

#####

<#

.SYNOPSIS

Writes data only to SQL Server tables.

.DESCRIPTION

Writes data only to SQL Server tables. However, the data source is not limited to SQL Server; any data source can be used, as long as the data can be loaded to a DataTable instance or read with a IDataReader instance.

.INPUTS

None

You cannot pipe objects to Write-DataTable

.OUTPUTS

None

Produces no output

.EXAMPLE

```
$dt = Invoke-Sqlcmd2 -ServerInstance "Z003\R2" -Database pubs "select * from authors"
```

```
Write-DataTable -ServerInstance "Z003\R2" -Database pubscopy -TableName authors -Data $dt
```

This example loads a variable dt of type DataTable from query and write the datatable to another database

.NOTES

Write-DataTable uses the SqlBulkCopy class see links for additional information on this class.

Version History

v1.0 - Chad Miller - Initial release

v1.1 - Chad Miller - Fixed error message

.LINK

<http://msdn.microsoft.com/en-us/library/30c3y597%28v=VS.90%29.aspx>

#>

```
function Write-DataTable
```

```
{
```

```
    [CmdletBinding()]
```

```
    param(
```

```
        [Parameter(Position=0, Mandatory=$true)] [string]$ServerInstance,
```

```
        [Parameter(Position=1, Mandatory=$true)] [string]$Database,
```

```
        [Parameter(Position=2, Mandatory=$true)] [string]$TableName,
```

```
        [Parameter(Position=3, Mandatory=$true)] $Data,
```

```
        [Parameter(Position=4, Mandatory=$false)] [string]$Username,
```

```
        [Parameter(Position=5, Mandatory=$false)] [string]$Password,
```

```
        [Parameter(Position=6, Mandatory=$false)] [Int32]$BatchSize=50000,
```

```
        [Parameter(Position=7, Mandatory=$false)] [Int32]$QueryTimeout=0,
```

```
        [Parameter(Position=8, Mandatory=$false)] [Int32]$ConnectionTimeout=15
```

```
)
```

```
$conn=new-object System.Data.SqlClient.SqlConnection
```

```
if ($Username)
```

```
{ $ConnectionString = "Server={0};Database={1};User ID={2};Password={3};Trusted_Connection=False;Connect  
Timeout={4}" -f $ServerInstance,$Database,$Username,$Password,$ConnectionTimeout }
```

```
else
```

```
{ $ConnectionString = "Server={0};Database={1};Integrated Security=True;Connect Timeout={2}" -f  
$ServerInstance,$Database,$ConnectionTimeout }
```

```
$conn.ConnectionString=$ConnectionString
```

```
try
```

```
{
```

```
$conn.Open()
```

```
# $bulkCopy = new-object ("Data.SqlClient.SqlBulkCopy") $connectionString
```

```
$bulkCopy = new-object Data.SqlClient.SqlBulkCopy $connectionString, FireTriggers
```

```
$bulkCopy.DestinationTableName = $tableName
```

```
$bulkCopy.BatchSize = $BatchSize
```

```
        $bulkCopy.BulkCopyTimeout = $QueryTimeOut

        $bulkCopy.WriteToServer($Data)

        $conn.Close()
    }

    catch

    {

        $ex = $_.Exception

        Write-Error "$ex.Message"

        write-eventlog -logname "Windows PowerShell" -source SP-STMTRECOMPILE -eventID 3001 -entrytype
Information -message "$ex.Message" -category 1 -rawdata 10,20

        continue
    }

} #Write-DataTable
```

```
#####
```

```
<#
```

.SYNOPSIS

Creates a DataTable for an object

.DESCRIPTION

Creates a DataTable based on an objects properties.

.INPUTS

Object

Any object can be piped to Out-DataTable

.OUTPUTS

System.Data.DataTable

.EXAMPLE

```
$dt = Get-Alias | Out-DataTable
```

This example creates a DataTable from the properties of Get-Alias and assigns output to \$dt variable

.NOTES

Adapted from script by Marc van Orsouw see link

Version History

v1.0 - Chad Miller - Initial Release

v1.1 - Chad Miller - Fixed Issue with Properties

v1.2 - Chad Miller - Added setting column datatype by property as suggested by emp0

v1.3 - Chad Miller - Correct issue with setting datatype on empty properties

v1.4 - Chad Miller - Corrected issue with DBNull

.LINK

<http://thepowershellguy.com/blogs/posh/archive/2007/01/21/powershell-gui-scripblock-monitor-script.aspx>

#>

```
function Out-DataTable
```

```
{
```

```
    [CmdletBinding()]
```

```
    param([Parameter(Position=0, Mandatory=$true, ValueFromPipeline = $true)] [PSObject[]]$InputObject)
```

```
    Begin
```

```
    {
```

```
        $dt = new-object Data.datatable
```

```
        $First = $true
```

```
    }
```

```
    Process
```

```
    {
```

```
        foreach ($object in $InputObject)
```

```
        {
```

```
            $DR = $DT.NewRow()
```

```
foreach($property in $object.PsObject.get_properties())
{
    if ($first)
    {
        $Col = new-object Data.DataColumn

        $Col.ColumnName = $property.Name.ToString()

        if ($property.value)
        {
            if ($property.value -isnot [System.DBNull])
            { $Col.DataType = $property.value.gettype() }

        }

        $DT.Columns.Add($Col)
    }

    if ($property.IsArray)
    { $DR.Item($property.Name) = $property.value | ConvertTo-XML -AS String -NoTypeInfo -Depth 1
}

    else { $DR.Item($property.Name) = $property.value }

}
}
```

```
        $DT.Rows.Add($DR)

        $First = $false
    }
}

End

{
    Write-Output @(,$dt)
}

} #Out-DataTable
```