TP déploiement PowerShell (Simu ninite)

Table des matières

Pour modifier notre code pour qu'il ressemble au logiciel ninite il pourvoir que l'ors du lancement du script une page s'affiche pour faire notre choix.

I. Vérification / création de l'arborescence

```powershell
# Vérification et création de l'arborescence
$arborescence = @("C:\MyTools", "C:\MyTools\Scripts", "C:\MyTools\Apps", "C:\MyTools\Miscelanious")

foreach ($dossier in $arborescence) {
    if (-Not (Test-Path -Path $dossier)) {
        Write-Host "$dossier n'existe pas. Voulez-vous le créer? [O/N]"
        $reponse = Read-Host
        if ($reponse -eq 'O') {
            New-Item -ItemType Directory -Path $dossier
            Write-Host "$dossier a été créé."
        } else {
            Write-Host "Le script ne peut pas continuer sans créer les dossiers nécessaires."
            exit
        }
    }
}
```

Comme pour le script précédent on commence par vérifier que l'arborescence est bien créée et s'il ne les pas le faire. La seule différence c'est que la vérification est dans un fichier a par.

II. Charger les assemblages nécessaires pour WPF :

```powershell
# Charger les assemblages nécessaires pour WPF
Add-Type -AssemblyName PresentationFramework
```

Ici, nous chargeons l'assembly PresentationFramework qui est nécessaire pour créer des applications WPF.

III. Fonction TéléchargerEtInstallerApps

1) Création de la fenêtre d'utilisation

```
# Fonction pour télécharger et installer des Apps
function TéléchargerEtInstallerApps {
    [xml]$xaml = @"
<Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" Title="Sélectionner les Apps à Installer" Height="300" Width="400" Background="#f0f0f0">
    <Grid>
        <StackPanel VerticalAlignment="Center" HorizontalAlignment="Center">
            <CheckBox Name="chkSteam" Content="Steam" Margin="10" FontSize="16"/>
            <CheckBox Name="chkDiscord" Content="Discord" Margin="10" FontSize="16"/>
            <CheckBox Name="chkOpera" Content="Opera" Margin="10" FontSize="16"/>
            <CheckBox Name="chkVScode" Content="VScode" Margin="10" FontSize="16"/>
            <Button Name="btnInstall" Content="Installer" Width="150" Margin="20" HorizontalAlignment="Center" Background="#4CAF50" Foreground="White" FontSize="16"/>
        </StackPanel>
    </Grid>
</Window>
"@
```

Ici on commence par initier la fonction Téléchargement et installation apps.

Puis on indique que la variable $xaml contiendra du XML puis on ouvre une chaine de caractères sur plusieurs ligne grâce au @'' puis on crée la fenêtre avec la ligne window … puis on utilise la balise Grid qui est utiliser pour organiser les éléments de la fenêtre. Ensuite on crée un conteneur qui va contenir les enfants dans un alignement vertical et horizontal centrée. Puis on crée les enfants avec l'indicateur CheckBox qui indique une case à cocher puis on lui donne un nom de référence pour l'appeler plus loin ensuite le texte afficher à côté et pour finir des élément HTML, une fois ceci fait on ferme tout.

2) Création du lecteur XML et chargement de l'interface

```
$reader = (New-Object System.Xml.XmlNodeReader $xaml)
$form = [Windows.Markup.XamlReader]::Load($reader)
```

Avec c'est deux lignes on vient commencer par Crée un lecteur XML à partir de la chaîne de caractères $xaml. Cela permet de lire le contenu XML. Puis sur la deuxième ligne on Charge le XML en tant qu'interface utilisateur WPF. La méthode Load de XamlReader lit le contenu du lecteur XML et génère les objets WPF correspondants. Le résultat est stocké dans $form, qui représente la fenêtre WPF créée à partir du XAML.

3) Récupération des contrôles de la fenêtre

```
$chkSteam = $form.FindName("chkSteam")
$chkDiscord = $form.FindName("chkDiscord")
$chkOpera = $form.FindName("chkOpera")
$chkVScode = $form.FindName("chkVScode")
$btnInstall = $form.FindName("btnInstall")
```

Ici toutes les ligne on la même fonction qui et de récupérer les objet boutons crée si dessus les en implémentent dans la variable $form pour permettre que l'osque les case seront cocher elles auront un effet.

4) Ajout de l'événement de clic au bouton Installer

```
$btnInstall.Add_Click({
    $logFile = "C:\MyTools\Miscelanious\download_install.log"
    Start-Transcript -Path $logFile -Append

    $appsToInstall = @()
    if ($chkSteam.IsChecked) {
        $appsToInstall += "Steam"
    }
    if ($chkDiscord.IsChecked) {
        $appsToInstall += "Discord"
    }
    if ($chkOpera.IsChecked) {
        $appsToInstall += "Opera"
    }
    if ($chkVScode.IsChecked) {
        $appsToInstall += "VScode"
    }
```

Ici on commence par crée une variable $btnInstall et on ajoute. add_Click qui indique d'exécuter un code avec un clique. Puis on enregistre ce qu'il se passe dans un journal avec la variable $logFile et le chemin de ou les journaux seront enregistré. Puis avec Start-Transcript on démarre la journalisation des actions qui seront effectuer dans la variable $logFile et avec le -Append on indique que le script ajoutera des entrées à la fin du fichier existant au lieu de l'écraser.

Par la suite on crée la variable $appsToInstall et on va alors rentrer des conditions qui disent toute la même chose. Si l'élément « … » et cocher donner à la variable $appsToInstall le logiciel correspondant (ici sur les deuxième ligne le mot entre crocher ne fait pas référence à celui que l'on a écrié pour la fenêtre au début car on n'a pas mis de lien entre cette partie en HTML et le script).

5) Boucle de récupération, téléchargement et installation des apps

```
    foreach ($app in $appsToInstall) {
        switch ($app) {
            "Steam" {
                $url = "https://cdn.cloudflare.steamstatic.com/client/installer/SteamSetup.exe"
                $destination = "C:\MyTools\Apps\SteamSetup.exe"
                Invoke-WebRequest -Uri $url -OutFile $destination
                Start-Process -FilePath $destination -ArgumentList "/S" -NoNewWindow -Wait
            }
            "Discord" {
                $url = "https://discord.com/api/downloads/distributions/app/installers/latest?channel=stable&platform=win&arch=x64"
                $destination = "C:\MyTools\Apps\DiscordSetup.exe"
                Invoke-WebRequest -Uri $url -OutFile $destination
                Start-Process -FilePath $destination2 -ArgumentList "/S" -NoNewWindow -Wait
            }
            "Opera" {
                $url = "https://net.geo.opera.com/opera_gx/stable/windows?utm_tryagain=yes&utm_source=google&utm_medium=pa&utm_campaign=OGX_FR_Search_FR_T1_Brand_V2&utm_content=641854432695&utm_id=CjOKCQjw0_WyBhDMARIsAL1Vz8ujo9Ou_AeyUL7OEbSmMw6Sla4b
                $destination = "C:\MyTools\Apps\OperaSetup.exe"
                Invoke-WebRequest -Uri $url -OutFile $destination
                Start-Process -FilePath $destination -ArgumentList "/S" -NoNewWindow -Wait
            }
            "VScode" {
                $url = "https://update.code.visualstudio.com/1.89.1/win32-x64/stable"
                $destination = "C:\MyTools\Apps\VScodeSetup.exe"
                Invoke-WebRequest -Uri $url -OutFile $destination
                Start-Process -FilePath $destination -ArgumentList "/verysilent" -NoNewWindow -Wait
            }
        }
    }

    Stop-Transcript
    Write-Host "Téléchargement et installation des Apps terminé."
    $form.Close()
})

$form.ShowDialog() | Out-Null
```

Ici la boucle n'a pas changé sauf sur deux points il y a 3 apps en plus, j'ai retiré la partie pour prendre en compte les .msi car il y en pas dans le script ce qui fessait des lignes pour rien et il n'y a plus la partie pour le menu car cela est dans une autre partie maintenant.

IV. Fonction de récupération des infos du pc

```
# Fonction pour récupérer les informations du PC
function RécupérerInfosPC {
    [xml]$xaml = @"
<Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" Title="Récupérer les Informations du PC" Height="200" Width="400" Background="#f0f0f0">
    <Grid>
        <StackPanel VerticalAlignment="Center" HorizontalAlignment="Center">
            <TextBlock Text="Entrez le nom du PC ou Server:" Margin="10" FontSize="16"/>
            <TextBox Name="txtComputerName" Width="200" Margin="10" FontSize="16"/>
            <Button Name="btnFetchInfo" Content="Récupérer" Width="150" Margin="20" HorizontalAlignment="Center" Background="#2196F3" Foreground="White" FontSize="16"/>
        </StackPanel>
    </Grid>
</Window>
"@

    $reader = (New-Object System.Xml.XmlNodeReader $xaml)
    $form = [Windows.Markup.XamlReader]::Load($reader)

    $txtComputerName = $form.FindName("txtComputerName")
    $btnFetchInfo = $form.FindName("btnFetchInfo")

    $btnFetchInfo.Add_Click({
        $computerName = $txtComputerName.Text
        $logFile = "C:\MyTools\Miscelanious\Remote-$computerName.log"

        $infos = @()
        $infos += "Nom du PC/Server: $computerName"
        $infos += "Adresse IP: " + (Test-Connection -ComputerName $computerName -Count 1).IPv4Address.IPAddressToString
        $infos += "Machine Physique ou VM: " + (Get-WmiObject -Class Win32_ComputerSystem -ComputerName $computerName).Model
        $infos += "Mémoire: " + (Get-WmiObject -Class Win32_ComputerSystem -ComputerName $computerName).TotalPhysicalMemory
        $infos += "Espace disque: " + (Get-WmiObject -Class Win32_LogicalDisk -Filter "DriveType=3" -ComputerName $computerName | Measure-Object -Property Size -Sum).Sum
        $infos += "Version de l'OS: " + (Get-WmiObject -Class Win32_OperatingSystem -ComputerName $computerName).Version

        $infos | Out-File -FilePath $logFile
        Write-Host "Les informations ont été récupérées et enregistrées dans $logFile."
        $form.Close()
    })

    $form.ShowDialog() | Out-Null
}
```

Ici on vient faire une fenêtre de récupération des infos du pc en utilisant la même méthode utiliser avant puis on utilise les mêmes ligne (les noms change pour correspondre) qu'avant pour faire marcher la fenêtre et la partie qui récupère les infos sur le pc est la même que le précédant script.

V. Fenêtre de choix

```
# Interface graphique principale
[xml]$mainXaml = @"
<Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" Title="Menu Principal" Height="250" Width="350" Background="#f0f0f0">
    <Grid>
        <StackPanel VerticalAlignment="Center" HorizontalAlignment="Center">
            <TextBlock Text="Menu Principal" FontSize="20" FontWeight="Bold" Margin="10" HorizontalAlignment="Center"/>
            <Button Name="btnApps" Content="Télécharger et installer des Apps" Width="250" Margin="10" Background="#FF9800" Foreground="White" FontSize="16"/>
            <Button Name="btnInfo" Content="Récupérer les informations du PC" Width="250" Margin="10" Background="#FF9800" Foreground="White" FontSize="16"/>
        </StackPanel>
    </Grid>
</Window>
"@

$mainReader = (New-Object System.Xml.XmlNodeReader $mainXaml)
$mainForm = [Windows.Markup.XamlReader]::Load($mainReader)

$btnApps = $mainForm.FindName("btnApps")
$btnInfo = $mainForm.FindName("btnInfo")

$btnApps.Add_Click({
    $mainForm.Close()
    TéléchargerEtInstallerApps
})

$btnInfo.Add_Click({
    $mainForm.Close()
    RécupérerInfosPC
})

$mainForm.ShowDialog() | Out-Null
```

Ici on vient crée la fenêtre principale qui va nous permettre de choisir entre la fonction de téléchargement et la fonction de Récupération. Pour ce faire un doit crée une page comme les deux fonctions mais avec juste deux bouton. Puis on fait comme les deux fois précédente pour que les boutons ont un effet a la seule différence est que lors qu'on appuyé sur un bouton cela ferme la page et lance la fonction sélectionner.