

Sharepoint : Création d'un filigrane personnalisé

par Ludovic Lefort ([Site web](#)) ([Blog](#))

Dernière mise à jour :

Comment créer un filigrane automatique lors de la création d'un document Word dans Sharepoint 2007 (uniquement MOSS 2007).

0 - Introduction.....	3
1 - Contenu de la solution.....	3
2 - Création de la solution.....	3
2.1 - L'event handler.....	3
2.2 - La policy.....	5
3 - Déploiement.....	8
4 - Activation de la policy.....	8
5 - Référence.....	9
6 - Liens.....	9
7 - Sources du projet.....	9

0 - Introduction

Pour commencer : qu'est ce qu'un filigrane (ou Watermark en anglais) ?

Il s'agit d'un texte ou d'une image présente dans le fond de votre document Word pour signaler par exemple qu'un document est confidentiel.

Dans notre exemple nous allons créer un filigrane contenant la mention DVP et la date de création du fichier :



Cette fonctionnalité sera activée via une policy définie sur la document library.

1 - Contenu de la solution

le déploiement de ce projet contiendra trois fichiers :

- Une dll contenant la policy
- Une dll pour l'event handler déclenchée lors de l'ajout d'un element dans la document library
- Un fichier XML contenant la description de la policy

2 - Création de la solution

2.1 - L'event handler

Pour débiter créez un nouveau projet de type class library et nommez le **CustomPolicyEventHandler** il contiendra le code à exécuter lors de l'ajout d'un élément dans la liste.

Ce projet contiendra deux classes.

La première classe contient le code nécessaire pour ajouter le filigrane dans un document word, nous l'appellerons helper (vous trouverez ce fichier dans les sources de cet article) :

```
using System.Collections.Generic;  
using System.Text;  
using Microsoft.SharePoint;  
using System.Reflection;  
using System.Xml;  
using System.IO;
```

```
using System;
using System.IO.Packaging;

namespace InsideMOSS.Samples
{
    public static class Helper
    {
        public static string SaveToTemp(SPFile doc)
        {
            //-- create temp folder if not existing
            if (!Directory.Exists(@"C:\temp"))
                Directory.CreateDirectory(@"c:\temp");
            //-- get a random file number
            Random rnd = new Random();
            String random = rnd.Next().ToString();
            String fileName = @"c:\temp\" + random + ".docx";
            //-- create the temporary file with the contents of the submitted document
            FileStream file = new FileStream(fileName, FileMode.Create);
            byte[] buffer = doc.OpenBinary();
            file.Write(buffer, 0, (int)doc.TotalLength);
            file.Flush();
            file.Close();

            return fileName;
        }

        public static void GetFromTemp(string filename, SPFile doc)
        {
            //-- load the temporary file back into the document in SharePoint
            FileStream file = File.OpenRead(filename);
            doc.SaveBinary(file, false);
            file.Close();
        }

        internal static void RemoveFromTemp(string filename)
        {
            if (File.Exists(filename))
                File.Delete(filename);
        }

        public static void AddWaterMark(string doc, string watermark)
        {
            string NS_WORDML = "http://schemas.openxmlformats.org/wordprocessingml/2006/3/main";
            string NS_REL = "http://schemas.openxmlformats.org/officeDocument/2006/relationships";
        }
    }
}
```

N'oubliez pas d'ajouter ces références à votre projet :

- Microsoft.Sharepoint
- Microsoft.Office.Policy
- WindowsBase

Créons à présent une deuxième classe contenant le code de l'event handler. Je l'ai appelée **DatePolicyEventHandler**

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Diagnostics;
using Microsoft.SharePoint;
using Microsoft.Office.RecordsManagement.InformationPolicy;
using InsideMOSS.Samples;

namespace CustomPolicyEventHandler
```

```

{
    public class DatePolicyEventHandler: SPItemEventReceiver
    {
        public override void ItemAdded(SPItemEventProperties properties)
        {
            Policy policy = Policy.GetPolicy(properties.ListItem.ContentType);
            PolicyItem policyItem = policy.Items[CustomWatermark.DatePolicy.PolicyID];

            string tempfile = Helper.SaveToTemp(properties.ListItem.File);
            Helper.AddWaterMark(tempfile, policyItem.CustomData);
            Helper.GetFromTemp(tempfile, properties.ListItem.File);
            Helper.RemoveFromTemp(tempfile);
            base.ItemAdded(properties);
        }
    }
}
    
```

Cette classe hérite de **SPItemEventReceiver** et override la méthode **ItemAdded**. Cela signifie que le code sera exécuté à chaque fois qu'un élément est ajouté à notre liste.

L'argument **properties** contient entre autres le fichier ajouté à notre document library.

Cette méthode récupère le fichier et lui ajoute le watermark.

Autre étape importante : signez votre assembly afin de pouvoir la déployer dans la GAC.

- Clic droit sur le projet
- properties
- Signing
- Choisir new dans la liste déroulante
- Entrez le nom **CustomPolicyEventHandler.snk**
- décochez **Protect my key with a password** et cliquez sur OK

2.2 - La policy

Nouvelle étape : Créez un nouveau projet de type class library (**CustomWatermark**) et ajoutez le en référence au projet précédent.

Ce projet va s'occuper d'exécuter du code lors de l'activation/désactivation de la policy. Il va aussi contenir un fichier XML décrivant la policy.

Première étape : ajoutez la classe **DatePolicy** dont voici le code :

```

using System;
using System.Collections.Generic;
using System.Text;
using Microsoft.Office.RecordsManagement.Internal;
using Microsoft.Office.RecordsManagement.InformationPolicy;
using Microsoft.SharePoint;

namespace CustomWatermark
{
    public class DatePolicy : IPolicyFeature
    {
        public static string PolicyID = "Developpez.Policy.DateWatermark";

        #region IPolicyFeature Members

        public void OnCustomDataChange(PolicyItem policyItem, SPContentType ct)
        {
        }

        public void OnGlobalCustomDataChange(PolicyFeature feature)
        
```

```

    {
    }

    public bool ProcessListItem(SPSite site, PolicyItem policyItem, SPListItem listItem)
    {
        return true;
    }

    public bool ProcessListItemOnRemove(SPSite site, SPListItem listItem)
    {
        return true;
    }

    public void Register(SPContentType ct)
    {
        try
        {
            SPEventReceiverDefinition evdef = ct.ParentList.EventReceivers.Add();

evdef.Assembly = "CustomPolicyEventHandler, Version=1.0.0.0, Culture=neutral, PublicKeyToken=0a5599bf8a9dala4";
evdef.Class = "CustomPolicyEventHandler.DatePolicyEventHandler";
evdef.Name = "Date Watermark";
evdef.Data = "DVP";
evdef.Type = SPEventReceiverType.ItemAdded;
evdef.SequenceNumber = 1001;
evdef.Update();
        }
        catch (Exception ex)
        {
            throw new Exception("Impossible d'enregistrer cet event handler", ex);
        }
    }

    public void UnRegister(SPContentType ct)
    {
        if (ct != null)
        {
            foreach (SPEventReceiverDefinition evdef in ct.ParentList.EventReceivers)
            {
                if (evdef.Name == "Date Watermark")
                    evdef.Delete();
            }
        }
    }

    #endregion
}
}

```

Note classe implémente l'interface IPolicyFeature, cela implique que nous devons implémenter certaines méthodes :

Méthode	Description
OnCustomDataChange()	Il est possible de développer des contrôles permettant aux administrateurs de modifier certain paramètres de la policy. Cette méthode permet de les sauver.
OnGlobalCustomDataChange()	Idem que la précédente mais au niveau de la ferme ou des Global Settings.
ProcessListItem()	Utile si vous avez déjà des éléments de votre document library et que vous souhaitez leurs appliquer la nouvelle policy.
ProcessListItemOnRemove()	Idem que la précédente mais lors de la désactivation de la policy.
Regsiter()	Ce code est appelé lorsqu'un administrateur assigne la policy à une liste
UnRegister()	Permet d'exécuter du code lors de la désactivation d'une policy.

Dans notre cas seules deux méthodes nous intéressent : **Register** et **Unregister**

```

public void Register(SPContentType ct)
{
    try
    {
        SPEventReceiverDefinition evdef = ct.ParentList.EventReceivers.Add();

        evdef.Assembly = "CustomPolicyEventHandler, Version=1.0.0.0, Culture=neutral, PublicKeyToken=0a5599bf8a9dala4";
        evdef.Class = "CustomPolicyEventHandler.DatePolicyEventHandler";
        evdef.Name = "Date Watermark";
        evdef.Data = "DVP - " + DateTime.Now.ToString("dd/MM/yyyy");
        evdef.Type = SPEventReceiverType.ItemAdded;
        evdef.SequenceNumber = 1001;
        evdef.Update();
    }
    catch (Exception ex)
    {
        throw new Exception("Impossible d'enregistrer cet event handler", ex);
    }
}
    
```

Lors de l'activation de la policy nous devons ajouter un nouvel event handler à notre document library (celui créé dans le projet précédent).

Pour récupérer le strong name de votre dll je vous conseille d'utiliser **.net reflector**

Voyons maintenant la méthode **Unregister**

```

public void UnRegister(SPContentType ct)
{
    if (ct != null)
    {
        foreach (SPEventReceiverDefinition evdef in ct.ParentList.EventReceivers)
        {
            if (evdef.Name == "Date Watermark")
                evdef.Delete();
        }
    }
}
    
```

Rien de bien compliqué : cette méthode supprime les event handler ajoutés à notre document library. Elle est déclenchée par la désactivation de la policy.

Il nous reste alors à créer notre manifest XML décrivant la policy et permettant son installation :

```
Date Watermark Policy
Policy qui ajoute un filigrane contenant la date de création du fichier
Developpez
CustomWatermark, Version=1.0.0.0, Culture=neutral, PublicKeyToken=f4dfb7b463ebd890
CustomWatermark.DatePolicy
```

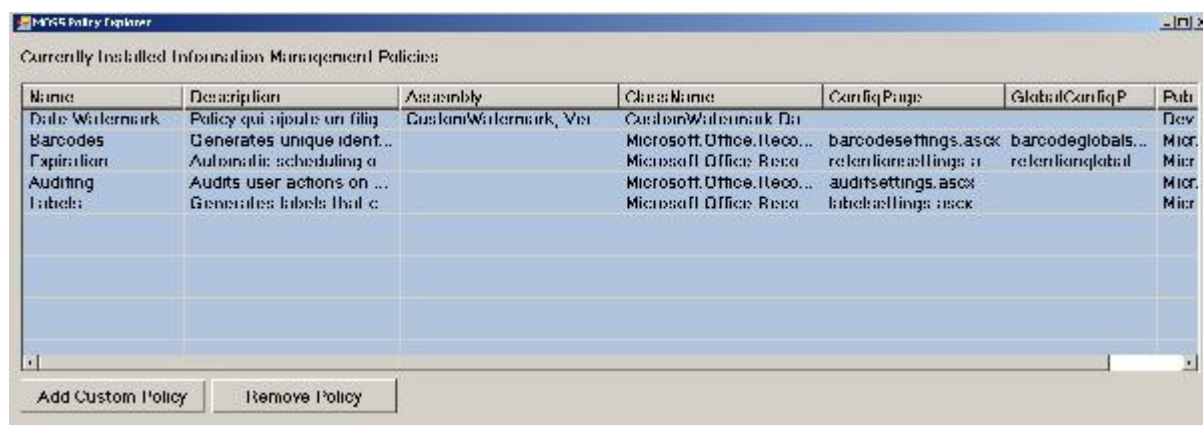
N'oubliez pas de signer ce projet également et compilez le tout.

3 - Déploiement

Le déploiement de la policy est relativement simple, la première étape consiste à installer les deux dll dans la GAC. Pour cela drag/dropper les deux fichiers dans le répertoire **c:\windows\assembly** de votre serveur.

Il nous reste à présent à installer la policy sur le serveur. Pour cela j'utilise le programme **Policy Explorer** téléchargeable **ici**.

Exécutez le programme sur votre serveur, cliquez sur le bouton **Add Custom Policy** et choisissez votre fichier **manifest.xml**.



Voilà votre policy est installée.

4 - Activation de la policy

Allez maintenant dans votre site Sharepoint et créez une nouvelle document library. Ouvrez ensuite les **document list settings**.

Dans la rubrique **Permissions and Management** cliquez sur **Information management policy Settings**

Permissions and Management

- Delete this document library
- Save document library as template
- Permissions for this document library
- Manage checked out files
- Workflow settings
- Information management policy settings

Sélectionnez le deuxième élément de la liste **Define a policy...** et cliquez sur **OK**

Specify the policy:

None

Define a policy...

Use a site collection policy:

Date Policy

Description:
None

OK Cancel

Votre policy apparait dans la liste, cochez la et cliquez sur **OK**

Date Watermark Policy

Policy qui ajoute un filigrane contenant la date de création du fichier

Enable Date Watermark Policy

OK Cancel

Voilà vous pouvez à présent créer un nouveau document dans votre liste et le sauver. Le filigrane sera ajouté automatiquement.

5 - Référence

Livre : Inside Microsoft Office Sharepoint Server 2007 de Patrick Tisseghem.

6 - Liens

Mon site : <http://lefortludovic.developpez.com>

Mon blog : <http://blogs.ezos.com/blog/le>

7 - Sources du projet

Cliquez [ici](#) pour télécharger les sources de ce projet.

Je tiens à remercier **RomainVALERI** pour la correction orthographique de cet article.

