TP Powershell supplémentaire

I SOUS SEVEN EXO 1

Créer un script Powershell qui, à son lancement, affiche les lignes suivantes :

Quel est le nom d'utilisateur ? *Saisir_le_nom_de_l'utilisateur ?*
Vérifier que ce compte existe bien dans la base SAM de votre poste Windows 7.
Puis afficher si l'utilisateur existe afficher

Bienvenue *nom_de_l'utilisateur*
Votre dernière connexion a eu lieu à : *date et heure de la dernière connexion*

L'intellisense vous proposera les propriétés du compte, à vous de choisir les bonnes.

Si le compte n'existe pas dans la base SAM, afficher un message d'erreur.

II SOUS SEVEN EXO 2

Écrire un script qui recherche si un fichier existe dans un répertoire.

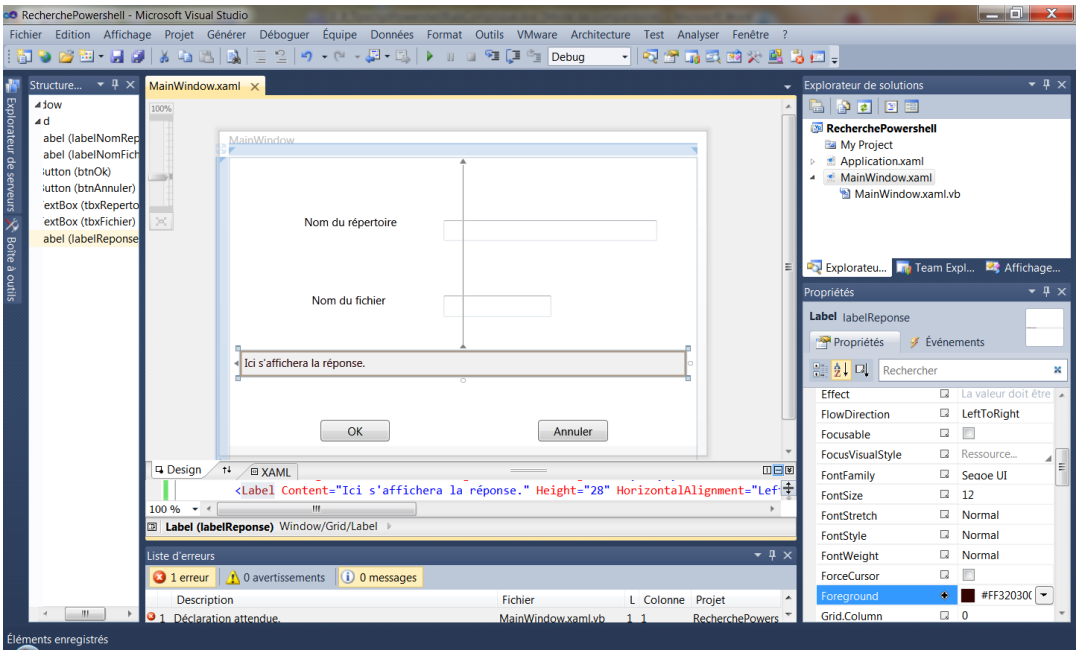
Le répertoire est saisi en premier, s'il existe alors le nom de fichier recherché est aussi saisi. Si le répertoire n'existe pas, le script affiche un message d'erreur.

Si le répertoire existe, le script recherche le nom de fichier dans ce répertoire.

Le script affiche ensuite un message indiquant si le fichier a été trouvé à l'endroit recherché ou non.

III SOUS SEVEN MÊME CHOSE QUE L'EXERCICE PRÉCÉDENT MAIS AVEC INTERFACE GRAPHIQUE EXO3

Lancer Visual Studio et créer un nouveau projet VB et WPF pour créer l'interface graphique ci-dessous.



Enregistrer votre projet et récupérer le contenu du fichier « .xaml » de votre projet pour le copier en tête de votre script Powershell après les lignes suivantes :

```
# chargement des classes du framework l'équivalent de l'import en vb
Add-Type -AssemblyName PresentationFramework
Add-Type -AssemblyName PresentationCore
Add-Type -AssemblyName WindowsBase

# on essaie une interface graphique
[xml] $XAML = @'
```

Supprimer les caractères `x:Class="MainWindow"` sur la première balise `<Windows`

Rajouter ensuite le code suivant pour lire ces balises XML et pouvoir afficher cette interface en Powershell :

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

Rajouter maintenant les lignes suivantes pour récupérer dans votre script Powershell les objets graphiques de votre interface :

```
# récupérer les d'objets graphiques du formulaire (extraits)
$tbxRepertoire = $form.FindName('tbxRepertoire')
$btnOK = $form.FindName('btnOK')
```

Vous pouvez désormais dans votre script Powershell, utiliser les accesseurs de ces objets pour récupérer entre autre la propriété « Text » des TextBoxs.

Il reste à gérer les évènements sur les boutons de la manière suivante :

```
# on traite ici l'évènement clic sur le bouton Ok
$btnOK.add_click({
# c'est ici qu'il faut mettre le code de l'évènement clic
})

# on traite l'évènement clic sur le bouton Annuler
$btnAnnuler.add_click({$form.close()})
```

Et pour finir on affiche le formulaire :

```
# maintenant on affiche le formulaire, les évènements seront interceptés
et traités par le code précédent
$form.ShowDialog() | Out-Null
```

Pour aller plus loin :

- Afficher la réponse en rouge ou vert selon les cas.
- Faire la recherche dans tous les répertoires et sous-répertoires.