**Resolvendo o problema com PowerShell**

A nossa primeira tarefa era **obter o serviço**. Queremos verificar o estado dele para saber se ele está rodando ou não. Em inglês ficaria "get service"! Essa é uma convenção bem intuitiva do PowerShell, que é sempre o **verbo** (o que desejamos realizar), e o **objeto**(serviço). Começaremos a digitar o comando:

PS > Get-Service

Não se esqueça que você pode usar o auto-completar (tab) do PowerShell, caso não se lembre de algum comando.

No PowerShell as coisas são um pouco diferentes. Para indicar um parâmetro, colocamos um -(hífen). E nós não precisamos saber exatamente o nome do comando, porque no PowerShell tudo tem autocomplete!

Queremos verificar o serviço que tem como o nome de "ByteBank Interface Service". Sendo assim, logo após o hífen, podemos digitar a letra n e usar o "tab" para o autocomplete. Logo após o argumento -Name, podemos inserir o nome do serviço:

Get-Service -Name "ByteBank Interface Service"

E agora passaremos o outro parâmetro referente ao **nome da máquina**. Não temos certeza se é computer, server, ou computerName. Podemos simplesmente colocar o hífen, e ir utilizando o tab até encontrar todos os parâmetros que o comando Get-Service suporta. Vamos escolher o parâmetro -ComputerName, passando o "SRV01" logo em seguida.

Get-Service -Name "ByteBank Interface Service" -ComputerName "SRV01"

Podemos verificar uma propriedade status, pois ao contrário do CMD, o PowerShell é uma linguagem de scripts orientada a objetos. Então, o retorno do Get-Service não é um texto, mas sim um **objeto**. Sendo assim, podemos guardá-lo em uma variável.

Por convenção, sempre começamos o nome de uma variável com o **$**:

PS > $servico = Get-Service -Name "ByteBank Interface Service" -ComputerName "SRV01"

Estamos guardando o comando repetido na variável servico. Podemos perceber que cada palavra-chave tem uma cor diferente, que é outra característica do PowerShell. Como ele é orientado a objetos, o **console** tem o conhecimento de que $servico será uma variável, e por isso é mostrado em verde.

O console também tem o conhecimento de que Get-Service é um comando, e então ele é mostrado em amarelo. O console sabe que -Name e -ComputerName são argumentos do comando Get-Service, por isso a cor deles é cinza. Isso acontece porque estamos utilizando a versão mais atual do PowerShell, que está rodando no Windows 10. Pode ser que essas cores não existam no Windows 7 ou 8, sendo tudo branco. Mas isso pode acontecer se a versão que está sendo utilizada não é a última.

Vamos dar um "Enter" nesse comando. Depois podemos verificar as suas propriedades.

Assim como nas linguagens de programação, utilizamos o **.** (ponto) para ter acesso aos dados de um objeto e damos um "Tab" para que ele vá auto completando com várias propriedades que temos nesse objeto. Vamos encontrar a propriedade status.

PS > $servico = Get-Service -Name "ByteBank Interface Service" -ComputerName "SRV01"

PS > $servico.Status

Running

E também precisamos verificar se o status é igual a running. Se isso for verdade, vamos parar o serviço, e depois reiniciá-lo.

PS > $servico = Get-Service -Name "ByteBank Interface Service" -ComputerName "SRV01"

PS > $servico.Status

Running

PS > if ($servico.Status -eq "Running") {

>> $servico.Stop()

>> $servico.Start()

>> } else {

>> echo "Serviço não estava executando!

>> }"

O operador -eq é o "igual" do PowerShell.

Se esse serviço estiver com o estado de Running, vamos pará-lo com o método .Stop(). Depois que ele for parado, ele será iniciado com o método .Start().

Esse método é muito mais rápido, pois não precisamos executar uma query, não precisamos mandar novamente o nome, e não precisamos abrir uma outra conexão com o servidor. Já temos tudo isso dentro do objeto $servico.

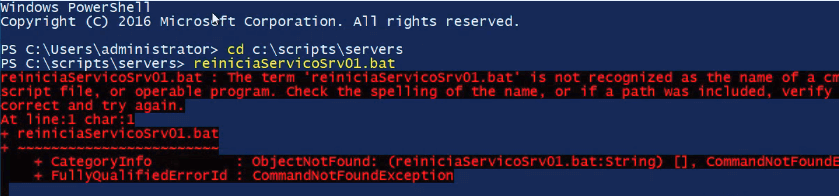
**Conhecendo o CommandPrecedence**

Uma outra preocupação do time é que temos muitos scripts escritos em arquivos .batque já foram testados, homologados e estão funcionando em produção. Será que teremos que reescrever todos esses scripts? Bom, vamos tentar executar, no PowerShell, o script reiniciaServicoSrv01 que montamos anteriormente da mesma forma que estamos acostumados com o prompt de comando. Vamos acessar a pasta que se encontra esse script.

PS > cd c:\scripts\servers

PS > reiniciaServicoSrv01.bat

Assim que damos um "Enter", aparece um erro:



O PowerShell não reconhece o termo reiniciaServicoSrv01.bat como um comando, um script ou um programa. Na verdade, essa é uma mudança que a Microsoft fez no PowerShell por questões de segurança. Um fato muito comum é executar um comando como o logoff, ou como exit, e errar alguma letra. Era uma prática comum de alguns vírus, de colocar no diretório local do prompt, um comando com a mesma letra que erramos, por exemplo exitt.

O PowerShell não permite que executemos um comando .bat ou executável .exedessa forma convencional. Ele exige que nós sejamos explícitos quando queremos executar esse comando. Veja na imagem abaixo, que nos é mostrada uma sugestão:

A sugestão que o PowerShell nos dá, é que esse comando seja executado com o **Command\_Precedence**. O Command Precedence nada mais é do que dois caracteres especiais: **.\**. Ao utilizar o Command Precedence, indicamos que queremos executar o reiniciaServicoSrv01.bat em nossa máquina. Vamos tentar novamente com essa alteração:

PS > .\reiniciaServicoSrv01.bat