How to convert PDF to PNG from URL asynchronously for PDF to image API in PowerShell with PDF.co Web API

See how to convert PDF to PNG from URL asynchronously to have PDF to image API in PowerShell

Today we will explain the steps and algorithm of how to convert PDF to PNG from URL asynchronously and how to make it work in your application. PDF to image API in PowerShell can be implemented with PDF.co Web API. PDF.co Web API is the flexible Web API that includes full set of functions from e-signature requests to data extraction, OCR, images recognition, pdf splitting and pdf splitting. Can also generate barcodes and read barcodes from images, scans and pdf.

The SDK samples like this one below explain how to quickly make your application do PDF to image API in PowerShell with the help of PDF.co Web API. Open your PowerShell project and simply copy & paste the code and then run your app! Tutorials are available along with installed PDF.co Web API if you'd like to dive deeper into the topic and the details of the API.

ByteScout free trial version is available for FREE download from our website. Programming tutorials along with source code samples are included.

PowerShell - ConvertPdfToPngFromUrlAsynchronously.ps1

```
# Cloud API asynchronous "PDF To PNG" job example.
# Allows to avoid timeout errors when processing huge or scanned PDF documents.

# The authentication key (API Key).
# Get your own by registering at https://app.pdf.co/documentation/api
$API_KEY = "***********************************"

# Direct URL of source PDF file.
$SourceFileUrl = "https://bytescout-com.s3.amazonaws.com/files/demo-files/cloud-
api/pdf-to-image/sample.pdf"
# Comma-separated list of page indices (or ranges) to process. Leave empty for all
pages. Example: '0,2-5,7-'.
$Pages = ""
# PDF document password. Leave empty for unprotected documents.
$Password = ""
# (!) Make asynchronous job
$Async = $true


# Prepare URL for `PDF To PNG` API call
$query = "https://api.pdf.co/v1/pdf/convert/to/png?password={0}&pages={1}&url=
{2}&async={3}" -f `
    $Password, $Pages, $SourceFileUrl, $Async
```

```
$query = [System.Uri]::EscapeUriString($query)

try {
    # Execute request
    $jsonResponse = Invoke-RestMethod -Method Get -Headers @{ "x-api-key" = $API_KEY
} -Uri $query

    if ($jsonResponse.error -eq $false) {
        # Asynchronous job ID
        $jobId = $jsonResponse.jobId
        # URL of generated JSON file available after the job completion; it will
contain URLs of result PDF files.
        $resultJsonFileUrl = $jsonResponse.url

        # Check the job status in a loop.
        do {
            $statusCheckUrl = "https://api.pdf.co/v1/job/check?jobid=" + $jobId
            $jsonStatus = Invoke-RestMethod -Method Get -Headers @{ "x-api-key" =
$API_KEY } -Uri $statusCheckUrl

            # Display timestamp and status (for demo purposes)
            Write-Host "$(Get-date): $($jsonStatus.status)"

            if ($jsonStatus.status -eq "success") {
                # Download JSON file with URLs of result PDF files
                $jsonPngUrls = Invoke-RestMethod -Method Get -Uri $resultJsonFileUrl

                # Download generated PNG files
                $part = 1;
                foreach ($url in $jsonPngUrls) {
                    $localFileName = ".\page$($part).png"

                    Invoke-WebRequest -Headers @{ "x-api-key" = $API_KEY } -OutFile
$localFileName -Uri $url

                    Write-Host "Downloaded `"$($localFileName)`""
                    $part++
                }
                break
            }
            elseif ($jsonStatus.status -eq "working") {
                # Pause for a few seconds
                Start-Sleep -Seconds 3
            }
            else {
                Write-Host $jsonStatus.status
                break
            }
        }
        while ($true)
    }
    else {
        # Display service reported error
        Write-Host $jsonResponse.message
    }
}
catch {
    # Display request error
    Write-Host $_.Exception
}
```

PowerShell - run.bat

```
@echo off

powershell -NoProfile -ExecutionPolicy Bypass -Command "&
.\ConvertPdfToPngFromUrlAsynchronously.ps1"
echo Script finished with errorlevel=%errorlevel%

pause
```
