Get-Member, PSBase et PSObject

Contenu

- **Utilisation de Get-Member**
- **PSBase vs PSObject**

Utilisation de Get-Member

Utilisation de Get-Member

- **Le cmdlet Get-Member permet d'obtenir la liste des propriétés et méthodes d'un objet**
  - Syntaxe : objet | Get-Member

Exemple

```
PS /home/plbrault/temp/PowerShell> $date = Get-Date

PS /home/plbrault/temp/PowerShell> $date | Get-Member



   TypeName: System.DateTime

Name                 MemberType     Definition
----                 ----------     ----------
Add                  Method         datetime Add(timespan value)
AddDays              Method         datetime AddDays(double value)
AddHours             Method         datetime AddHours(double value)
AddMilliseconds      Method         datetime AddMilliseconds(double value)
AddMinutes           Method         datetime AddMinutes(double value)
AddMonths            Method         datetime AddMonths(int months)
AddSeconds           Method         datetime AddSeconds(double value)
AddTicks             Method         datetime AddTicks(long value)
AddYears             Method         datetime AddYears(int value)
CompareTo            Method         int CompareTo(System.Object value), int CompareTo(datetime value), int IComparable.CompareTo(System.Object obj)…
Equals               Method         bool Equals(System.Object value), bool Equals(datetime value), bool IEquatable[datetime].Equals(datetime other)
```

Exemple

```
Date                Property       datetime Date {get;}
Day                 Property       int Day {get;}
DayOfWeek           Property       System.DayOfWeek DayOfWeek {get;}
DayOfYear           Property       int DayOfYear {get;}
Hour                Property       int Hour {get;}
Kind                Property       System.DateTimeKind Kind {get;}
Millisecond         Property       int Millisecond {get;}
Minute              Property       int Minute {get;}
Month               Property       int Month {get;}
Second              Property       int Second {get;}
Ticks               Property       long Ticks {get;}
TimeOfDay           Property       timespan TimeOfDay {get;}
Year                Property       int Year {get;}
DateTime            ScriptProperty System.Object DateTime {get=if ((& { Set-StrictMode -Version 1; $this.DisplayHint }) -ieq  "Date")…
```

PSBase vs PSObject

Les objets utilisables en PowerShell

- **Comme vous le savez, on manipule souvent des objets .NET en PowerShell**
- **On peut aussi manipuler des objets provenant d'autres technologies, ex :**
  - COM (*Component Object Model*)
  - WMI (*Windows Management Instrumentation*)

Component Object Model (COM)

Selon Wikipédia :

« Component Object Model est une spécification créée par Microsoft, qui décrit comment un programme exécutable peut être emballé dans un objet par un programmeur, permettant l'utilisation de l'objet par d'autres programmeurs. »

Windows Management Instrumentation (WMI)

Selon Wikipédia :

« WMI est un système de gestion interne de Windows qui prend en charge la surveillance et le contrôle de ressources systèmes via un ensemble d’interfaces. Il fournit un modèle cohérent et organisé logiquement des états de Windows. »

La représentation PSObject

- **Les objets sont représentés de manières différentes selon les technologies desquelles ils proviennent**
- **Pour nous aider, PowerShell nous fournit une représentation commune à tous les objets (avec des propriétés et des méthodes)**
- **Cette représentation s'appelle PSObject**

La représentation PSBase

- **Il arrive que la représentation PSObject « cache » certaines caractéristiques d'un objet**
- **La représentation PSBase permet alors d'accéder aux caractéristiques de l'objet sous-jacent (provenant directement de .NET, COM, WMI, etc)**

Exemple

- **Cette ligne de code permet de déclarer un objet contenant des données au format XML :**
  - `$xml = [xml]"<root><a /></root>"`

Exemple

- **Avec `$xml.psobject`, on peut alors obtenir les caractéristiques de l'objet selon sa représentation PSObject**
  - C'est-à-dire la liste de ses propriétés et méthodes

Exemple

```
PS /home/plbrault/temp/PowerShell> $xml.psobject


BaseObject          : #document
Members             : {static string XmlNode(psobject instance), System.Xml.XmlElement root {get;}, add_NodeInserting,
                      remove_NodeInserting…}
Properties          : {System.Xml.XmlElement root {get;}, System.Xml.XmlNodeType NodeType {get;}, System.Xml.XmlNode ParentNode
                      {get;}, System.Xml.XmlDocumentType DocumentType {get;}…}
Methods             : {static string XmlNode(psobject instance), add_NodeInserting, remove_NodeInserting, add_NodeInserted…}
ImmediateBaseObject : #document
TypeNames           : {System.Xml.XmlDocument, System.Xml.XmlNode, System.Object}
```

Exemple

- **Avec `$xml.psbase`, on obtient plutôt les caractéristiques de l'objet selon sa représentation PSBase**

```
PS /home/plbrault/temp/PowerShell> $xml.psbase


NodeType            : Document
ParentNode          :
DocumentType        :
Implementation      : System.Xml.XmlImplementation
Name                : #document
LocalName           : #document
DocumentElement     : root
OwnerDocument       :
Schemas             : System.Xml.Schema.XmlSchemaSet
XmlResolver         :
NameTable           : System.Xml.NameTable
PreserveWhitespace  : False
IsReadOnly          : False
InnerText           :
InnerXml            : <root><a /></root>
SchemaInfo          : System.Xml.Schema.XmlSchemaInfo
BaseURI             :
Value               :
ChildNodes          : {root}
PreviousSibling     :
NextSibling         :
Attributes          :
FirstChild          : root
LastChild           : root
```

Exemple

- **On peut ensuite accéder directement aux attributs de l'objet PSBase**

```
PS /home/plbrault/temp/PowerShell> $xml.psbase.InnerXML

<root><a /></root>
```

Conclusion

- **À retenir**
  - PSObject = Représentation commune à tous les objets (propriétés et méthodes)
  - PSBase = Représentation propre à la technologie de laquelle l'objet provient
  - PSBase permet parfois d'accéder à des caractéristiques cachées par PSObject
- **Nous verrons si ces notions s'avéreront utiles pour la suite du cours!**

Fin de la présentation

Des questions?

Photo par Emily Morter sur Unsplash