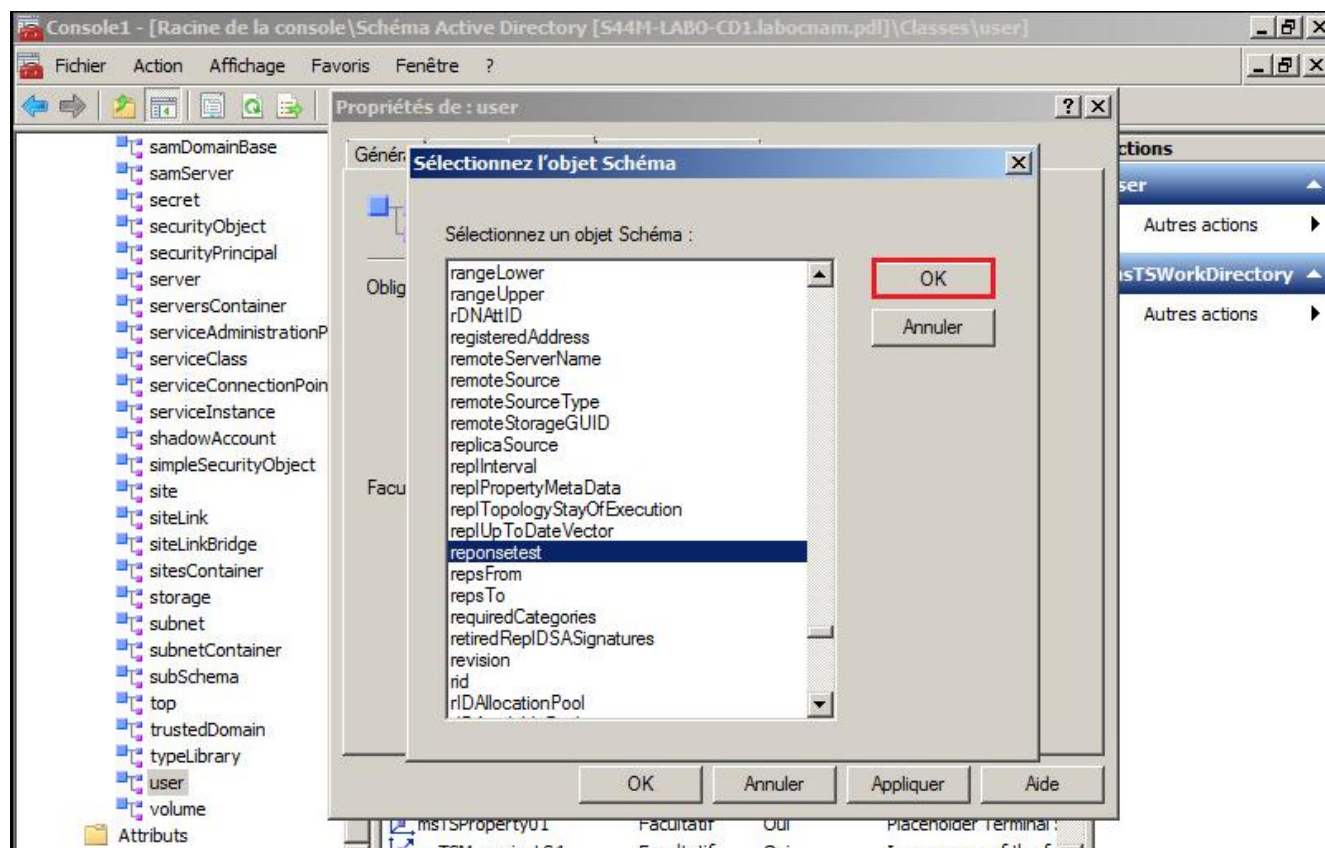


Cliquer sur l'onglet « **Attributs** » puis sur le bouton « **Ajouter...** ».



Rechercher l'attribut désiré (« reponsetest ») dans la liste et cliquer sur le bouton « ok » puis

« Appliquer » pour valider.



2. PowerShell

2.1. Ajout du module Active Directory

2.1.1. Etape 1 :

Installer PowerShell sur le poste (V3. ou supérieur recommander).

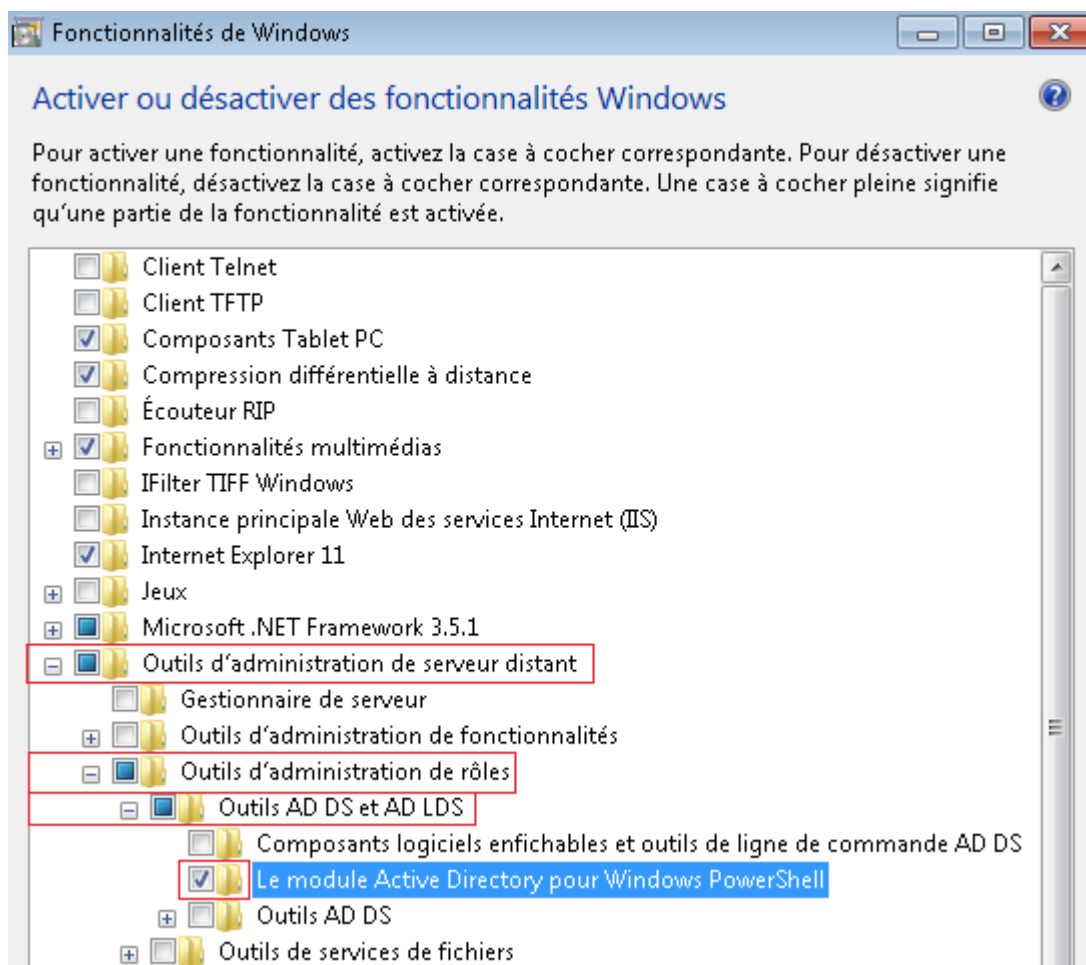
Télécharger : Outils d'administration de serveur distant pour Windows 7 :

<http://www.microsoft.com/fr-FR/download/details.aspx?id=7887>

2.1.2. Etape 2 :

Après l'installation de « **KB958830** » :

Accéder au panneau « fonctionnalités Windows » (Démarrer -> Panneau de configuration -> Programmes -> Activer ou désactiver des fonctionnalités Windows)

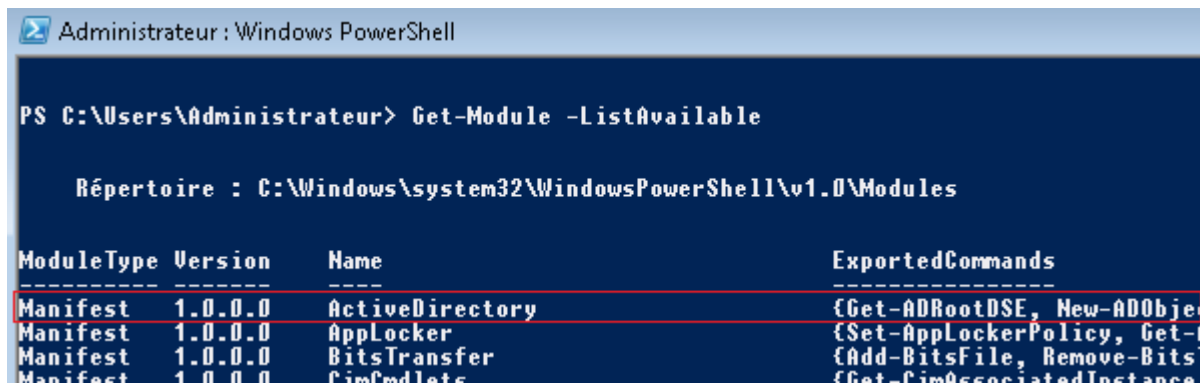


Et cocher la case « Le module Active Directory pour Windows PowerShell ».

2.1.3. Etape 3 :

Vérifier que le module Active Directory et ses commandes sont bien détectées :

Get-Module -ListAvailable



```

PS C:\Users\Administrateur> Get-Module -ListAvailable

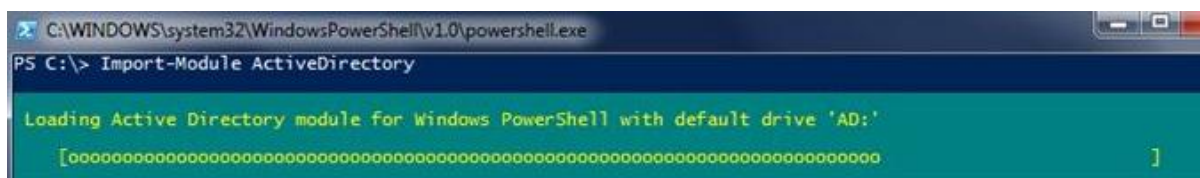
Répertoire : C:\Windows\system32\WindowsPowerShell\v1.0\Modules

ModuleType Version      Name                               ExportedCommands
-----
Manifest 1.0.0.0 ActiveDirectory {Get-ADRootDSE, New-ADObject, ...}
Manifest 1.0.0.0 AppLocker      {Set-AppLockerPolicy, Get-AppLockerPolicy, ...}
Manifest 1.0.0.0 BitsTransfer   {Add-BitsFile, Remove-BitsFile, ...}
Manifest 1.0.0.0 CimCmdlets     {Get-CimAssociatedInstance, Get-CimInstance, ...}

```

Pour utiliser le module :

Import-Module ActiveDirectory



```

PS C:\> Import-Module ActiveDirectory

Loading Active Directory module for Windows PowerShell with default drive 'AD:'
[Progress bar]

```

Tutoriel original en anglais sur :

<http://blogs.msdn.com/b/rkamesh/archive/2012/01/17/how-to-add-active-directory-module-in-powershell-in-windows-7.aspx>

2.2. Script PowerShell

2.2.1. Afficher les utilisateurs :

```
# Test connexion AD et affichage des user  
  
# démarrer le 14/01/2015  
  
# Repris le 20/10/2015  
  
# Afficher les utilisateurs de l'AD sans filtre  
  
# activer les commandes spécifiques à Active Directory  
  
Import-Module ActiveDirectory  
  
# Chargement de toutes les propriétés, affichage uniquement les champs GivenName, Surname, reponsetest,  
description, distinguishedName  
  
Get-ADUser -Filter 'ObjectClass -eq "user"' -Property * | Format-List GivenName, Surname, reponsetest,  
description,distinguishedName  
  
# Toutes les propriétés de tous les comptes  
  
#Get-ADUser -Filter 'ObjectClass -eq "user"' -Properties * | Format-List *
```

2.2.2. Modifier le mot de passe d'un utilisateur :

```
# Test connexion AD et afficher UN user  
  
# démarrer le 20/01/2015  
  
# repris le 22/01/2015  
  
# Rechercher et afficher un utilisateur  
  
# Recherche faite avec nom et prénom( écrit en dure)  
  
# permet de modifier l'utilisateur sélectionner.  
  
# activer les commandes spécifiques à Active Directory  
  
Import-Module ActiveDirectory
```

Recherche en dure

\$sn = 'Lee'

\$givenName ="Bruce"

Write-Host "Recherche de l'utilisateur : " \$sn \$givenName

L'utilisateur trouver est stocké dans une variable

\$userWay = Get-ADUser -Filter{ (GivenName -eq \$givenName) -And (sn -eq \$sn) } -Property *

Write-Host "Utilisateur trouvé :"

\$userWay | Format-List GivenName, Surname, reponsetest, description,distinguishedName

On créer la chaine de connexion permettant de modifier le mot de passe

\$compte = [adsi]('LDAP://labocnam/'+\$userWay)

Write-Host "Tentative de changement de mot de passe..."

try {

\$compte.invoke("SetPassword","toto")

\$compte.SetInfo()

Write-Host "Mot de passe changé avec succès."

}

Catch [System.Net.WebException],[System.Exception]{

Write-Host "Échec:"

\$ErrorMessage = \$_.Exception.Message

\$FailedItem = \$_.Exception.ItemName

\$FailedItem

\$ErrorMessage

}

2.2.3. Cas particulier :

Recherche d'un utilisateur par nom et prénom, avec trois utilisateurs dont les nom et prénom sont identiques (Romain Test).

```
# Test connexion AD et afficher UN user

# démarrer le 20/10/2015 # Repris le 22/01/2015

# Test afficher user par recherche nom et prenom # et modifier le mot de passe

# avec 3 utilisateurs avec nom et prénom identiques


# activer les commandes spécifiques à Active Directory

Import-Module ActiveDirectory


# Nom et prenom à chercher

$sn = 'Test'

$givenName = 'Romain'

Write-Host "recherche de l'utilisateur :"$sn $givenName

# Les utilisateurs trouvés sont stocker dans une variable (tableau)

$userWay = Get-ADUser -Filter{ (GivenName -eq $givenName) -And (sn -eq $sn) -And(ObjectClass -eq "user") } -
Property *


# détermine le nombre de résultat trouvés (longueur du tableau)

$nbrRes = $userWay.length

# Si le résultat est multiple on affiche les propriétés de chacun des comptes

if ($nbrRes -ge 2) {

    Write-Host " Attention : " $userWay.length "utilisateurs ont été trouvés pour nom : " $sn "et prénom : "

    $givenname

    for ($i=0 ; $i -lt $nbrRes ; $i++) {

        Write-Host "-----"

        $user = $userWay[$i]

        Write-Host "utilisateur n°$i

        $user | Format-List -Property GivenName, Surname ,sAMAccountName
```

```

        Write-Host "-----"
    }

    # propose à l'utilisateur les différents résultats

    Write-Host ""

    Write-Host 'quel utilisateur êtes-vous?'

    $whatUser = read-host "entrée votre numéro"

    Write-Host ""

    Write-Host "-----"

    Write-Host "Vous avez choisi l'utilisateur répertorié:" $userWay[$whatUser] | Format-List -Property cn

    Write-Host "-----"

    Write-Host ""

    $compte = [adsisearcher]('LDAP://labocnam/' + $userWay[$whatUser])

    Write-Host ""

    Write-Host "Tentative de changement de mot de passe..."

    Write-Host ""

    try
    {
        $compte.invoke("SetPassword","toto")

        $compte.SetInfo()

        Write-Host "-----"

        Write-Host 'Mot de passe changé avec succès.'

        Write-Host "-----"
    }

    catch [System.Net.WebException],[System.Exception] {

        Write-Host "Échec:"

        $ErrorMessage = $_.Exception.Message

        $FailedItem = $_.Exception.ItemName

        $FailedItem

        $ErrorMessage

    }

```

```
}  
  
# Sinon on applique directement le mot de passe au résultat unique trouvé  
  
else {  
  
    $compte = [adsisearcher]('LDAP://labocnam/'+$userWay)  
  
    try    {  
  
        $compte.invoke("SetPassword","toto")  
  
        $compte.SetInfo()  
  
        Write-Host "Mot de passe changé avec succès."  
  
    }  
  
  
  
    Catch [System.Net.WebException] , [System.Exception] {  
  
        Write-Host "Échec:"  
  
        $ErrorMessage = $_.Exception.Message  
  
        $FailedItem = $_.Exception.ItemName  
  
        $FailedItem  
  
        $ErrorMessage  
  
    }  
  
}
```

3. Visual Studio 2013 community

3.1. Script C# mode console.

Le script : presentationV3 est le résultat de différents scripts et recherches afin de modifier des paramètres Active Directory.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
/* Requis : ajout de la référence System.DirectoryServices */
using System.DirectoryServices;

/* Programme de présentation et centralisation des fonctions pour Active Directory
 * Auteur : Romain Ledru, stagiaire
 * Reprise: du script presentationV2
 *
 * Début : 19/01/2015 Romain Ledru, stagiaire
 * reprise le 21/01/2015 Romain Ledru, stagiaire
 * reprise le 23/01/2015 Romain Ledru, stagiaire
 * reprise le 26/01/2015 Romain Ledru, stagiaire
 */

/* Permet de :
 *      1-Afficher l'ensemble des utilisateurs (ok)
 *      2-Saisir un utilisateur (login) et modifier son mot de passe (ok)
 *      3-Ajouter un nouvel utilisateur (ok)
 *      4-Créer un utilisateur (script en dure) et le supprimer. (Ok)
 * 5-Saisir un utilisateur (login), demander sa réponse secrète et modifier son mot de passe. (Ok)
 *      6-Appliquer une réponse secrète a l'ensemble des utilisateurs. (Ok)
 *      7-Recherche d'utilisateur par nom (sn) et prénom (givenName).
 */

namespace presentationV3
{
    class Program
    {
        static void Main(string[] args)
        {
            Menu();
        }

        #region Fonctions

        /* Fonctions principales */

        /* Menu */
        public static void Menu()
        {
            int menuPick = 0;
            do
            {
                Console.WriteLine("");
                Console.WriteLine("////////////////////////////////////////");
```



```

Console.WriteLine("");
Console.WriteLine("Quel action désirez-vous effectuer?");
Console.WriteLine("1 : Afficher les utilisateurs.");
Console.WriteLine("2 : Modifier le mot de passe d'un utilisateur.");
Console.WriteLine("3 : Ajouter un nouvel utilisateur.");
Console.WriteLine("4 : Supprimer un utilisateur.");
Console.WriteLine("5 : Demander réponse secrète utilisateur pour modifier son mot de
    passe.");

Console.WriteLine("6 : Appliquer une réponse a l'ensemble des utilisateurs.");
Console.WriteLine("7 : Recherche d'utilisateur avec son nom et prénom.");
Console.WriteLine("8 : Quitter.");
Console.WriteLine("");
Console.WriteLine("////////////////////////////////////////");
Console.WriteLine("");

// contrôle de saisie sélection menu
menuPick = controlMenu();

// variables pour connection AD admin
string aD = "labocnam";
string admin = "administrateur";
string adminPass = "P@ssword";

switch (menuPick)
{
    case 1:
        Console.WriteLine("Menu1.");
        Console.WriteLine("-----");
        printUsers(aD, admin, adminPass);
        // Repasse la valeur 0 pour revenir au menu.
        menuPick = 0;
        Console.WriteLine("-----");
        break;

    case 2:
        Console.WriteLine("Menu2.");
        Console.WriteLine("-----");
        SetUserPass(aD, admin, adminPass);
        menuPick = 0;
        Console.WriteLine("-----");
        break;

    case 3:
        Console.WriteLine("Menu3.");
        Console.WriteLine("-----");
        addUser(aD, admin, adminPass);
        menuPick = 0;
        Console.WriteLine("-----");
        break;

    case 4:
        Console.WriteLine("Menu4.");
        Console.WriteLine("-----");
        delUser(aD, admin, adminPass);

```

```

        menuPick = 0;
        Console.WriteLine("-----");
        break;

    case 5:
        Console.WriteLine("Menu5.");
        Console.WriteLine("-----");
        SetUserPassReqRep(aD, admin, adminPass);
        menuPick = 0;
        Console.WriteLine("-----");
        break;

    case 6:
        Console.WriteLine("Menu6.");
        Console.WriteLine("-----");
        setRepAllUser(aD, admin, adminPass);
        menuPick = 0;
        Console.WriteLine("-----");
        break;

    case 7:
        Console.WriteLine("Menu7.");
        Console.WriteLine("-----");
        FindUserSnGivenName(aD, admin, adminPass);
        menuPick = 0;
        Console.WriteLine("-----");
        break;
    }
}
while (menuPick == 0);
}

// -----
/* printUsers(menu1) :Fonction connexion AD avec un compte privilégié(en dure dans le
programme) et affiche les utilisateurs et leurs détails(attributs)
* Fonctions requises :
*/
// -----

public static void printUsers(string domainName, string userNameAdm, string AdmPassword)
{
    try
    {
        DirectoryEntry Ldap = new DirectoryEntry("LDAP://" + domainName, userNameAdm,
AdmPassword, AuthenticationTypes.Secure);
        DirectorySearcher searcher = new DirectorySearcher(Ldap);
        searcher.Filter = "(objectClass=user)";
        foreach (SearchResult result in searcher.FindAll())
        {
            DirectoryEntry DirEntry = result.GetDirectoryEntry();
            Console.WriteLine("Login : " + DirEntry.Properties["samaccountname"].Value);
            Console.WriteLine("Nom : " + DirEntry.Properties["sn"].Value);
            Console.WriteLine("Prénom : " + DirEntry.Properties["givenName"].Value);
            Console.WriteLine("Le CN : " + DirEntry.Properties["CN"].Value);
        }
    }
}

```

```

        Console.WriteLine("Tél : " + DirEntry.Properties["TelephoneNumber"].Value);
        Console.WriteLine("Description : " + DirEntry.Properties["description"].Value);
        Console.WriteLine("Service, localisation : " +
DirEntry.Properties["distinguishedName"].Value);
        Console.WriteLine("réponse : " + DirEntry.Properties["info"].Value);

        Console.WriteLine("-----");
        DirEntry.Close();
        DirEntry.Dispose();
        Ldap.Close();
        Ldap.Dispose();
    }
}
catch (Exception Ex)
{
    Console.WriteLine(Ex.Message);
}
}

// -----
/* SetUserPass(menu2) : Fonction permettant de saisir un utilisateur(fonction userPick)
 * et de modifier son mot de passe (fonction controlPassPick) */

public static void SetUserPass(string domainName, string userNameAdm, string AdmPassword)
{
    string userName, newPassword = "";
    try
    {
        DirectoryEntry directionEntry = new DirectoryEntry("LDAP://" + domainName,
            userNameAdm, AdmPassword);

        try
        {
            // appel fonction UserPick()
            Console.WriteLine("Quel est votre identifiant utilisateur?");
            userName = userPick();
            DirectorySearcher search = new DirectorySearcher(directionEntry);
            search.Filter = "(SAMAccountName=" + userName + ")";
            SearchResult result = search.FindOne();

            if (result != null)
            {
                DirectoryEntry userEntry = result.GetDirectoryEntry();

                if (userEntry != null)
                {
                    Console.WriteLine("Utilisateur " + userName + " trouvé:");
                    // changement du mot de passe avec appel fonction SelectMyPassword()
                    newPassword = selectMyPassword();
                    userEntry.Invoke("SetPassword", new object[] { newPassword });
                    userEntry.CommitChanges();
                    Console.WriteLine("La modification du mot de passe a été prise en
                                compte.");
                    // fermeture des connexions
                    userEntry.Close();
                    userEntry.Dispose();
                    directionEntry.Close();
                    directionEntry.Dispose();
                }
            }
        }
    }
}

```

```

        }

        }
        // Renvois erreur si l'utilisateur saisie avec la fonction userPick n'existe pas
dans la base AD
        else { Console.WriteLine("Utilisateur " + userName + " est introuvable."); }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }

    }

    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}

// -----
/* addUser(menu3) : Fonction permettant de saisir un NOUVEL utilisateur de l'ajouter a l'AD
et de modifier son mot de passe */
// -----
public static void addUser(string domainName, string userNameAdm, string AdmPassword)
{
    string login, nom, prenom, agence, service, passWd;
    try
    {
        DirectoryEntry de = new DirectoryEntry("LDAP://" + domainName, userNameAdm,
AdmPassword, AuthenticationTypes.Secure);
        while (de != null)
        {
            try
            {
                Console.WriteLine("bonjours vous allez créer un nouvel utilisateur");

                Console.WriteLine("login?:");
                login = Convert.ToString(Console.ReadLine());

                Console.WriteLine("nom?:");
                nom = Convert.ToString(Console.ReadLine());

                Console.WriteLine("prenom?:");
                prenom = Convert.ToString(Console.ReadLine());

                Console.WriteLine("Agence?(Paris,Tokyo,NewYork):");
                agence = Convert.ToString(Console.ReadLine());

                Console.WriteLine("Service?(Vente,SAV,Comptabilité):");
                service = Convert.ToString(Console.ReadLine());

                Console.WriteLine("Passeword?:");
                passWd = Convert.ToString(Console.ReadLine());

                DirectoryEntry user = de.Children.Add("CN=" + login + ",OU=" + service +
", OU=" + agence, "user");
            }
            catch { }
        }
    }
}

```

```

        user.Properties["givenname"].Add(nom);
        user.Properties["sn"].Add(prenom);
        user.Properties["samAccountName"].Add(login);

        // importe changement
        user.CommitChanges();
        user.Invoke("SetPassword", new object[] { passWd });

        //activer le compte
        // Valeur a 0x0002 pour le désactiver
        user.Properties["userAccountControl"].Value = 0x0200;
        user.CommitChanges();
        Console.WriteLine("Utilisateur crée:");
        de.Dispose();
        user.Dispose();
        de = null;
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
}

// -----
/* delUser(menu 4) : Fonction permettant de créer un utilisateur(sript en dure)
   * et de le supprimer */
// -----
public static void delUser(string domainName, string userNameAdm, string AdmPassword)
{
    try
    {
        DirectoryEntry de = new DirectoryEntry("LDAP://" + domainName, userNameAdm,
        AdmPassword);
        try
        {
            //--- script création user dans paris vente
            Console.WriteLine("Création de l'utilisateur : User Test ,cn=userScript2 ");
            DirectoryEntry user = de.Children.Add("cn=userScript, OU=Vente, OU=Paris",
"user");

            user.Properties["SAMAccountName"].Add("userScript");

            user.Properties["sn"].Add("User");

            user.Properties["givenName"].Add("Test");

```

```

        user.Properties["description"].Add("Compte de test créé par le code");

        // On envoie les modifications au serveur
        user.CommitChanges();

        // Pareil pour le mdp
        user.Invoke("SetPassword", new object[] { "toto" });
        user.Properties["userAccountControl"].Value = 0x0200;
        user.CommitChanges();
        Console.WriteLine("Utilisateur créer");
        Console.WriteLine("Login : " + user.Properties["samaccountname"].Value);
        Console.WriteLine("Nom : " + user.Properties["sn"].Value);
        Console.WriteLine("Prénom : " + user.Properties["givenName"].Value);
        Console.WriteLine("Le CN : " + user.Properties["CN"].Value);
        Console.WriteLine("-----");
        user.Dispose();

    }

    catch (Exception Ex)
    {
        Console.WriteLine(Ex.Message);
    }
    try
    {
        //-----
        Console.WriteLine("recherche de l'utilisateur a supprimer :");
        DirectorySearcher search = new DirectorySearcher(de);
        search.Filter = "(samaccountname=userScript)";
        SearchResult result = search.FindOne();
        DirectoryEntry userAsupr = result.GetDirectoryEntry();
        DirectoryEntry ou = userAsupr.Parent;

        Console.WriteLine("Login : " + userAsupr.Properties["samaccountname"].Value);
        Console.WriteLine("Nom : " + userAsupr.Properties["sn"].Value);
        Console.WriteLine("Prénom : " + userAsupr.Properties["givenName"].Value);
        Console.WriteLine("Le CN : " + userAsupr.Properties["CN"].Value);

        //--
        ou.Children.Remove(userAsupr);
        ou.CommitChanges();
        Console.WriteLine("Utilisateur supprimé");
        //----
    }

    catch (Exception Ex)
    {
        Console.WriteLine(Ex.Message);
    }
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}

```

```

    }

    // -----
    /* SetUserPassReqRep(menu5) : Fonction permettant de saisir un utilisateur(fonction
userPick)
    * de contrôler son accès en demandant ça réponse secrète.
    * et de modifier son mot de passe (fonction controlPassPick) */

    public static void SetUserPassReqRep(string domainName, string userNameAdm, string
AdmPassword)
    {
        string userName, newPassword = "";
        string repSaisi, repVrai;
        bool repValid = false;
        try
        {
            DirectoryEntry directionEntry = new DirectoryEntry("LDAP://" + domainName,
userNameAdm, AdmPassword);
            try
            {
                Console.WriteLine("Quel est votre identifiant utilisateur?");
                userName = userPick();
                DirectorySearcher search = new DirectorySearcher(directionEntry);
                search.Filter = "(SAMAccountName=" + userName + ")";
                SearchResult result = search.FindOne();

                if (result != null)
                {
                    DirectoryEntry userEntry = result.GetDirectoryEntry();

                    if (userEntry != null)
                    {
                        Console.WriteLine("Utilisateur " + userName + " trouvé:");
                        repVrai = Convert.ToString(userEntry.Properties["reponseTest"].Value);

                        //-----Début Code vérifié réponse
                        do
                        {
                            Console.WriteLine("Quel est votre réponse secrète?");
                            repSaisi = Convert.ToString(Console.ReadLine());
                            if (repSaisi == repVrai)
                            {
                                repValid = true;
                            }
                        } while (repValid == false);

                        //-----Fin Code vérifié réponse

                        // changement du mot de passe avec appel fonction SelectMyPassword()

```



```

        newPassword = selectMyPassword();
        userEntry.Invoke("SetPassword", new object[] { newPassword });
        userEntry.CommitChanges();
        Console.WriteLine("La modification du mot de passe à été prise en
compte.");

        // fermeture des connexions
        userEntry.Close();
        userEntry.Dispose();
        directionEntry.Close();
        directionEntry.Dispose();
    }

}
// Renvois erreur si l'utilisateur saisie avec la fonction userPick n'existe pas
dans la base AD
else { Console.WriteLine("Utilisateur " + userName + " est introuvable."); }
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
}

// -----
/* SetAllRepUser(menu6) : Fonction permettant de changer la réponse secrète.
 * de tous les utilisateurs
 */
// -----
public static void setRepAllUser(string domainName, string userNameAdm, string AdmPassword)
{
    string rep,nom,prenom,newAnswer;
    try
    {
        DirectoryEntry Ldap = new DirectoryEntry("LDAP://" + domainName, userNameAdm,
AdmPassword, AuthenticationTypes.Secure);
        DirectorySearcher searcher = new DirectorySearcher(Ldap);
        searcher.Filter = "(objectClass=user)";
        Console.WriteLine("Quel réponse désirez-vous appliquer a l'ensemble des
utilisateurs?");
        rep = Console.ReadLine();

        foreach (SearchResult result in searcher.FindAll())
        {
            DirectoryEntry cetUser = result.GetDirectoryEntry();
            Console.WriteLine("Utilisateur " + cetUser.Properties["SAMAccountName"].Value +
" trouvé" + "(ancienne réponse : " + cetUser.Properties["reponsetest"].Value + ")");

            //----
            /* permet de changer la réponse avec des valeurs des attributs user de l'AD */

```

```

        nom = Convert.ToString(cetUser.Properties["SAMAccountName"].Value);
        prenom = Convert.ToString(cetUser.Properties["cn"].Value);
        newAnswer = nom.Substring(0, 3) + prenom.Substring(0, 3);
        //----
        // Changement de la réponse secrète
        cetUser.Properties["reponsetest"].Value = rep;
        Console.WriteLine("Utilisateur " + cetUser.Properties["SAMAccountName"].Value +
" modifié" + "(nouvelle réponse : " + cetUser.Properties["reponsetest"].Value + ")");
        cetUser.CommitChanges();

        Console.WriteLine("-----");
        cetUser.Close();
        cetUser.Dispose();
        Ldap.Close();
        Ldap.Dispose();
    }
}
catch (Exception Ex)
{
    Console.WriteLine(Ex.Message);
}

}

// -----
/* FindUserSnGivenName(menu7) : Fonction permettant de rechercher un utilisateur avec son
nom et prénom
*/
// -----
public static void FindUserSnGivenName(string domainName, string userNameAdm, string
AdmPassword)
{
    string nom, prenom;
    nom = "Test";
    prenom = "Romain";
    try
    {
        DirectoryEntry Ldap = new DirectoryEntry("LDAP://" + domainName, userNameAdm,
AdmPassword, AuthenticationTypes.Secure);
        DirectorySearcher searcher = new DirectorySearcher(Ldap);

        /* filtre */
        searcher.Filter = "(&" + "(sn=" + nom + ")" + "(givenName=" + prenom + ")" + ")"; //
Quentin Tarantino Test Romain

        // Si aucun utilisateur n'est trouvé.
        SearchResult resultTest = searcher.FindOne();
        if (null == resultTest)
        {
            Console.WriteLine("Utilisateur introuvable.");
        }
        // sinon on affiche ses attributs
        else
        {
            Console.WriteLine("-----");
            Console.WriteLine("Utilisateur(s) trouvés pour nom: " + nom + ", prénom: " +
prenom + ".");

```

```

        Console.WriteLine("-----");
        foreach (SearchResult result in searcher.FindAll())
        {
            // On récupère l'objet trouvé lors de la recherche
            DirectoryEntry DirEntry = result.GetDirectoryEntry();
            if (null == DirEntry)
            {
                Console.WriteLine("Utilisateur introuvable.");
            }

            Console.WriteLine("Login : " +
DirEntry.Properties["SAMAccountName"].Value);

            Console.WriteLine("Nom : " + DirEntry.Properties["sn"].Value);

            Console.WriteLine("Prénom : " + DirEntry.Properties["givenName"].Value);

            Console.WriteLine("Email : " + DirEntry.Properties["mail"].Value);

            Console.WriteLine("Tél : " +
DirEntry.Properties["TelephoneNumber"].Value);

            Console.WriteLine("Description : " +
DirEntry.Properties["description"].Value);

            Console.WriteLine("-----");
        }

    }
    // Fermeture de la connexion
    Ldap.Close();
    Ldap.Dispose();
}
catch (Exception Ex)
{
    Console.WriteLine(Ex.Message);
}
}

// -----
/* Fonctions de saisies et contrôles */
// -----

/* Fonction contrôle saisie menu
* Boucle tant que l'utilisateur n'as pas saisie une entrée de type int */
// -----
public static int controlMenu()
{
    int num = 0;
    string UserEntry = "";
    bool IsOk = false;
    string quit = "8";
    do
    {
        Console.WriteLine("- - - - -");
        Console.WriteLine("Entrez un nombre pour naviguer, ou " + quit + " pour quitter.");
    }
}

```

```

        Console.WriteLine("- - - - -");
        UserEntry = Console.ReadLine();
        IsOk = int.TryParse(UserEntry, out num);
    } while (!IsOk && UserEntry != quit);
    return Convert.ToInt32(UserEntry);
}

/* fonction choix de l'user
* UserPick : permet à l'utilisateur de saisir le compte user Active Directory à modifier
*             redemande la saisie, si celle-ci est null ou vide.
*             La vérification de l'existence de l'user dans l'AD ne ce fait pas ici
*             Supprime tous les espaces de la saisie
*/
// -----
public static string userPick()
{
    string userSelect;
    string trimUserSelect;
    do
    {
        userSelect = Convert.ToString(Console.ReadLine());
        trimUserSelect = userSelect.Trim();

    } while (trimUserSelect == null || trimUserSelect == String.Empty);

    return userSelect;
}

/* fonction choix du mot de passe
* SelectMyPassword : permet a l'utilisateur de saisir lui-même son mots de passe.
*                   Accepte les chaines de caractères, les caractères spéciaux et numériques.
*                   Redemande la saisie, si celle-ci est null ou vide.
*                   Supprime tous les espaces de la saisie( le mieux serais de les détectés et de
renvoyer une alerte).
*                   /\ Mot de passe saisie en claire
*                   https://social.msdn.microsoft.com/Forums/fr-FR/55e423d6-7917-4e7d-822d-ce1adcd547c6/comment-masquer-les-caractres-de-mot-de-passe-dans-la-console-lors-de-la-saisie-?forum=visualcsharpfr
*/
public static string selectMyPassword()
{
    string selectPass;
    string trimSelectPass;
    do
    {
        Console.WriteLine("Veuillez saisir votre nouveau mot de passe:");
        selectPass = Convert.ToString(Console.ReadLine());
        trimSelectPass = selectPass.Trim();
    } while (trimSelectPass == null || trimSelectPass == String.Empty);
    return selectPass;
}

#endregion
}
}

```

3.1.2 Script C# IHM, asp.net

Voir les différentes parties du projet web : _UsersFindSetPswd_V4(crypte Rep)

Partie principales :

Programme

```
// USING SYSTEM
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

// Système using à ajouter

// Active directory
using System.DirectoryServices;
using System.DirectoryServices.ActiveDirectory;

// Utilisation dataGridView
using System.Data;

// Cryptage
using System.Text;
using System.Security.Cryptography;

/* Par Ledru romain, stagiaire au CNAM de Nantes
* démarré : 27/01/2015
* Repris : 28/01/2015
* Repris : 30/01/2015
* Repris : 03/02/2015
* Repris : 04/02/2015
* Repris : 05/02/2015
```

* Repris : 06/02/2015

* Repris : 07/02/2015

*

*

* Page web : Recherche du/des contrôleurs de domaine.

* : Permet une recherche utilisateur par la saisie de son nom.

* : Affiche le/les résultats trouvés pour le nom saisi ou renvoie un message d'erreur.

* : Demande la réponse secrète pour le nom sélectionné afin d'authentifier l'utilisateur.

* : +Captha de protection anti-bot.

* : Permet de modifier le mot de passe après double saisie de validation.

* : Pour changer le mot de passe utilisateur, une connexion administrateur est écrite en dur.

* : Chaque modification de mot de passe est enregistrée dans un fichier log:

* : ip de poste client, nom de la session client, nom de la session modifiée, date et heure

*/

```
public partial class _default : System.Web.UI.Page
```

```
{
    #region VARIABLES
    // PARAMETRE CONNEXION AD
    public class varGlobal
    {
        //récupérer le nom du contrôleur de domaine auquel la machine est connectée (ici "labocnam")
        public static string aD =
System.Net.NetworkInformation.IPGlobalProperties.GetIPGlobalProperties().DomainName;

        // Connexion AD information admin
        public static string admin = "administrateur";
        public static string adminPass = "P@ssword";

        // var globale pour le nom que saisira l'utilisateur
        public static string nomUtilisateur;
```

```
// filtre global

public static DirectorySearcher ActiveDirectory = new DirectorySearcher(ConnexionLDAP.etablirConnexion());

// userEntry global

public static SearchResult resultatAD;

public static DirectoryEntry userEntry;

}

#endregion VARIABLES

//-----

protected void Page_Load(object sender, EventArgs e)
{

    // Détection des CD (contrôleurs de domaine)

    DirectoryContext adDispo = new DirectoryContext(DirectoryContextType.Domain, varGlobal.ad);

    // get the domain object

    System.DirectoryServices.ActiveDirectory.Domain domain;

    domain = System.DirectoryServices.ActiveDirectory.Domain.GetDomain(adDispo);

    lbListCD.Text = "Liste des contrôleurs pour le domaine trouvé: </br>";

    // get the domain controllers belonging to that domain

    DomainControllerCollection dcc = domain.FindAllDiscoverableDomainControllers();

    foreach (DomainController controller in dcc)
    {

        lbListCD.Text += controller + " </br>";

    }

    // CONNEXION
```



```
try
{

    // initial connexion

    DirectoryEntry coAD;

    coAD = ConnexionLDAP.etablirConnexion();

    DirectorySearcher searcher = new DirectorySearcher(coAD);

    // .Path : Obtient ou définit le chemin d'accès de ce DirectoryEntry.
    // .Existe: détermine si le chemin d'accès spécifié représente une entrée réelle dans le service d'annuaire.
    if (DirectoryEntry.Exists(coAD.Path))
    {
        lbMsgConnexion.Text = "Serveur AD atteint: " + coAD.Path;
    }

    else // else jamais utilisé, en cas d'erreur (ex:labocnam1), le catch a la priorité.
    {
        lbMsgConnexion.Text = "Serveur introuvable : " + coAD.Path;
    }

}

catch (Exception msg)
{
    lbMsgConnexion.Text += "</br> catch erreur 1 (CO AD): </br>" + msg.ToString();
}

}
```

```
//----- Recherche nom utilisateur -----  
  
protected void btOK_Click(object sender, EventArgs e)  
{  
  
    panelRes.Visible = false;  
  
    lbMessage.Text = "";  
  
    // Contrôle de la saisie utilisateur (oblige la saisie du champ "Nom").  
    if (tbNom.Text == null || tbNom.Text == String.Empty || tbNom.Text.Trim() == String.Empty ||  
    tbNom.Text.Trim().StartsWith("*"))  
    {  
        // RAZ textbox  
        textRAZ(lbReqNom, lbMessage);  
  
        lbReqNom.Text = ("Vous n'avez pas saisi de nom.");  
    }  
    else  
    {  
        // RAZ textbox  
        textRAZ(lbReqNom, lbMessage);  
  
        // Recherche en fonction de la saisie du "nom" utilisateur.  
        varGlobal.nomUtilisateur = tbNom.Text;  
        varGlobal.ActiveDirectory.Filter = "(sn=" + varGlobal.nomUtilisateur + ")";  
  
        // Compteur nombre utilisateur  
        int nbUserFind = 0;  
  
        // Tableau pour stocker les utilisateurs potentiellement trouvés  
        Utilisateurs unUser = new Utilisateurs();  
  
        // Formatage dataGridView
```

```

DataTable table = new DataTable();

table.Columns.Add("Login", typeof(string));

table.Columns.Add("Nom", typeof(string));

table.Columns.Add("Prénom", typeof(string));


// Recherche des utilisateurs et affichage du résultat
foreach (SearchResult result in varGlobal.ActiveDirectory.FindAll())
{

    nbUserFind += 1;

    panelRes.Visible = true;


    DirectoryEntry userEntry = result.GetDirectoryEntry();

    // Récupération des attributs login, nom prénom de la session utilisateur trouvée
    unUser.samaccountname = Convert.ToString(userEntry.Properties["samaccountname"].Value);
    unUser.sn = Convert.ToString(userEntry.Properties["sn"].Value);
    unUser.givenName = Convert.ToString(userEntry.Properties["givenName"].Value);

    // Ajout de la récupération dans un tableau
    table.Rows.Add(unUser.samaccountname, unUser.sn, unUser.givenName);

}

lbMessage.Text = ("Utilisateur: \" " + varGlobal.nomUtilisateur + "\" obtient: " + nbUserFind + " Résultat(s).");

// Passage du tableau dans le dataGridView et affichage du résultat
GridView1.DataSource = table;

GridView1.DataBind();

// Si aucun utilisateur n'est trouvé, on revoit un message d'information.
if (nbUserFind == 0)
{

```

```
// RAZ textbox

textRAZ(lbReqNom, lbMessage);

lbMessage.Text = ("Aucun utilisateur trouvé pour le nom :" + tbNom.Text + ".");

}

}

}

//----- DGV SELECTED -----

protected void GridView1_SelectedIndexChanged(object sender, EventArgs e)
{

    int Row = GridView1.SelectedRow.RowIndex;

    string logUtilisateur = GridView1.Rows[Row].Cells[1].Text;

    try
    {

        varGlobal.resultatAD = varGlobal.ActiveDirectory.FindOne();//type du filtre

        varGlobal.ActiveDirectory.Filter = "(samaccountname=" + logUtilisateur + ")";

        SearchResult result = varGlobal.ActiveDirectory.FindOne();

        varGlobal.userEntry = result.GetDirectoryEntry();

        // Si un utilisateur est trouvé pour ce nom, on affiche le panel suivant

        lbUserSelect.Text = "Vous avez sélectionné l'utilisateur dont le login est : " + logUtilisateur;

        pnlAskReponse.Visible = true;

        //chargement image pour le captcha

        SetCaptchaText();

        // le bouton de recherche est désactivé
```

```
        btOK.Enabled = false;

    }

    catch (Exception msgSelect)

    {

        pnlAskReponse.Visible = false;

        lbUserSelect.Text = "catch erreur 2 (sélection utilisateur). </br>" + msgSelect.ToString();

    }

}

//-----BUTTON VALIDE REPONSE-----

protected void btReponse_Click(object sender, EventArgs e)

{

    if (Session["Captcha"].ToString() != tbCaptcha.Text.Trim())

    {

        pnlSetPswd.Visible = false;

        lbMsgCaptcha.Text = "Le captcha n'est pas identique à l'image ci-contre.";

        //rechargement image pour le Captcha

        SetCaptchaText();

    }

    else

    {

        lbMsgCaptcha.Text = "Captcha bon.";

        // Contrôle réponse utilisateur

        if (tbReponse.Text.ToString() == deCryptage(varGlobal.userEntry.Properties["info"].Value.ToString()))

        {

            lbMsgReponse.Text = "Votre réponse est correcte.";
```

```
// le bouton de reponse est désactivé

btReponse.Enabled = false;

pnlSetPswd.Visible = true;

}

else
{
    lbMsgReponse.Text = "Votre réponse est incorrecte.";
    pnlSetPswd.Visible = false;
    //rechargement image pour le captcha
    SetCaptchaText();
}
}

//-----BUTTON RESET CAPTCHA-----

protected void btCaptcha_Click(object sender, EventArgs e)
{
    SetCaptchaText();
}

//-----BOUTTON SET PASSWORD -----

protected void btSetPass_Click(object sender, EventArgs e)
{
    if (tbPswd.Text == String.Empty)
    {
```

```
lbMsgSetPass.Text = "Erreur: Le mot de passe ne peut être nul ou vide.";

}

else

{

    if (tbPswd.Text.ToString() == tbPswValid.Text.ToString())

    {

        try

        {

            varGlobal.userEntry.Invoke("SetPassword", new object[] { tbPswValid.Text.ToString() });

            varGlobal.userEntry.CommitChanges();

            lbMsgSetPassValid.Text = "Mot de passe changé avec succès de l'utilisateur (login) : " +
varGlobal.userEntry.Properties["samaccountname"].Value.ToString() + ".";

            // Appel de la fonction report log

            reportLog();

            // Fermeture entrée utilisateur

            varGlobal.userEntry.Dispose();

            // on cache le bouton de changement

            btSetPass.Enabled = false;

        }

        catch (Exception msgPass)

        {

            lbMsgSetPassValid.Text = "catch erreur 3 (setPswd).| | " + msgPass;

        }

    }

    else

    {

        lbMsgSetPassValid.Text = "Erreur: La confirmation et le mot de passe ne correspondent pas.";

    }

}
```



```
}  
  
}  
  
  
# region FONCTIONS  
  
  
// cryptage  
static string Cryptage(string clearPassword)  
{  
    byte[] bytes = Encoding.UTF8.GetBytes(clearPassword);  
  
    byte[] protectedBytes = ProtectedData.Protect(bytes, null, DataProtectionScope.CurrentUser);  
  
    return Convert.ToBase64String(protectedBytes);  
}  
  
// décryptage  
static string deCryptage(string protectedPassword)  
{  
    byte[] protectedBytes = Convert.FromBase64String(protectedPassword);  
  
    byte[] bytes = ProtectedData.Unprotect(protectedBytes, null, DataProtectionScope.CurrentUser);  
  
    return Encoding.UTF8.GetString(bytes);  
}  
  
  
  
// Reporting et fichier log  
private void reportLog()  
{  
    // FICHIER LOG  
  
    /* adresse ip poste client  
  
    * fonction qui ne renvoie pas de résultat lisible dans le contexte actuel  
  
    * il est certifié qu'elle fonctionne dans un "vrai" réseau
```

```

* En attendant on utilisera context.Request.ServerVariables["REMOTE_ADDR"]

* pour récupérer une @ip sous forme :::1

*/

System.Web.HttpContext context = System.Web.HttpContext.Current;

string ipAddress = context.Request.ServerVariables["HTTP_X_FORWARDED_FOR"];

if (!string.IsNullOrEmpty(ipAddress))

{

    string[] addresses = ipAddress.Split(',');

    if (addresses.Length != 0)

    {

        lbMsgInfoCo.Text += addresses[0];

    }

}

//écriture fichier log (pour le site^une fois publié: "~/Log/logSetPass.txt" ) ||
@"C:\Users\Administrateur\Documents\Visual Studio 2013\WebSites\LOG_SET_AD\logSetPass.txt"

using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@"C:\Users\Administrateur\Documents\Visual Studio
2013\WebSites\LOG_SET_AD\logSetPass.txt", true))

{

    // Les informations de l'ip du poste client, le nom de la machine client, le nom de la session client, le compte
modifié et la date

    // sont respectivement stockés dans les colonnes 0 à 4

    // Le nom de la session client ne fonctionne pas.

    // HttpContext.Current.Request.UserHostAddress == context.Request.ServerVariables["REMOTE_ADDR"]

    string[] tabLog = { HttpContext.Current.Request.UserHostAddress, Environment.MachineName,
Environment.UserName, varGlobal.userEntry.Properties["samaccountname"].Value.ToString(),
DateTime.Now.ToString() };

    // Puis écrites avec séparateur dans un fichier log : logSetPass.txt sous la forme :

    // ipSend xx posteSend xx NomSessionSend xx LoginModifie xx Date

    file.WriteLine(tabLog[0] + " " + tabLog[1] + " " + tabLog[2] + " " + tabLog[3] + " " + tabLog[4]);

}

```

```
}

// CAPTCHA

private void SetCaptchaText()

{
    Random oRandom = new Random();

    int iNumber = oRandom.Next(100000, 999999);

    Session["Captcha"] = iNumber.ToString();

}

// CONNEXION ACTIVE DIRECTORY

public static class ConnexionLDAP

{
    public static DirectoryEntry etablirConnexion()

    {
        DirectoryEntry coAD = new DirectoryEntry("LDAP://" + varGlobal.aD, varGlobal.admin, varGlobal.adminPass,
AuthenticationTypes.Secure);

        return coAD;

    }

}

// Fonction remise à zéro des champs textes

public static void textRAZ(Label lb1, Label lb2)

{
    // RAZ textbox

    lb1.Text = ("");
```

```

    lb2.Text = ("");

}

#endregion

}

```

Interface graphique

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="default.aspx.cs" Inherits="_default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>

    <title>Changer le mot de passe.</title>

    <link href="css/monStyle.css" rel="stylesheet" type="text/css" />
</head>
<body>

    <form id="form1" runat="server">
<!-- Panel Start -->

        <asp:Panel class="mesTab" ID="panelStart" runat="server" Visible="true">

            <asp:Label ID="lbListCD" runat="server"></asp:Label>

            <br />

            <asp:Label ID="lbMsgConnexion" runat="server"></asp:Label>

            <br />

        </asp:Panel>
<!-- Panel recherche -->

        <asp:Panel class="mesTab" ID="panel0" runat="server" Visible="true">

            <table id="tabFindUser">

                <tr>

                    <td class="auto-style1" colspan="3">Cherche un utilisateur (* : champs obligatoires):</td>

```

```

</tr>

<tr>

    <td class="auto-style12">Nom :</td>

    <td class="auto-style5">

        <asp:TextBox ID="tbNom" runat="server" Width="151px" CssClass="textBox"></asp:TextBox>

    </td>

    <td>

        <asp:Label ID="lbReqNom" runat="server"></asp:Label>

    </td>

</tr>

<tr>

    <td class="auto-style12">

        <asp:Button ID="btOK" runat="server" OnClick="btOK_Click" Text="OK" />

    </td>

    <td class="auto-style2" colspan="2">

        <asp:Label ID="lbMessage" runat="server"></asp:Label>

    </td>

</tr>

</table>

</asp:Panel>

<!-- Panel Résultat recherche -->

<asp:Panel class="mesTab" ID="panelRes" runat="server" Visible="False">

    <asp:GridView ID="GridView1" runat="server" OnSelectedIndexChanged="GridView1_SelectedIndexChanged"
>

        <Columns>

            <asp:ButtonField ButtonType="Button" CommandName="Select" Text="Selectionner" />

        </Columns>

    </asp:GridView>

    <asp:Label ID="lbUserSelect" runat="server"></asp:Label>

</asp:Panel>

```

```
<!-- Panel Demande réponse secrète + captcha -->
```

```
<asp:Panel class="mesTab" ID="pnlAskReponse" runat="server" Visible="False">
```

```
<table id="tabAskAnswer">
```

```
<tr>
```

```
<td class="auto-style3" colspan="3">Quelle est votre réponse secrète?</td>
```

```
</tr>
```

```
<tr>
```

```
<td class="auto-style3">&nbsp;</td>
```

```
<td class="auto-style18" colspan="2">
```

```
<asp:TextBox ID="tbReponse" runat="server" AutoCompleteType="Disabled" CssClass="textBox"
Width="420px"></asp:TextBox>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td class="auto-style3">&nbsp;</td>
```

```
<td class="auto-style18" colspan="2">
```

```
<asp:Label ID="lbMsgReponse" runat="server"></asp:Label>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td class="auto-style3">Protection anti-bot</td>
```

```
<td class="auto-style19">
```

```
<asp:Image ID="imgCaptcha" runat="server" ImageUrl="Captcha.ashx" />
```

```
</td>
```

```
<td class="auto-style18">
```

```
<asp:Button ID="btCaptcha" runat="server" CssClass="tbCaptcha" OnClick="btCaptcha_Click" />
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td class="auto-style3">Recopiez la suite de nombres:</td>
```

```

        <td class="auto-style18" colspan="2">

            <asp:TextBox ID="tbCaptcha" runat="server" AutoCompleteType="Disabled"
CssClass="textBox"></asp:TextBox>

            <asp:Label ID="lbMsgCaptcha" runat="server"></asp:Label>

        </td>

    </tr>

    <tr>

        <td class="auto-style3">&nbsp;</td>

        <td class="auto-style18" colspan="2">

            <asp:Button ID="btReponse" runat="server" OnClick="btReponse_Click" Text="Valider" />

        </td>

    </tr>

</table>

</asp:Panel>

<!-- Panel Changement mot de passe -->

<asp:Panel class="mesTab" ID="pnlSetPswd" runat="server" Visible="False">

    <table id="tabSetPswd" >

        <tr>

            <td class="auto-style9">Votre nouveau mot de passe:</td>

            <td class="auto-style7">

                <asp:TextBox ID="tbPswd" runat="server" TextMode="Password" CssClass="textBox"
></asp:TextBox>

            </td>

            <td class="auto-style7">

                <asp:Label ID="lbMsgSetPass" runat="server"></asp:Label>

            </td>

        </tr>

        <tr>

            <td class="auto-style16">Confirmation du mot de passe :</td>

```

```

        <td class="auto-style17">

            <asp:TextBox ID="tbPswValid" runat="server" TextMode="Password"
CssClass="textBox"></asp:TextBox>

        </td>

        <td class="auto-style8" rowspan="2">

            <asp:Label ID="lbMsgSetPassValid" runat="server"></asp:Label>

        </td>

    </tr>

    <tr>

        <td class="auto-style16"></td>

        <td class="auto-style7">

            <asp:Button ID="btSetPass" runat="server" OnClick="btSetPass_Click" Text="Valider" />

        </td>

    </tr>

</table>

<asp:Label ID="lbMsgInfoCo" runat="server"></asp:Label>

</asp:Panel>

</form>

</body>

</html>

```

CSS

```

body {

    background: -moz-linear-gradient(top, rgba(0,0,0,0.65) 0%, rgba(0,0,0,0) 100%); /* FF3.6+ */

    background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,rgba(0,0,0,0.65)), color-
stop(100%,rgba(0,0,0,0))); /* Chrome,Safari4+ */

    background: -webkit-linear-gradient(top, rgba(0,0,0,0.65) 0%,rgba(0,0,0,0) 100%); /* Chrome10+,Safari5.1+ */

    background: -o-linear-gradient(top, rgba(0,0,0,0.65) 0%,rgba(0,0,0,0) 100%); /* Opera 11.10+ */

    background: -ms-linear-gradient(top, rgba(0,0,0,0.65) 0%,rgba(0,0,0,0) 100%); /* IE10+ */

    background: linear-gradient(to bottom, rgba(0,0,0,0.65) 0%,rgba(0,0,0,0) 100%); /* W3C */

```



```
filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#a6000000',
endColorstr='#00000000',GradientType=0 ); /* IE6-9 */
```

```
font-family:verdana, sans-serif;

font-size:100%;

width: auto;

height:1200px;
}
```

```
/* Style appliqué aux panels*/
```

```
.mesTab {

width :540px;

height :auto;

margin:10px;

padding:10px;

margin-left:25%;

background-color:#E4EFFF;

border:2px solid #EF1212;


/*arrondir les coins en haut à gauche et en bas à droite*/

-moz-border-radius:10px 0;

-webkit-border-radius:10px 0;

border-radius:10px 0;


box-shadow:2px 2px 10px gray;

-moz-box-shadow:2px 2px 10px gray;

-webkit-box-shadow:2px 2px 10px gray;
}
```

```
/* bouton reset du Captcha*/

.tbCaptcha {

    padding: 5px;

    margin: 5px;

    background-image: url('../images/btCaptha.png');

    width: 45px;

    height: 45px;

    border: 1px solid gray;

    box-shadow: 2px 2px 10px gray;

    -moz-box-shadow: 2px 2px 10px gray;

    -webkit-box-shadow: 2px 2px 10px gray;

}

/* style pour les textBox */

.textBox{

    border:1px solid gray;

    box-shadow:2px 2px 10px gray;

    -moz-box-shadow:2px 2px 10px gray;

    -webkit-box-shadow:2px 2px 10px gray;

}

/* style du DGV (tableau utilisateurs trouvés)*/

#GridView1 {

    padding: 10px;

    margin: 10px;

    border: 2px solid black;

    box-shadow: 2px 2px 10px gray;

    -moz-box-shadow: 2px 2px 10px gray;
```

```
-webkit-box-shadow: 2px 2px 10px gray;

background-color:#CFCFCF;

}

/* style du Captcha*/
#imgCaptcha{

    height : 80px;

    width:200px;


    border:1px solid gray;


    box-shadow:2px 2px 10px gray;
    -moz-box-shadow:2px 2px 10px gray;
    -webkit-box-shadow:2px 2px 10px gray;
}

/*style auto des cellules des différents tableaux */

.auto-style1 { height: 20px;

}

.auto-style5 {

    width: 160px;

}

#form1 {

    height: 780px;

}

.auto-style7 {
```

```
    height: 30px;

    width: 175px;

}

.auto-style8 {

    width: 175px;

}

.auto-style9 {

    height: 30px;

    width: 227px;

}

.auto-style10 {

    width: 225px;

}

.auto-style12 {

    width: 105px;

}

.auto-style16 {

    width: 225px;

    height: 30px;

}

.auto-style17 {

    width: 175px;

}

.auto-style18 {

    width: 440px;

}

.auto-style19 {

    width: auto;

}
```

Captcha

```
<%@ WebHandler Language="C#" Class="Captcha" %>

using System;

using System.Web;

using System.Drawing;

using System.IO;

using System.Web.SessionState;

using System.Drawing.Imaging;

using System.Drawing.Text;

using System.Drawing.Drawing2D;

public class Captcha : IHttpHandler, IReadOnlySessionState
{
    public void ProcessRequest(HttpContext context)
    {
        int iHeight = 80;

        int iWidth = 190;

        Random oRandom = new Random();

        int[] aBackgroundNoiseColor = new int[] { 150, 150, 150 };

        int[] aTextColor = new int[] { 0, 0, 0 };

        int[] aFontEmSizes = new int[] { 15, 20, 25, 30, 35 };

        string[] aFontNames = new string[]
        {
            "Comic Sans MS",
            "Arial",
            "Times New Roman",
```

```
"Georgia",
"Verdana",
"Geneva"
};

FontStyle[] aFontStyles = new FontStyle[]
{
    FontStyle.Bold,
    FontStyle.Italic,
    FontStyle.Regular,
    FontStyle.Strikeout,
    FontStyle.Underline
};

HatchStyle[] aHatchStyles = new HatchStyle[]
{
    HatchStyle.BackwardDiagonal, HatchStyle.Cross,
        HatchStyle.DashedDownwardDiagonal, HatchStyle.DashedHorizontal,
    HatchStyle.DashedUpwardDiagonal, HatchStyle.DashedVertical,
        HatchStyle.DiagonalBrick, HatchStyle.DiagonalCross,
    HatchStyle.Divot, HatchStyle.DottedDiamond, HatchStyle.DottedGrid,
        HatchStyle.ForwardDiagonal, HatchStyle.Horizontal,
    HatchStyle.HorizontalBrick, HatchStyle.LargeCheckerBoard,
        HatchStyle.LargeConfetti, HatchStyle.LargeGrid,
    HatchStyle.LightDownwardDiagonal, HatchStyle.LightHorizontal,
        HatchStyle.LightUpwardDiagonal, HatchStyle.LightVertical,
    HatchStyle.Max, HatchStyle.Min, HatchStyle.NarrowHorizontal,
        HatchStyle.NarrowVertical, HatchStyle.OutlinedDiamond,
    HatchStyle.Plaid, HatchStyle.Shingle, HatchStyle.SmallCheckerBoard,
        HatchStyle.SmallConfetti, HatchStyle.SmallGrid,
```

```
HatchStyle.SolidDiamond, HatchStyle.Sphere, HatchStyle.Trellis,
    HatchStyle.Vertical, HatchStyle.Wave, HatchStyle.Weave,
    HatchStyle.WideDownwardDiagonal, HatchStyle.WideUpwardDiagonal, HatchStyle.ZigZag
};

//Get Captcha in Session
string sCaptchaText = context.Session["Captcha"].ToString();

//Creates an output Bitmap
Bitmap oOutputBitmap = new Bitmap(iWidth, iHeight, PixelFormat.Format24bppRgb);
Graphics oGraphics = Graphics.FromImage(oOutputBitmap);
oGraphics.TextRenderingHint = TextRenderingHint.AntiAlias;

//Create a Drawing area
RectangleF oRectangleF = new RectangleF(0, 0, iWidth, iHeight);
Brush oBrush = default(Brush);

//Draw background (Lighter colors RGB 100 to 255)
oBrush = new HatchBrush(aHatchStyles[oRandom.Next
    (aHatchStyles.Length-1)], Color.FromArgb((oRandom.Next(100, 255)),
    (oRandom.Next(100, 255)), (oRandom.Next(100, 255))), Color.White);
oGraphics.FillRectangle(oBrush, oRectangleF);

System.Drawing.Drawing2D.Matrix oMatrix = new System.Drawing.Drawing2D.Matrix();
int i = 0;
for (i = 0; i <= sCaptchaText.Length - 1; i++)
{
    oMatrix.Reset();
    int iChars = sCaptchaText.Length;
    int x = iWidth / (iChars + 1) * i;
```

```
int y = iHeight / 2;

//Rotate text Random
oMatrix.RotateAt(oRandom.Next(-40, 40), new PointF(x, y));
oGraphics.Transform = oMatrix;

//Draw the letters with Random Font Type, Size and Color
oGraphics.DrawString
(
//Text
sCaptchaText.Substring(i, 1),
//Random Font Name and Style
new Font(aFontNames[oRandom.Next(aFontNames.Length - 1)],
        aFontEmSizes[oRandom.Next(aFontEmSizes.Length-1)],
        aFontStyles[oRandom.Next(aFontStyles.Length - 1)]),
//Random Color (Darker colors RGB 0 to 100)
new SolidBrush(Color.FromArgb(oRandom.Next(0, 100),
        oRandom.Next(0, 100), oRandom.Next(0, 100))),
x,
oRandom.Next(10, 40)
);
oGraphics.ResetTransform();
}

MemoryStream oMemoryStream = new MemoryStream();
oOutputBitmap.Save(oMemoryStream, System.Drawing.Imaging.ImageFormat.Png);
byte[] oBytes = oMemoryStream.GetBuffer();

oOutputBitmap.Dispose();
oMemoryStream.Close();
```



```
context.Response.BinaryWrite(oBytes);
```

```
context.Response.End();
```

```
}
```

```
public bool IsReusable
```

```
{
```

```
    get
```

```
    {
```

```
        return false;
```

```
    }
```

```
}
```

```
}
```