УДК 004.652.5

ПЕРСПЕКТИВИ СТВОРЕННЯ ОБ'ЄКТНОЇ БАЗИ ДАНИХ ЗАСОБАМИ ОБОЛОНКИ КОМАНДНОГО РЯДКА WINDOWS POWERSHELL

Горбань Гліб

Чорноморський національний університет імені Петра Могили

Анотація

У доповіді представлені основні принципи роботи у командній оболонці PowerShell для операційної системи Windows на прикладі виведення вмісту каталогу. Показано, що результат виконання команд є сукупністю об'єктів певних класів .NET. Представлені команди, що можуть здійснювати фільтрацію об'єктів, та проведена їх аналогія із SQL-запитами для реляційних баз даних. Показана можливість створення власних класів у PowerShell та побудові на їх основі об'єктної бази даних.

Abstract

The paper presents the basic principles of working in the PowerShell shell for the Windows operating system as an example of outputting the contents of a directory. It is shown that the result of executing commands is a set of objects of certain .NET-classes. The commands that can filter objects are presented, and their analogy with SQL queries for relational databases is carried out. The ability of own classes creation in PowerShell and building on their basis an object database is shown.

Командна оболонка PowerShell для операційної системи Windows є відносно новою. Перша її версія вийшла у 2006 році, а починаючи з Windows 7, PowerShell є невід'ємною частиною даної операційної системи.

Дана оболонка була покликана замінити більш старі засоби автоматизації завдань у ОС Windows: командної оболонки cmd.exe та сервера сценаріїв Windows Script Host (WSH) [1,2]. Кожен з даних засобів мав свої недоліки. Cmd.exe хоча і був найбільш універсальним та простим у вивченні засобом автоматизації, в тім за його допомогою можна було вирішити достатньо обмежене коло задач, оскільки дана командна оболонка не забезпечувала доступ до багатьох об'єктів, що підтримуються операційною системою. У свою чергу WSH представляв собою технологію написання сценаріїв, виконуваних у ОС Windows, на мовах програмування високого рівня, однак не був командною оболонкою. До того ж WSH вимагав великої роботи по вивченню даних мов (VBScript, Jscript) та суміжних технологій управління.

З появою PowerShell було вирішено багато проблем, що виникали при автоматизації рутинних задач у ОС Windows, зокрема проблема інтеграції командного рядка з об'єктами COM, WMI та .NET; робота з довільними джерелами даних в командному рядку за принципом файлової системи.

Головною особливістю нової командної оболонки стало те, що результатом виведення її команд є не текст, як було у командній оболонці Cmd.exe або у командних оболонках UNIX-подібних систем, а об'єкт. При цьому можна як вивести потрібні значення властивостей даних об'єктів, так і виконати їх методи. Оскільки сама оболонка PowerShell побудована на платформі .NET, будь який об'єкт, дані якого виводяться певною командною, є об'єктом .NET. До того ж внутрішня команда у PowerShell називається командлетом і теж представляє собою деякий клас .NET, який є нащадком класу `Cmdlet`, що є базовим класом для всіх внутрішніх команд. Більша кількість командлетів повертає певну інформацію, що виводиться на консоль та представляє сукупність об'єктів певного класу .NET.

Розглянемо, наприклад, командлет `Get-ChildItem`, що виводить вміст певного каталогу (якщо ім'я каталогу не задане, буде виведено вміст поточного робочого

каталогу). Даний командлет можна також викликати за псевдонімами `dir` та `ls` відповідно до назв команд оболонки cmd.exe та оболонок UNIX-подібних систем. Виконаємо його для виведення каталогу `C:\Windows`:

```
Get-ChildItem C:\Windows
```

В результаті буде виведена наступна інформація:

```
    Каталог: C:\Windows

Mode         LastWriteTime          Length Name
----         -------------          ------ ----
d-----       16.07.2016     11:30          addins
……….
-a----       16.07.2016     11:26     707 _default.pif
```

По суті кожний виведений рядок представляє властивості окремого об'єкту деякого класу .NET. Однак за замовчуванням виводяться не всі властивості. Для того, щоб отримати назву класу, що є типом об'єктів, а також всі члени даного класу (властивості та методи), потрібно використати командлет `Get-Member`:

```
Get-ChildItem | Get-Member
```

Результат виконання командлету:

```
   TypeName: System.IO.DirectoryInfo

Name         MemberType      Definition
----         ----------      ----------
LinkType     CodeProperty    System.String
LinkType{get=GetLinkType;}
……
Root         Property        System.IO.DirectoryInfo     Root
{get;}
BaseName     ScriptProperty  System.Object           BaseName
{get=$this.Name;}
```

Подібний механізм виклику в оболонках командного рядка називається конвеєром, який представляє собою ланцюг виконуваних команд, у якому результат виведення попередньої команди подається на вхід наступної. У конвеєрах часто використовуються команди, що здійснюються фільтрацію виведеної інфомації.

Наприклад, командлет `Where-Object` здійснює вибірку об'єктів за певною умовою. Для прикладу, виведемо інформацію тільки про виконувані файли (`.exe`):

```
Get-ChildItem C:\Windows | Where-Object {$_.Extension -eq
".exe"}
```

Іншим прикладом роботи конвеєра є командлет `Select-Object`, що дозволяє вивести тільки вказані властивості об'єктів. Виведемо, наприклад, вміст каталогу, залишивши тільки ім'я підкаталогів або файлів та розмір файлів (для підкаталогів властивість розміру залишається порожньою).

```
Get-ChildItem C:\Windows | Select-Object Name,Length
```

До того ж існує командлет Group-Object, який виконує групування об'єктів за відповідною властивістю. Так, наприклад, можна дізнатись, скільки файлів, що мають відповідне розширення знаходиться у каталозі.

```
Get-ChildItem C:\Windows | Group-Object Extension
```

Результати роботи розглянутих вище командлетів нагадують результати запитів на мові SQL для реляційних баз даних. Дійсно, у сукупності об'єктів, що повертаються у якості результату виконання певних командлетів можна знайти аналогію з реляційною таблицею.

Однак в даному випадку дані все ж представлені у вигляді об'єктів класів. Все ж існує аналогія між табличними даними та даними, представленими у вигляді об'єктів. Так клас представляється таблицею, об'єкт – одним рядком таблиці, властивість – стовпчиком таблиці.

Об'єктно-орієнтована парадигма у свій час поклала початок розвитку об'єктної моделі даних, яка повністю побудована на класах [3,4]. Однією з основних відмінностей з реляційною моделлю даних є те, що зв'язки між об'єктами здійснюються не за полями (властивостями), а за посиланнями. У свою чергу посилання представляють такі самі властивості, тільки їх типами є не прості типи даних, а класи. Іншою відмінністю є те, що зв'язки між об'єктами є двонаправленими.

PowerShell дозволяє описувати свої власні класи [5]. У якості прикладу опишемо класи «Студент» та «Група» та оголосимо зв'язки між ними.

```
class Student
{
    [int]$id; [string]$surname
    [string]$name; [string]$midname
    [Group]$group;
}
class Group
{
    [int]$number; [string]$specialty
    [Student[]]$students
}
```

Так, об'єкт класу «Студент» містить посилання на об'єкт класу «Група», оскільки студент може належати до однієї групи, у свою чергу до однієї групи входить множина студентів, що відображується у властивості, що представляє собою масив посилань на об'єкти класу «Студент».

Створити нові об'єкти власних класів дозволяє командлет `New-Object`.

```
$s=New-Object Student; $g=New-Object Group; $s.Group=$g
```

Довгостроково ж зберегти об'єкти дозволяють файли у форматі XML, з якими у PowerShell існують достатньо розвинуті методи роботи [6]. У подальші плани автора входить дослідження побудови повноцінної об'єктної бази даних засобми оболонки PowerShell у ОС Windows.

Список використаних джерел:

1. Попов А. В. Введение в Windows PowerShell. – СПб.: БХВ-Петербург, 2009.- 464 с.
2. Б. Книттель. Windows 7. Скрипты, автоматизация и командная строка. СПб.: Питер, 2012, 764с.
3. The Object Data Standard: ODMG 3.0. Edited by R.G.G. Cattell Douglas K. Barry. Morgan Kaufmann Publishers San Francisco, California, 2000.
4. Харрингтон Д. Проектирование объектно-ориентированных баз данных: Пер. с англ. – М.: ДМК Пресс, 2001. – 272 с.: ил.
5. B. Payette, R. Siddaway. Windows PowerShell in Action, Third Edition. Manning Publications Co, 2018., 897 p.
6. M. Shepard, C. Venkatesan, S. Talaat, B. Blawat. PowerShell: Automating Administrative Tasks. Packt Publishing Ltd., 2017, 718 p.