Managing everything in Active Directory via C# (Using System.DirectoryServices.AccountManagement)

Before .Net, managing Active Directory objects was a bit lengthy and you needed a good knowledge on the principal store to have your head around on what you want to do. We ususally use the System.DirectoryServices namespace but with .Net 3.5 they introduced System.DirectoryServices.AccountManagement which is manages directory objects independent of the System.DirectoryServices namespace.

So what are the advantages of using this? Everything is really simple in terms of managing a user, computer or group principal and performing queries on the stores are much faster thanks to the Fast Concurrent Bind (FSB) feature which caches the connection which decreases the number of ports used in the process.

The code is divided into several regions but here are the 5 key regions with their methods explained

Validate Methods

- ValidateCredentials – This Method will validate the users credentials.
- IsUserExpired – Checks if the User Account is Expired.
- IsUserExisiting – Checks if user exsists on AD.
- IsAccountLocked – Checks if user account is locked

Search Methods

- GetUser – This will return a UserPrincipal Object if the User Exists

User Account Methods

- SetUserPassword – This Method will set the Users Password
- EnableUserAccount – This Method will Enable a User Account
- DisableUserAccount – This Methoid will Disable the User Account
- ExpireUserPassword – This Method will Force Expire a Users Password
- UnlockUserAccount – This Method will unlocks a User Account
- CreateNewUser – This Method will Create a new User Directory Object
- DeleteUser – This Method will Delete an AD User based on Username.

Group Methods

- CreateNewGroup – This Method will create a New Active Directory Group
- AddUserToGroup – This Method will add a User to a group

- RemoveUserFromGroup – This Method will remove a User from a Group
- IsUserGroupMember – This Method will Validate whether the User is a Memeber of a Group
- GetUserGroups – This Method will return an ArrayList of a User Group Memberships

Helper Methods

- GetPrincipalContext – Gets the base principal context

```
using System;
using System.Collections;
using System.Text;
using System.DirectoryServices.AccountManagement;
using System.Data;
using System.Configuration;

public class ADMethodsAccountManagement
{

#region Variables

private string sDomain = "test.com";
private string sDefaultOU = "OU=Test Users,OU=Test,DC=test,DC=com";
private string sDefaultRootOU = "DC=test,DC=com";
private string sServiceUser = @"ServiceUser";
private string sServicePassword = "ServicePassword";

#endregion
#region Validate Methods

/// <summary>
/// Validates the username and password of a given user
/// </summary>
/// <param name="sUserName">The username to validate</param>
/// <param name="sPassword">The password of the username to validate</param>
/// <returns>Returns True of user is valid</returns>
public bool ValidateCredentials(string sUserName, string sPassword)
{
    PrincipalContext oPrincipalContext = GetPrincipalContext();
    return oPrincipalContext.ValidateCredentials(sUserName, sPassword);

}

/// <summary>
/// Checks if the User Account is Expired
/// </summary>
/// <param name="sUserName">The username to check</param>
/// <returns>Returns true if Expired</returns>
public bool IsUserExpired(string sUserName)
{
    UserPrincipal oUserPrincipal = GetUser(sUserName);
```

```
    if (oUserPrincipal.AccountExpirationDate != null)
    {
        return false;
    }
    else
    {
        return true;
    }
}

/// <summary>
/// Checks if user exsists on AD
/// </summary>
/// <param name="sUserName">The username to check</param>
/// <returns>Returns true if username Exists</returns>
public bool IsUserExisiting(string sUserName)
{
    if (GetUser(sUserName) == null)
    {
        return false;
    }
    else
    {
        return true;
    }
}

/// <summary>
/// Checks if user accoung is locked
/// </summary>
/// <param name="sUserName">The username to check</param>
/// <returns>Retruns true of Account is locked</returns>
public bool IsAccountLocked(string sUserName)
{
    UserPrincipal oUserPrincipal = GetUser(sUserName);
    return oUserPrincipal.IsAccountLockedOut();
}
#endregion

#region Search Methods

/// <summary>
/// Gets a certain user on Active Directory
/// </summary>
/// <param name="sUserName">The username to get</param>
/// <returns>Returns the UserPrincipal Object</returns>
public UserPrincipal GetUser(string sUserName)
{
    PrincipalContext oPrincipalContext = GetPrincipalContext();

    UserPrincipal oUserPrincipal =
```

```
UserPrincipal.FindByIdentity(oPrincipalContext, sUserName);
    return oUserPrincipal;
}

/// <summary>
/// Gets a certain group on Active Directory
/// </summary>
/// <param name="sGroupName">The group to get</param>
/// <returns>Returns the GroupPrincipal Object</returns>
public GroupPrincipal GetGroup(string sGroupName)
{
    PrincipalContext oPrincipalContext = GetPrincipalContext();

    GroupPrincipal oGroupPrincipal =
GroupPrincipal.FindByIdentity(oPrincipalContext, sGroupName);
    return oGroupPrincipal;
}

#endregion

#region User Account Methods

/// <summary>
/// Sets the user password
/// </summary>
/// <param name="sUserName">The username to set</param>
/// <param name="sNewPassword">The new password to use</param>
/// <param name="sMessage">Any output messages</param>
public void SetUserPassword(string sUserName, string sNewPassword, out
string sMessage)
{
    try
    {
        UserPrincipal oUserPrincipal = GetUser(sUserName);
        oUserPrincipal.SetPassword(sNewPassword);
        sMessage = "";
    }
    catch (Exception ex)
    {
        sMessage = ex.Message;
    }

}

/// <summary>
/// Enables a disabled user account
/// </summary>
/// <param name="sUserName">The username to enable</param>
public void EnableUserAccount(string sUserName)
{
    UserPrincipal oUserPrincipal = GetUser(sUserName);
```

```
    oUserPrincipal.Enabled = true;
    oUserPrincipal.Save();
}

/// <summary>
/// Force disbaling of a user account
/// </summary>
/// <param name="sUserName">The username to disable</param>
public void DisableUserAccount(string sUserName)
{
    UserPrincipal oUserPrincipal = GetUser(sUserName);
    oUserPrincipal.Enabled = false;
    oUserPrincipal.Save();
}

/// <summary>
/// Force expire password of a user
/// </summary>
/// <param name="sUserName">The username to expire the password</param>
public void ExpireUserPassword(string sUserName)
{
    UserPrincipal oUserPrincipal = GetUser(sUserName);
    oUserPrincipal.ExpirePasswordNow();
    oUserPrincipal.Save();

}

/// <summary>
/// Unlocks a locked user account
/// </summary>
/// <param name="sUserName">The username to unlock</param>
public void UnlockUserAccount(string sUserName)
{
    UserPrincipal oUserPrincipal = GetUser(sUserName);
    oUserPrincipal.UnlockAccount();
    oUserPrincipal.Save();
}

/// <summary>
/// Creates a new user on Active Directory
/// </summary>
/// <param name="sOU">The OU location you want to save your user</param>
/// <param name="sUserName">The username of the new user</param>
/// <param name="sPassword">The password of the new user</param>
/// <param name="sGivenName">The given name of the new user</param>
/// <param name="sSurname">The surname of the new user</param>
/// <returns>returns the UserPrincipal object</returns>
public UserPrincipal CreateNewUser(string sOU, string sUserName, string
sPassword, string sGivenName, string sSurname)
{
    if (!IsUserExisiting(sUserName))
```

```
    {
        PrincipalContext oPrincipalContext = GetPrincipalContext(sOU);

        UserPrincipal oUserPrincipal = new UserPrincipal(oPrincipalContext,
sUserName, sPassword, true /*Enabled or not*/);

        //User Log on Name
        oUserPrincipal.UserPrincipalName = sUserName;
        oUserPrincipal.GivenName = sGivenName;
        oUserPrincipal.Surname = sSurname;
        oUserPrincipal.Save();

        return oUserPrincipal;
    }
    else
    {
        return GetUser(sUserName);
    }
}

/// <summary>
/// Deletes a user in Active Directory
/// </summary>
/// <param name="sUserName">The username you want to delete</param>
/// <returns>Returns true if successfully deleted</returns>
public bool DeleteUser(string sUserName)
{
    try
    {
        UserPrincipal oUserPrincipal = GetUser(sUserName);

        oUserPrincipal.Delete();
        return true;
    }
    catch
    {
        return false;
    }
}

#endregion

#region Group Methods

/// <summary>
/// Creates a new group in Active Directory
/// </summary>
/// <param name="sOU">The OU location you want to save your new
Group</param>
/// <param name="sGroupName">The name of the new group</param>
/// <param name="sDescription">The description of the new group</param>
```

```
/// <param name="oGroupScope">The scope of the new group</param>
/// <param name="bSecurityGroup">True is you want this group to be a
security group, false if you want this as a distribution group</param>
/// <returns>Retruns the GroupPrincipal object</returns>
public GroupPrincipal CreateNewGroup(string sOU, string sGroupName, string
sDescription, GroupScope oGroupScope, bool bSecurityGroup)
{
    PrincipalContext oPrincipalContext = GetPrincipalContext(sOU);

    GroupPrincipal oGroupPrincipal = new GroupPrincipal(oPrincipalContext,
sGroupName);
    oGroupPrincipal.Description = sDescription;
    oGroupPrincipal.GroupScope = oGroupScope;
    oGroupPrincipal.IsSecurityGroup = bSecurityGroup;
    oGroupPrincipal.Save();

    return oGroupPrincipal;
}

/// <summary>
/// Adds the user for a given group
/// </summary>
/// <param name="sUserName">The user you want to add to a group</param>
/// <param name="sGroupName">The group you want the user to be added
in</param>
/// <returns>Returns true if successful</returns>
public bool AddUserToGroup(string sUserName, string sGroupName)
{
    try
    {
        UserPrincipal oUserPrincipal = GetUser(sUserName);
        GroupPrincipal oGroupPrincipal = GetGroup(sGroupName);
        if (oUserPrincipal != null && oGroupPrincipal != null)
        {
            if (!IsUserGroupMember(sUserName, sGroupName))
            {
                oGroupPrincipal.Members.Add(oUserPrincipal);
                oGroupPrincipal.Save();
            }
        }
        return true;
    }
    catch
    {
        return false;
    }
}

/// <summary>
/// Removes user from a given group
/// </summary>
```

```
/// <param name="sUserName">The user you want to remove from a group</param>
/// <param name="sGroupName">The group you want the user to be removed
from</param>
/// <returns>Returns true if successful</returns>
public bool RemoveUserFromGroup(string sUserName, string sGroupName)
{
    try
    {
        UserPrincipal oUserPrincipal = GetUser(sUserName);
        GroupPrincipal oGroupPrincipal = GetGroup(sGroupName);
        if (oUserPrincipal != null && oGroupPrincipal != null)
        {
            if (IsUserGroupMember(sUserName, sGroupName))
            {
                oGroupPrincipal.Members.Remove(oUserPrincipal);
                oGroupPrincipal.Save();
            }
        }
        return true;
    }
    catch
    {
        return false;
    }
}

/// <summary>
/// Checks if user is a member of a given group
/// </summary>
/// <param name="sUserName">The user you want to validate</param>
/// <param name="sGroupName">The group you want to check the membership of
the user</param>
/// <returns>Returns true if user is a group member</returns>
public bool IsUserGroupMember(string sUserName, string sGroupName)
{
    UserPrincipal oUserPrincipal = GetUser(sUserName);
    GroupPrincipal oGroupPrincipal = GetGroup(sGroupName);

    if (oUserPrincipal != null && oGroupPrincipal != null)
    {
        return oGroupPrincipal.Members.Contains(oUserPrincipal);
    }
    else
    {
        return false;
    }
}

/// <summary>
/// Gets a list of the users group memberships
/// </summary>
```

```
/// <param name="sUserName">The user you want to get the group
memberships</param>
/// <returns>Returns an arraylist of group memberships</returns>
public ArrayList GetUserGroups(string sUserName)
{
    ArrayList myItems = new ArrayList();
    UserPrincipal oUserPrincipal = GetUser(sUserName);

    PrincipalSearchResult<Principal> oPrincipalSearchResult =
oUserPrincipal.GetGroups();

    foreach (Principal oResult in oPrincipalSearchResult)
    {
        myItems.Add(oResult.Name);
    }
    return myItems;
}

/// <summary>
/// Gets a list of the users authorization groups
/// </summary>
/// <param name="sUserName">The user you want to get authorization
groups</param>
/// <returns>Returns an arraylist of group authorization
memberships</returns>
public ArrayList GetUserAuthorizationGroups(string sUserName)
{
    ArrayList myItems = new ArrayList();
    UserPrincipal oUserPrincipal = GetUser(sUserName);

    PrincipalSearchResult<Principal> oPrincipalSearchResult =
oUserPrincipal.GetAuthorizationGroups();

    foreach (Principal oResult in oPrincipalSearchResult)
    {
        myItems.Add(oResult.Name);
    }
    return myItems;
}

#endregion

#region Helper Methods

/// <summary>
/// Gets the base principal context
/// </summary>
/// <returns>Retruns the PrincipalContext object</returns>
public PrincipalContext GetPrincipalContext()
{
    PrincipalContext oPrincipalContext = new
```

```
PrincipalContext(ContextType.Domain, sDomain, sDefaultOU,
ContextOptions.SimpleBind, sServiceUser, sServicePassword);
    return oPrincipalContext;
}

/// <summary>
/// Gets the principal context on specified OU
/// </summary>
/// <param name="sOU">The OU you want your Principal Context to run
on</param>
/// <returns>Retruns the PrincipalContext object</returns>
public PrincipalContext GetPrincipalContext(string sOU)
{
    PrincipalContext oPrincipalContext = new
PrincipalContext(ContextType.Domain, sDomain, sOU,
ContextOptions.SimpleBind, sServiceUser, sServicePassword);
    return oPrincipalContext;
}

#endregion

}
```

Now this is how to use it.

```
ADMethodsAccountManagement ADMethods = new ADMethodsAccountManagement();

UserPrincipal myUser = ADMethods.GetUser("Test");
myUser.GivenName = "Given Name";
myUser.Surname = "Surname";
myUser.MiddleName = "Middle Name";
myUser.EmailAddress = "Email Address";
myUser.EmployeeId = "Employee ID";
myUser.Save();
```

From:
https://wiki.plecko.hr/ - **Eureka Moment**

Permanent link:
https://wiki.plecko.hr/doku.php?id=windows:ad:ad.net

Last update: **2019/10/31 09:05**