SharePoint et PowerShell

Utilisation de PowerShell pour un script d'admin

PowerShell est le nouveau moteur de script de Microsoft. Il permet tout comme VBScript de créer des scripts afin d'automatiser certaines taches qui ne nécessitent pas de réel développement. Ce principe est utilisé constamment par les administrateurs système, mais il peut être aussi très utile pour les développeurs dans le cas de taches simples de gestion.

Introduction

PowerShell est développé par Microsoft pour ajouter un véritable outil de script aux environnements Windows. Avant cela, la seule solution possible était le passage par des scripts VBS.

Cette solution permet aussi aux administrateurs Unix de pouvoir écrire des scripts de gestion en restant dans un univers assez connu.

Présentation

Une des forces de PowerShell est qu'il est développé en .NET et qu'il peut donc charger à la volée des librairies .NET installées sur le serveur. On peut donc alors accéder aux objets de ces composants .NET.

C'est ainsi exactement le cas de SharePoint qui devient un composant instanciable depuis les scripts PowerShell comme on le fait classiquement en C#.

Nous verrons donc avec cet article comment installer PowerShell sur notre ferme (un des serveurs suffit pour utiliser PowerShell dans la ferme), puis nous verrons le cas d'un script basique reprenant la même idée que pour notre composant précédent :

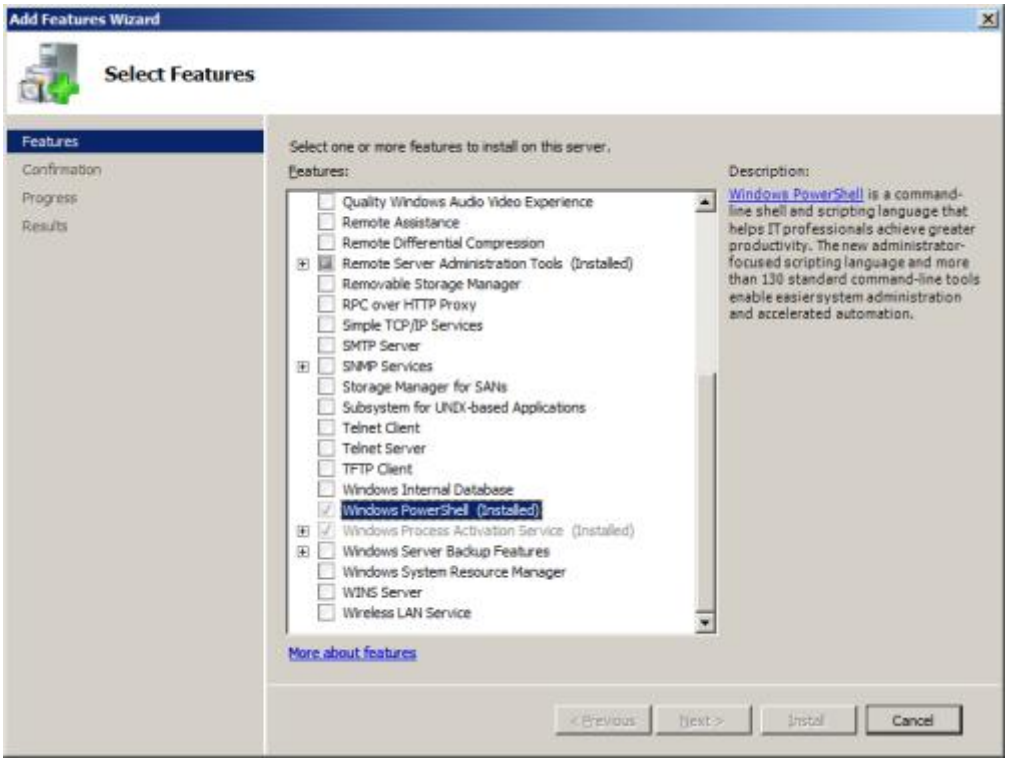
- **Créer une Feature d'administration sous SharePoint**

Nous créerons un script permettant d'appliquer le thème choisi à tous les sous-sites d'une collection de sites.

Installation de PowerShell

La méthode d'installation de PowerShell dépend de la version de Windows sur laquelle vous souhaitez l'utiliser.

En effet, Windows 2008 propose maintenant PowerShell comme composant activable depuis la gestion du serveur.

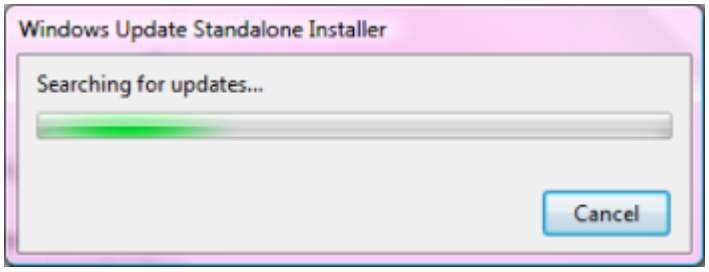


Vous n'avez donc rien à télécharger pour cet environnement.

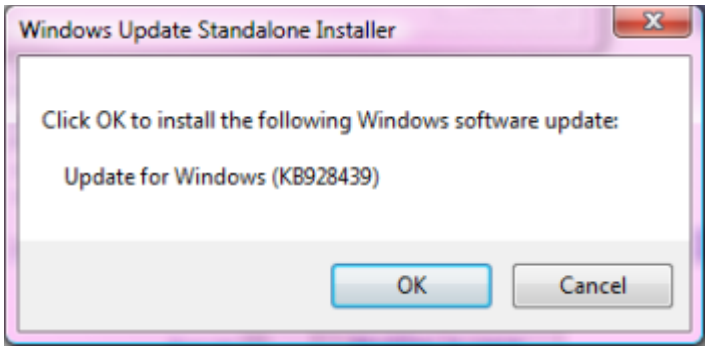
Pour les autres versions de Windows (2003, XP ou Vista), vous devez aller sur le site de **Microsoft** pour télécharger la version 1.0 en fonction de votre type de processeur (32 ou 64 Bits) :

- **How to Download Windows PowerShell 1.0**

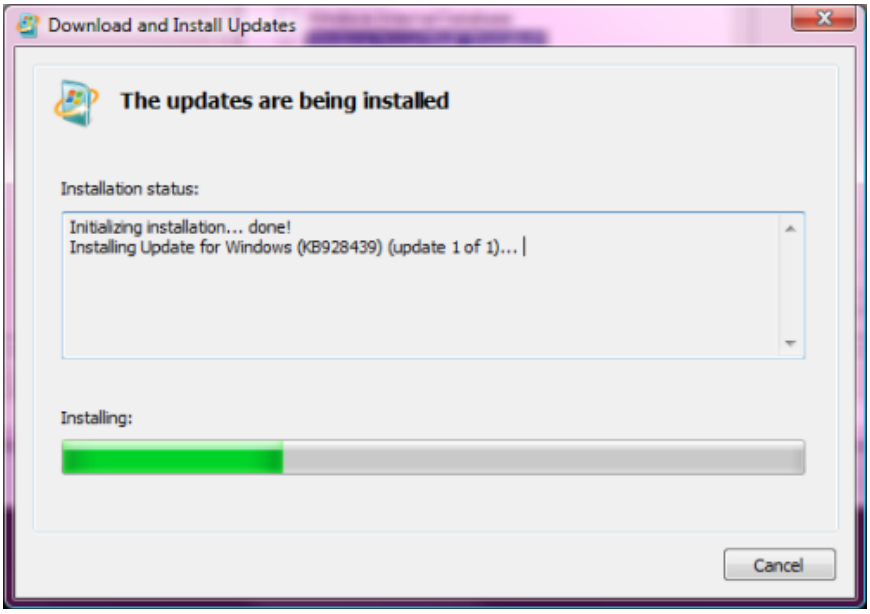
Nous installerons pour la présentation la version pour VISTA (il en est exactement de même pour celle de Windows 2003). Nous obtenons donc un fichier exécutable qui va dans un premier temps faire un check de votre machine



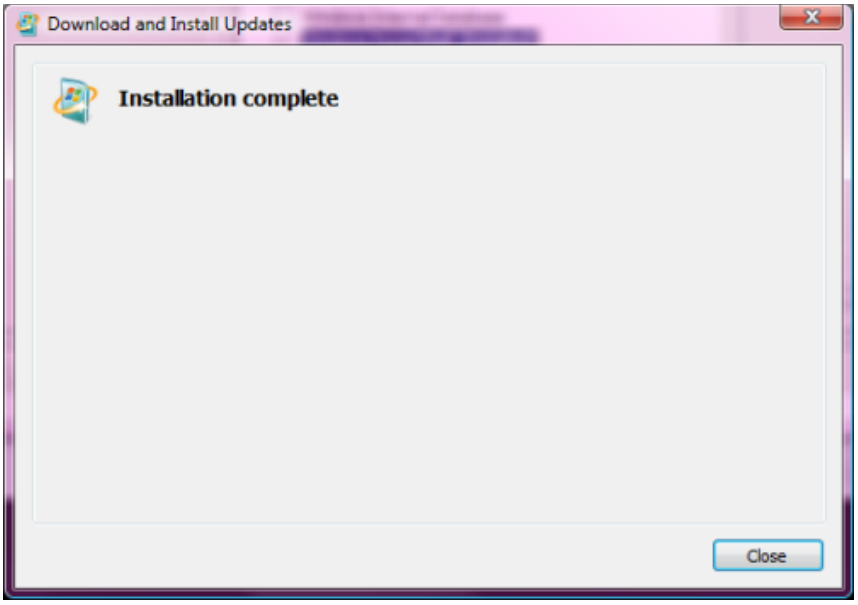
Il vous propose si besoin d'installer les patchs nécessaires



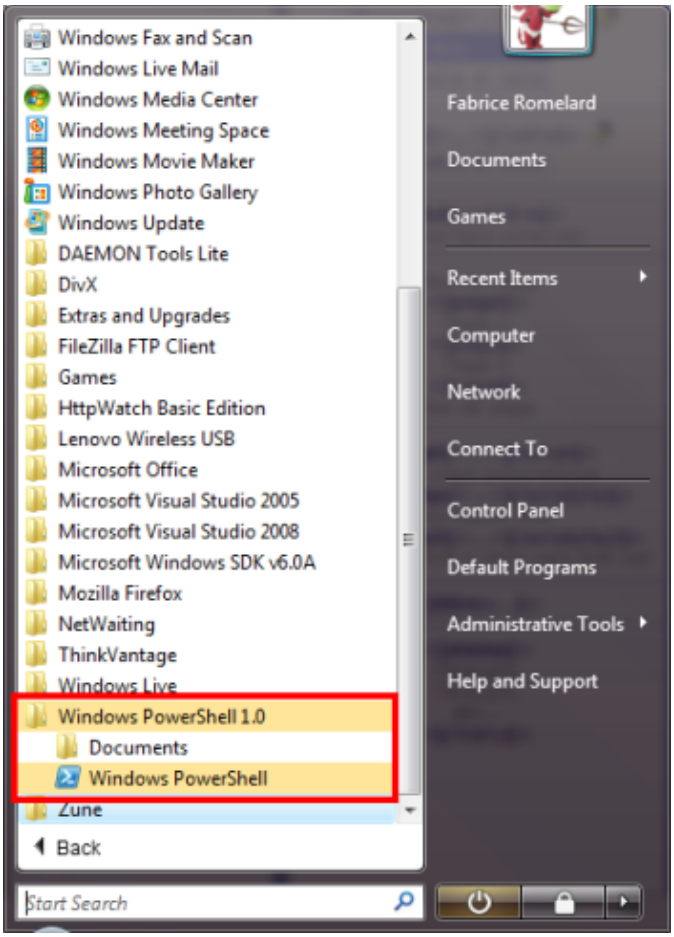
On accepte afin de démarrer l'installation des patchs et du produit



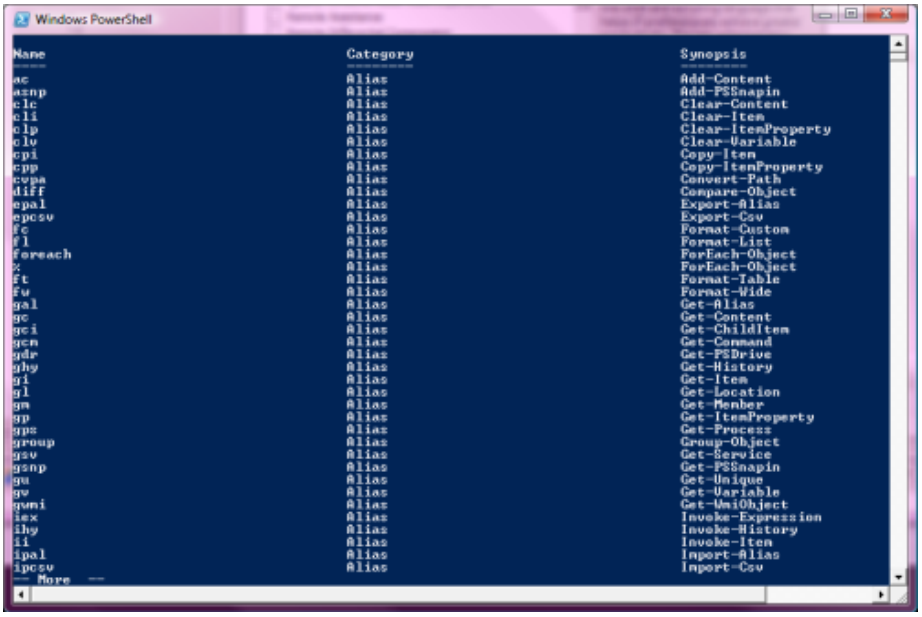
Après quelques minutes, l'installation se termine



On a donc maintenant un nouveau groupe qui s'est ajouté dans notre menu



Vous pouvez alors lancer l'outil "PowerShell" afin d'avoir sa console, sur laquelle on peut taper "help" pour avoir l'aide sur les commandes possibles



Nous avons donc maintenant PowerShell d'installé sur notre serveur (ou notre PC dans notre cas).

La suite de l'article sera effectuée depuis Windows Server 2008.

Exemple de script

Nous ne ferons pas dans cet article une documentation sur la façon de développer en PowerShell, de très bon site se chargent de cette partie (voir après la conclusion).

Nous prendrons plus spécifiquement un exemple de script simple reprenant le même objectif que notre feature d'administration :

- **Créer une Feature d'administration sous SharePoint**

Nous allons donc créer un script afin d'appliquer le Thème voulu sur une collection de sites définie en paramètre.

Pour créer des scripts PowerShell, Notepad suffit amplement, mais nous pouvons aussi utiliser **Notepad ++**, qui ne nécessite aucune installation sur le serveur. Nous pouvons choisir le type de langage "Shell" dans cet éditeur pour avoir une meilleure lecture.

Une des possibilités intéressantes de PowerShell est le fait de pouvoir créer des fonctions (comme tout langage un peu évoluer), mais le plus gros avantage de PowerShell est qu'il est basé sur le FrameWork 2.0 et peut charger les DLL .NET en dynamique.

On peut donc tout naturellement charger les DLL SharePoint au travers de la ligne :

```
[System.Reflection.Assembly]::LoadWithPartialName("Microsoft.SharePoint") > $null
```

On peut ensuite créer un objet comme on le ferait en C# :

```
$site = new-object Microsoft.SharePoint.SPSite($SiteCollectionURL)
$web = $site.openweb($WebURL)
```

On peut donc créer une fonction récursive (comme en C#) qui va recevoir un objet SPWeb et le nom du thème à appliquer

```
# Function:          Update-Theme
# Description:       Update the Theme
# Parameters:          web : SPWeb Object
#            TempThemeName : ThemeName
#
function Update-Theme([object]$web, [string]$TempThemeName)
{
  Write-Host "web", $web.Name
  Write-Host "Theme", $web.Theme

  $web.ApplyTheme($TempThemeName)
  $web.Update()

  foreach ($subweb in $web.GetSubwebsForCurrentUser())
  {
    Update-Theme $subweb $TempThemeName
   }
}
```

Cette fonction récursive sera appelée par une seconde fonction qui aura chargé en mémoire l'objet SPWeb du site racine pour la collection de sites choisie.

```
# Function:         Apply-Theme
# Description:      Update the Theme
# Parameters:         SiteCollectionURL : URL for Site Collection
#        ThemeName   Theme Name
function Apply-Theme([string]$SiteCollectionURL, [string]$ThemeName)
{
  [System.Reflection.Assembly]::LoadWithPartialName("Microsoft.SharePoint") > $null
  $site = new-object Microsoft.SharePoint.SPSite($SiteCollectionURL)
  $web = $site.openweb($SiteCollRelativeURL)

  #Debugging - show ThemeName
  Write-Host "ThemeName", $ThemeName
  #Debugging - show SiteCollectionURL
  Write-Host "SiteCollectionURL", $SiteCollectionURL

  Update-Theme $web $ThemeName
}
```

On peut alors simplement ajouter une dernière ligne qui va appeler cette seconde fonction avec les paramètres souhaités

```
Apply-Theme "http://win-urb7xo3pw9h/sites/ASP-PHP" "vintage"
```

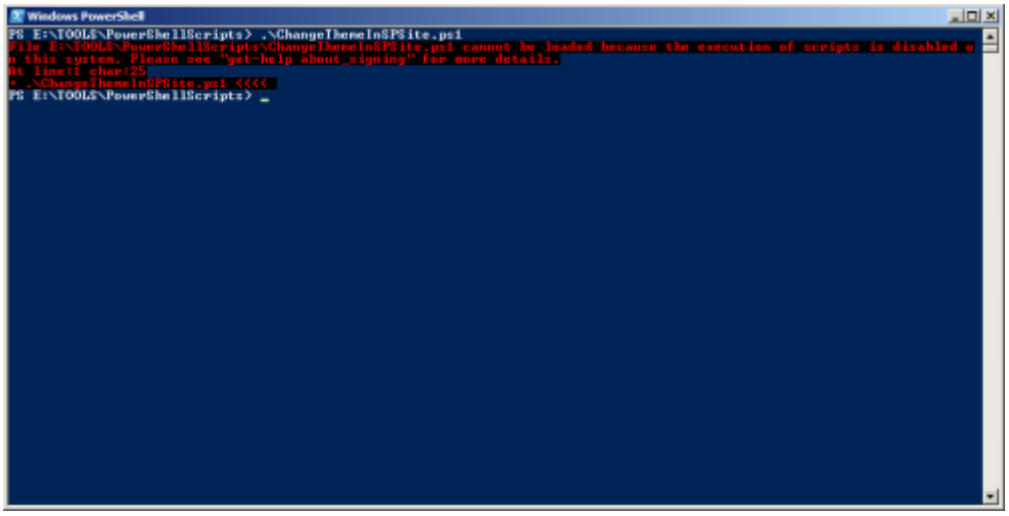
On peut maintenant utiliser notre console PowerShell, aller dans notre répertoire et exécuter le fichier (qui doit avoir l'extension PS1) :

```
CD FolderdeScripts
NomDuFichierPowerShell.PS1
```

ATTENTION :

Lors de l'exécution de tout script PowerShell, un accent est mis sur la sécurité et ce script ne peut s'exécuter si le fichier PS1 n'est pas signé. Si vous demandez l'exécution, vous obtiendrez un message :

- File E:\TOOLS\PowerShellScripts\ChangeThemeInSPSite.ps1 cannot be loaded because the execution of scripts is disabled o n this system. Please see "get-help about_signing" for more details. At line:1 char:25 + .\ChangeThemeInSPSite.ps1 <<<<



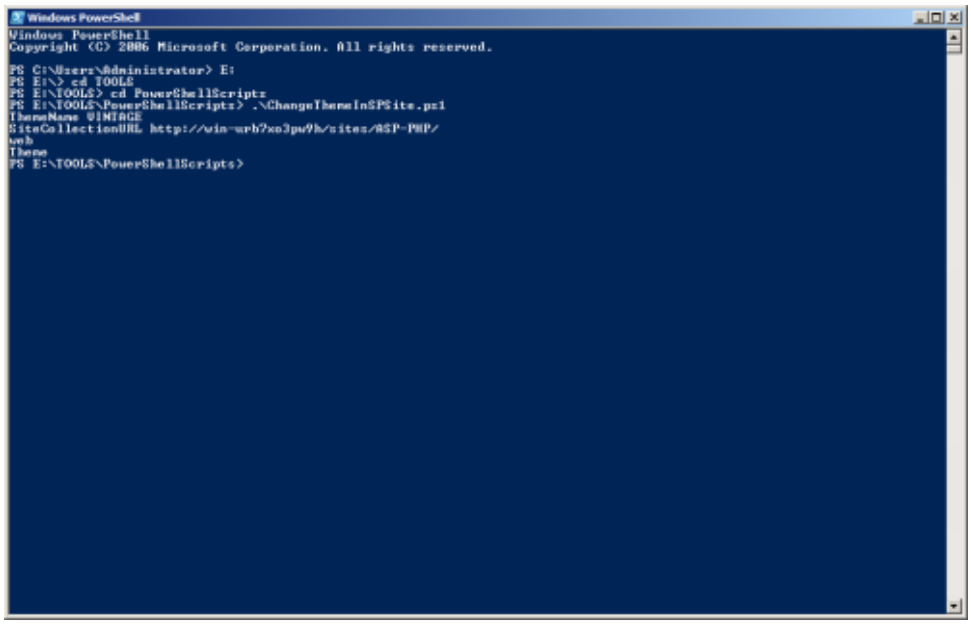
Vous devez donc lors de vos développements :

- Soit signer votre fichier PS1
- Soit désactiver la sécurité

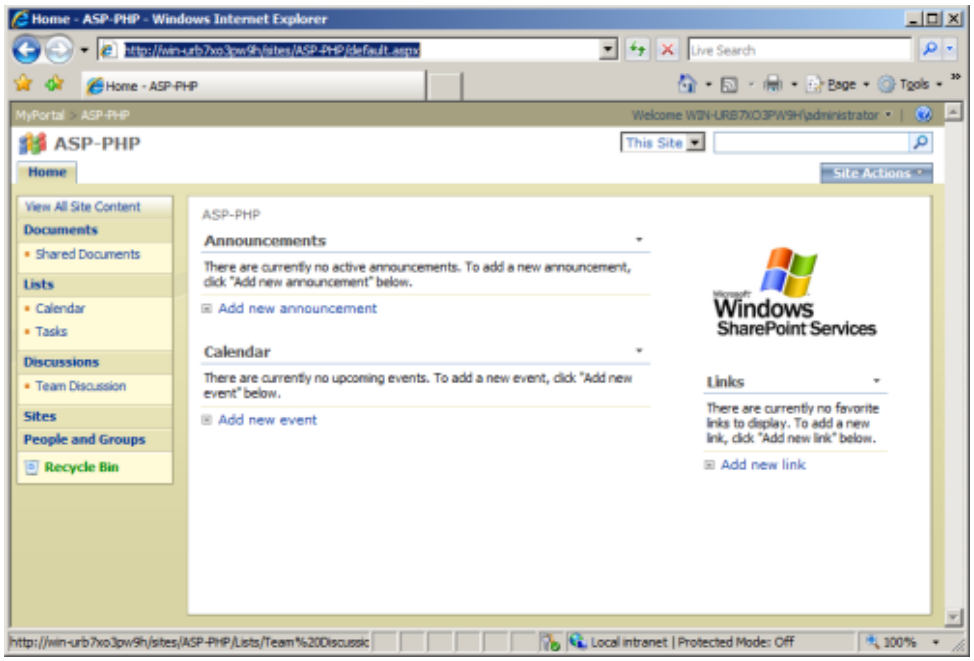
Je vous conseille de lire l'aide associée fournie par PowerShell "get-help about-Signing". Pour désactiver la sécurité (comme pour un serveur de développement), il vous suffit de lancer la commande :

- Set-ExecutionPolicy Unrestricted

On peut alors lancer le script directement



Et contrôler que la modification a bien été faite.



Voyons maintenant un exemple un peu plus complexe.

Exemple Avancé

Nous prendrons dans cet exemple le cas d'un fichier Master Page qui a été développé (par un designer) et que l'on souhaite appliquer à l'ensemble des sites existant dans l'ensemble des collections de sites de notre Web Application.

Nous pouvons donc utiliser le script suivant :

```
# Function:          Apply-MasterPageTheme
# Description:       Upload the New Master Page and apply it on all sites
# Parameters:          SiteCollectionURL   URL for Site Collection
#         SiteCollRelativeURL   Relative URL for Site Collection
#         FileMasterPage      Master Page to Apply
#         ThemeName      Theme Name
function Apply-MasterPageTheme([string]$SiteCollectionURL,
  [string]$SiteCollRelativeURL,
  [string]$FileMasterPage,
  [string]$ThemeName)
{
   [System.Reflection.Assembly]::LoadWithPartialName("Microsoft.SharePoint") > $null
   $site = new-object Microsoft.SharePoint.SPSite($SiteCollectionURL)
   $web = $site.openweb($SiteCollRelativeURL)
   $MasterURL = $SiteCollRelativeURL + "/_catalogs/masterpage/" + $FileMasterPage

#   Write-Host "FileMasterPage", $FileMasterPage
#   Write-Host "MasterURL", $MasterURL

   Update-MasterTheme $web $MasterURL $ThemeName

   }

# Function:          Update-MasterTheme
# Description:       Update the New Master Page and the Theme
# Parameters:          web              SPWeb Object
#         TempFileMasterPage      Master Page
#         TempThemeName         Theme Name
function Update-MasterTheme([object]$web,
  [string]$TempFileMasterPage,
  [string]$TempThemeName)
{
#   Write-Host "FileMasterPage", $TempFileMasterPage
#   Write-Host "Theme", $web.Theme

   $web.MasterUrl = $TempFileMasterPage
   $web.Update()

   $web.ApplyTheme($TempThemeName)
   $web.Update()

   foreach ($subweb in $web.GetSubwebsForCurrentUser())
   {
     Update-MasterTheme $subweb $TempFileMasterPage $TempThemeName
    }
}

# Function:          Upload-MasterPage
# Description:       Upload the New Master Page
# Parameters:          SiteCollectionURL   URL for Site COllection
#         RelativeWebURL      Relative Web Url
#         PathMasterPage      Path Master Page
#         MasterPageName      Master Page Name
function Upload-MasterPage([string]$SiteCollectionURL,
  [string]$RelativeWebURL,
  [string]$PathMasterPage,
  [string]$MasterPageName)
{
#   Write-Host "SiteCollectionURL", $SiteCollectionURL
#   Write-Host "RelativeWebURL", $RelativeWebURL
#   Write-Host "PathMasterPage", $PathMasterPage

   $propbag = @{"ContentType"="Master Page"}
   $docliburl = $SiteCollectionURL + "/_catalogs/masterpage"
   [System.Reflection.Assembly]::LoadWithPartialName("Microsoft.SharePoint") > $null
   $site = new-object Microsoft.SharePoint.SPSite($docliburl)
   $web = $site.openweb($RelativeWebURL)
   $folder = $web.getfolder($docliburl)
   $list = $web.lists[$folder.ContainingDocumentLibrary]
```

```
  $stream = [IO.File]::OpenRead($PathMasterPage)
#   Write-Host "MasterPageName", $MasterPageName

  $folder.files.Add($MasterPageName, $stream, $propbag, $true) > $null
  $stream.close()

}

# Function:          Update-All-SiteCollections
# Description:       Upload the New Master Page
# Parameters:          WebAppURL   URL for Web Application
#         LocalPathMasterPage      Local Path for Master Page
#         LocalMasterPageName      Local Master Page
#         LocalTheme            Local Theme
function Update-All-SiteCollections([string]$WebAppURL,
  [string]$LocalPathMasterPage,
  [string]$LocalMasterPageName,
  [string]$LocalTheme)
{

  [System.Reflection.Assembly]::LoadWithPartialName("Microsoft.SharePoint") > $null

  $Thesite = new-object Microsoft.SharePoint.SPSite($WebAppURL)
  $oApp = $Thesite.WebApplication

  foreach ($Sites in $oApp.Sites)
  {
    $mySubweb = $Sites.RootWeb
    $TempRelativeURL = $mySubweb.Url
    Write-Host "TempRelativeURL", $TempRelativeURL

    $RelativeURL = $TempRelativeURL -replace $WebAppURL, ""
    Write-Host "RelativeURL", $RelativeURL

    Upload-MasterPage $mySubweb.Url $RelativeURL $LocalPathMasterPage $LocalMasterPageName

    Apply-MasterPageTheme $mySubweb.Url $RelativeURL $LocalMasterPageName $LocalTheme
   }

}

Update-All-SiteCollections "http://win-urb7xo3pw9h" "C:\sample.master" "sample.master" "VINTAGE"
```

On peut alors exécuter le script directement (en adaptant les paramètres de la fonction de base).

Conclusion

Ce petit article vous a montré au travers de deux petits exemple la puissance de cet outil pour tous les besoins ou le développement de page ASPX ou de Winforms sont non réellement justifiés (tache d'administration, application d'une modification, ...).

Nous devons dès lors réfléchir lors des développement de pages techniques, car dans bien des cas, l'intégration de Code dans SharePoint (par les Feature) complexifie le code, alors qu'un simple script PowerShell fait le même travail.

Un facteur aussi très important est la rapidité d'exécution du script en question. EN effet, le fait de passer en direct évite tout le chargement de la couche graphique WEB, il permet donc d'accélérer considérablement l'exécution de la modification par rapport au développement éventuel en ASPX.

Voici quelques liens utiles si cet article vous a intéressé :

- **Windows Power Shell Quick Start**
- **Windows Power Shell - Blog MSDN**
- **Power Shell Scripting**
- **Introduction à PowerShell (nom de code Monad)**
- **Adventures in SPWonderland - PowerShell**
- **Defining multiple authentication providers to a single MOSS 2007 web application**
- **PowerShell for EveryOne**
- **Signing PowerShell Scripts**
- **Installation de WSS V3 (FR)**
- **Office Online**
- **Club SPS MOSS FRANCE(FR)**

En vous souhaitant de bons projets de développement.

Romelard Fabrice (alias F___)
Intranet/Extranet CTO - **SGS**