PowerShell et la photo

Organiser ces photos avec la lecture de l'exif

L'EXIF, c'est un format de fichier utilisé pour les images prises par un appareil photographique numérique, la première version de l'Exif date de 1995.
C'est une mine d'or pour organiser ces photos.
Je suis tombé sur le travail de **Jon Gallant**, sur son blog , depuis je n'utilise que ce script.
Attention pour les utilisateurs de la suite Adobe, on est loin des capacités d'un Adobe Camera Raw.

```
Param(
    [string]$source,
    [string]$dest,
    [string]$format = "yyyy/MM_yyyy/dd_MM_yyyy"
)

$shell = New-Object -ComObject Shell.Application

function Get-File-Date {
    [CmdletBinding()]
    Param (
        $object
    )

    $dir = $shell.NameSpace( $object.Directory.FullName )
    $file = $dir.ParseName( $object.Name )

    # First see if we have Date Taken, which is at index 12
    $date = Get-Date-Property-Value $dir $file 12

    if ($null -eq $date) {
        # If we don't have Date Taken, then find the oldest date from all
date properties
        0..287 | ForEach-Object {
            $name = $dir.GetDetailsof($dir.items, $_)

            if ( $name -match '(date)|(created)') {
                # Only get value if date field because the GetDetailsOf call
is expensive
                $tmp = Get-Date-Property-Value $dir $file $_
                if ( ($null -ne $tmp) -and (($null -eq $date) -or ($tmp -lt
$date))) {
                    $date = $tmp
                }
            }
        }
    }
```

```
    return $date
}

function Get-Date-Property-Value {
    [CmdletBinding()]

    Param (
        $dir,
        $file,
        $index
    )

    $value = ($dir.GetDetailsof($file, $index) -replace "`u{200e}") -replace
"`u{200f}"
    if ($value -and $value -ne '') {
        return [DateTime]::ParseExact($value, "g", $null)
    }
    return $null
}

Get-ChildItem -Attributes !Directory $source -Recurse |
Foreach-Object {
    Write-Host "Processing $_"

    $date = Get-File-Date $_

    if ($date) {
        $destinationFolder = Get-Date -Date $date -Format $format
        $destinationPath = Join-Path -Path $dest -ChildPath
$destinationFolder

        # See if the destination file exists and rename until we get a
unique name
        $newFullName = Join-Path -Path $destinationPath -ChildPath $_.Name
        if ($_.FullName -eq $newFullName) {
            Write-Host "Skipping: Source file and destination files are at
the same location. $_"
            return
        }

        $newNameIndex = 1
        $newName = $_.Name

        while (Test-Path -Path $newFullName) {
            $newName = ($_.BaseName + "_$newNameIndex" + $_.Extension)
            $newFullName = Join-Path -Path $destinationPath -ChildPath
$newName
            $newNameIndex += 1
        }
```

```
        # If we have a new name, then we need to rename in current location
before moving it.
        if ($newNameIndex -gt 1) {
            Rename-Item -Path $_.FullName -NewName $newName
        }

        Write-Host "Moving $_ to $newFullName"

        # Create the destination directory if it doesn't exist
        if (!(Test-Path $destinationPath)) {
            New-Item -ItemType Directory -Force -Path $destinationPath
        }

        robocopy $_.DirectoryName $destinationPath $newName /mov
    }
}
```

Lire des informations EXIF dans un fichier image

```
Function Get-Image
{
    begin{
[void][System.Reflection.Assembly]::LoadWithPartialName("System.Drawing")
|Out-Null
    }
     process{
          $fi=[System.IO.FileInfo]$_
          if( $fi.Exists){
               $img = [System.Drawing.Image]::FromFile($_)
               $img.Clone()
               $img.Dispose()
          }else{
               Write-Host "File not found: $_" -fore yellow
          }
     }
    end{}
}

Get-ChildItem -Path "F:\Wallpaper" -Filter *.jpg | ForEach-Object {
    $image = $_ | Get-Image
    New-Object PSObject -Property  @{
        File = $_.name
        Fullname = $_.Fullname
        Height = $image.Height
        Width = $image.Width
    };
}
```

Effacer les dossiers vides

```
# Set to true to test the script
$whatIf = $false
# Remove hidden files, like thumbs.db
$removeHiddenFiles = $false
# Get hidden files or not. Depending on removeHiddenFiles setting
$getHiddelFiles = !$removeHiddenFiles
# Remove empty directories locally
Function Delete-EmptyFolder($path)
{
    # Go through each subfolder,
    Foreach ($subFolder in Get-ChildItem -Force -Literal $path -Directory)
    {
        # Call the function recursively
        Delete-EmptyFolder -path $subFolder.FullName
    }
    # Get all child items
    $subItems = Get-ChildItem -Force:$getHiddelFiles -LiteralPath $path
    # If there are no items, then we can delete the folder
    # Exluce folder: If (($subItems -eq $null) -and (-
Not($path.contains("DfsrPrivate"))))
    If ($subItems -eq $null)
    {
        Write-Host "Removing empty folder '${path}'"
        Remove-Item -Force -Recurse:$removeHiddenFiles -LiteralPath $Path -
WhatIf:$whatIf
    }
}
# Run the script
Delete-EmptyFolder -path "E:\_Backup"
```

Déplacer les fichiers par date vers un autre dossier

Je n'utilise plus cette version de script, je préfère utiliser le script avec l'utilisation des informations EXIF

```
$Destination = 'F:\Photos\'
$Source = 'F:\Photos\2019'
Get-ChildItem -Path $Source -File -Filter *.jpg -Recurse| ForEach-Object {
$Year = $_.lastwritetime.year.ToString("0000")
$Month = $_.lastwritetime.month.ToString("00")
$Day = $_.lastwritetime.day.ToString("00")
$D = Join-Path $Destination -ChildPath "$Year\$Month-$Year\$Day-$Month-
$Year"
```

```
if ((Test-Path $D) -eq $false){New-Item -Path $D -ItemType Directory -Force
| Out-Null}
Move-Item -Path $_.FullName -Destination (Join-Path -Path $D -ChildPath
$_.name ) -Force
}
```