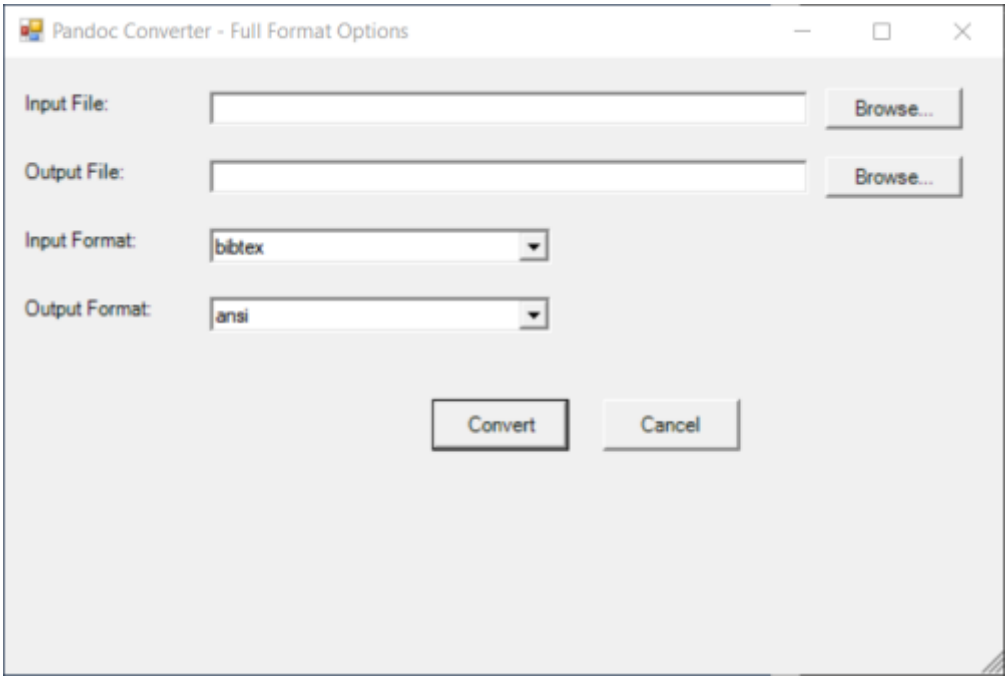

PS: Pandoc Converter GUI

The script is a PowerShell application that uses Windows Forms to create a graphical user interface (GUI) for converting files from one format to another using the Pandoc tool. In short, the script allows the user to select the input file, output file and input and output formats supported by Pandoc.

For me personally, this script comes in handy many times whenever I create something. Usually when I write something it is in several formats. One is the syntax dokuwiki which is used here on this site, very often many of the things I write are also written in LaTeX and sometimes I write something in MarkDown. Pandoc is a tool that allows you to easily convert between these formats.

Quite a few of the articles on this wiki have been converted from .md and .tex formats using this script.

Loading the required libraries

The script loads two .NET libraries to create the GUI:

- System.Windows.Forms - for GUI operations (windows, buttons, etc.).
- System.Drawing - for graphical operations.

Selection of input and output formats

The script defines two lists containing the input and output formats supported by Pandoc:

- `$availableInputFormats` - a list of file formats that can be used as input (e.g. markdown, docx, bibtex).
- `$availableOutputFormats` - a list of file formats that can be generated as output (e.g. pdf,

html, docx, dokuwiki).

The user selects the formats from drop-down lists (ComboBox) in the GUI.

Form creation

The script creates a form with the following elements:

- A text box to select the input file (Input File).
- A text box to select the output file (Output File).
- „Browse...” buttons. for browsing input and output files.
- Drop-down combinations (ComboBox) for selecting input and output formats.
- „Convert” button to start the conversion.
- „Cancel” button to cancel the operation.

Input validation

When the 'Convert' button is clicked, the script checks that the user has selected the input and output files. If either of them is empty, an error message is displayed.

Support for specific cases

If the input file has the extension `.tex` and the user selects the format „texinfo”, the script automatically changes the input format to „latex”, as Pandoc requires TeX files to be converted as „latex” rather than „texinfo”.

Running the conversion

The script checks whether Pandoc is installed and available on the system (by checking the `pandoc` command). If Pandoc is not available, the user receives an appropriate error message.

If all conditions are met, the script runs the Pandoc command, passing the appropriate arguments:

- `-f` - input format
- `-t` - output format
- `-o` - output file

When the conversion is complete, a success message is displayed.

Summary

The script provides an easy way to convert files between multiple formats, offering the user a simple graphical interface for selecting files and formats. It supports a variety of input and output formats

supported by Pandoc and provides data validation and error handling.

script code:

pandoc.ps1

pandoc.ps1

```powershell
# Ensure necessary assemblies are loaded
Add-Type -AssemblyName System.Windows.Forms
Add-Type -AssemblyName System.Drawing

# Comprehensive list of Pandoc-supported input formats
$availableInputFormats = @(
    "bibtex",
    "biblatex",
    "bits",
    "commonmark",
    "commonmark_x",
    "creole",
    "csljson",
    "csv",
    "tsv",
    "djot",
    "docbook",
    "docx",
    "dokuwiki",
    "endnotexml",
    "epub",
    "fb2",
    "gfm",
    "haddock",
    "html",
    "ipynb",
    "jats",
    "jira",
    "json",
    "latex",
    "markdown",
    "markdown_mmd",
    "markdown_phpextra",
    "markdown_strict",
    "mediawiki",
    "man",
    "mdoc",
    "muse",
    "native",
    "odt",
    "opml",
    "org",
```

```
    "pod",
    "ris",
    "rtf",
    "rst",
    "t2t",
    "textile",
    "tikiwiki",
    "twiki",
    "typst",
    "vimwiki"
)

# Comprehensive list of Pandoc-supported output formats
$availableOutputFormats = @(
    "ansi",
    "asciidoc",
    "asciidoc_legacy",
    "asciidoctor",
    "beamer",
    "bibtex",
    "biblatex",
    "chunkedhtml",
    "commonmark",
    "commonmark_x",
    "context",
    "csljson",
    "djot",
    "docbook",
    "docbook4",
    "docbook5",
    "docx",
    "dokuwiki",
    "epub",
    "epub3",
    "epub2",
    "fb2",
    "gfm",
    "haddock",
    "html",
    "html5",
    "html4",
    "icml",
    "ipynb",
    "jats_archiving",
    "jats_articleauthoring",
    "jats_publishing",
    "jats",
    "jira",
    "json",
    "latex",
    "man",
```

```
    "markdown",
    "markdown_mmd",
    "markdown_phpextra",
    "markdown_strict",
    "markua",
    "mediawiki",
    "ms",
    "muse",
    "native",
    "odt",
    "opml",
    "opendocument",
    "org",
    "pdf",
    "plain",
    "pptx",
    "rst",
    "rtf",
    "texinfo",
    "textile",
    "slideous",
    "slidy",
    "dzslides",
    "revealjs",
    "s5",
    "tei",
    "typst",
    "xwiki",
    "zimwiki"
)

# Create the main form
$form = New-Object System.Windows.Forms.Form
$form.Text = "Pandoc Converter - Full Format Options"
$form.Size = New-Object System.Drawing.Size(600,400)
$form.StartPosition = "CenterScreen"

# --- Input File Controls ---
$inputLabel = New-Object System.Windows.Forms.Label
$inputLabel.Location = New-Object System.Drawing.Point(10,20)
$inputLabel.Size = New-Object System.Drawing.Size(100,20)
$inputLabel.Text = "Input File:"
$form.Controls.Add($inputLabel)

$inputTextBox = New-Object System.Windows.Forms.TextBox
$inputTextBox.Location = New-Object System.Drawing.Point(120,20)
$inputTextBox.Size = New-Object System.Drawing.Size(350,20)
$form.Controls.Add($inputTextBox)

$inputBrowseButton = New-Object System.Windows.Forms.Button
$inputBrowseButton.Location = New-Object System.Drawing.Point(480,18)
```

```
$inputBrowseButton.Size = New-Object System.Drawing.Size(80,24)
$inputBrowseButton.Text = "Browse..."
$inputBrowseButton.Add_Click({
    $openDialog = New-Object System.Windows.Forms.OpenFileDialog
    $openDialog.Filter = "All Files (*.*)|*.*"
    $openDialog.Title = "Select an Input File"
    if ($openDialog.ShowDialog() -eq
[System.Windows.Forms.DialogResult]::OK) {
        $inputTextBox.Text = $openDialog.FileName
    }
})
$form.Controls.Add($inputBrowseButton)

# --- Output File Controls ---
$outputLabel = New-Object System.Windows.Forms.Label
$outputLabel.Location = New-Object System.Drawing.Point(10,60)
$outputLabel.Size = New-Object System.Drawing.Size(100,20)
$outputLabel.Text = "Output File:"
$form.Controls.Add($outputLabel)

$outputTextBox = New-Object System.Windows.Forms.TextBox
$outputTextBox.Location = New-Object System.Drawing.Point(120,60)
$outputTextBox.Size = New-Object System.Drawing.Size(350,20)
$form.Controls.Add($outputTextBox)

$outputBrowseButton = New-Object System.Windows.Forms.Button
$outputBrowseButton.Location = New-Object System.Drawing.Point(480,58)
$outputBrowseButton.Size = New-Object System.Drawing.Size(80,24)
$outputBrowseButton.Text = "Browse..."
$outputBrowseButton.Add_Click({
    $saveDialog = New-Object System.Windows.Forms.SaveFileDialog
    $saveDialog.Filter = "All Files (*.*)|*.*"
    $saveDialog.Title = "Select Output File Destination"
    if ($saveDialog.ShowDialog() -eq
[System.Windows.Forms.DialogResult]::OK) {
        $outputTextBox.Text = $saveDialog.FileName
    }
})
$form.Controls.Add($outputBrowseButton)

# --- Input Format ComboBox ---
$inputFormatLabel = New-Object System.Windows.Forms.Label
$inputFormatLabel.Location = New-Object System.Drawing.Point(10,100)
$inputFormatLabel.Size = New-Object System.Drawing.Size(100,20)
$inputFormatLabel.Text = "Input Format:"
$form.Controls.Add($inputFormatLabel)

$inputFormatCombo = New-Object System.Windows.Forms.ComboBox
$inputFormatCombo.Location = New-Object System.Drawing.Point(120,100)
$inputFormatCombo.Size = New-Object System.Drawing.Size(200,20)
$inputFormatCombo.DropDownStyle =
```

```
[System.Windows.Forms.ComboBoxStyle]::DropDownList
foreach ($fmt in $availableInputFormats) {
    $inputFormatCombo.Items.Add($fmt) | Out-Null
}
$inputFormatCombo.SelectedIndex = 0
$form.Controls.Add($inputFormatCombo)

# --- Output Format ComboBox ---
$outputFormatLabel = New-Object System.Windows.Forms.Label
$outputFormatLabel.Location = New-Object System.Drawing.Point(10,140)
$outputFormatLabel.Size = New-Object System.Drawing.Size(100,20)
$outputFormatLabel.Text = "Output Format:"
$form.Controls.Add($outputFormatLabel)

$outputFormatCombo = New-Object System.Windows.Forms.ComboBox
$outputFormatCombo.Location = New-Object System.Drawing.Point(120,140)
$outputFormatCombo.Size = New-Object System.Drawing.Size(200,20)
$outputFormatCombo.DropDownStyle =
[System.Windows.Forms.ComboBoxStyle]::DropDownList
foreach ($fmt in $availableOutputFormats) {
    $outputFormatCombo.Items.Add($fmt) | Out-Null
}
$outputFormatCombo.SelectedIndex = 0
$form.Controls.Add($outputFormatCombo)

# --- Convert and Cancel Buttons ---
$convertButton = New-Object System.Windows.Forms.Button
$convertButton.Location = New-Object System.Drawing.Point(250,200)
$convertButton.Size = New-Object System.Drawing.Size(80,30)
$convertButton.Text = "Convert"
$convertButton.DialogResult = [System.Windows.Forms.DialogResult]::OK
$form.Controls.Add($convertButton)

$cancelButton = New-Object System.Windows.Forms.Button
$cancelButton.Location = New-Object System.Drawing.Point(350,200)
$cancelButton.Size = New-Object System.Drawing.Size(80,30)
$cancelButton.Text = "Cancel"
$cancelButton.DialogResult =
[System.Windows.Forms.DialogResult]::Cancel
$form.Controls.Add($cancelButton)

$form.AcceptButton = $convertButton
$form.CancelButton = $cancelButton

# Show the form and wait for user interaction
$result = $form.ShowDialog()

if ($result -eq [System.Windows.Forms.DialogResult]::OK) {
    $inputFile = $inputTextBox.Text
    $outputFile = $outputTextBox.Text
    $selectedInputFormat = $inputFormatCombo.SelectedItem
```

```
    $selectedOutputFormat = $outputFormatCombo.SelectedItem

    # Validate file selections
    if ([string]::IsNullOrWhiteSpace($inputFile)) {
        [System.Windows.Forms.MessageBox]::Show("Please select an input
file.","Error",
            [System.Windows.Forms.MessageBoxButtons]::OK,
            [System.Windows.Forms.MessageBoxIcon]::Error)
        exit
    }
    if ([string]::IsNullOrWhiteSpace($outputFile)) {
        [System.Windows.Forms.MessageBox]::Show("Please select an
output file.","Error",
            [System.Windows.Forms.MessageBoxButtons]::OK,
            [System.Windows.Forms.MessageBoxIcon]::Error)
        exit
    }

    # If input file has .tex extension and user selected 'texinfo',
override to 'latex'
    $ext = [System.IO.Path]::GetExtension($inputFile).ToLower()
    if ($ext -eq ".tex" -and $selectedInputFormat -eq "texinfo") {
        [System.Windows.Forms.MessageBox]::Show("For TeX files, the
input format has been changed from 'texinfo' to 'latex'.",
            "Info", [System.Windows.Forms.MessageBoxButtons]::OK,
            [System.Windows.Forms.MessageBoxIcon]::Information)
        $selectedInputFormat = "latex"
    }

    # Check if pandoc is available
    if (-not (Get-Command pandoc -ErrorAction SilentlyContinue)) {
        [System.Windows.Forms.MessageBox]::Show("Pandoc is not
installed or not in the system path. Please install pandoc and try
again.",
            "Error", [System.Windows.Forms.MessageBoxButtons]::OK,
            [System.Windows.Forms.MessageBoxIcon]::Error)
        exit
    }

    # Run pandoc conversion using selected formats (-f for input, -t
for output)
    pandoc "$inputFile" -f $selectedInputFormat -t
$selectedOutputFormat -o "$outputFile"

    [System.Windows.Forms.MessageBox]::Show("Conversion
complete.`nOutput saved as:" + "`n" + "$outputFile",
        "Success", [System.Windows.Forms.MessageBoxButtons]::OK,
        [System.Windows.Forms.MessageBoxIcon]::Information)
} else {
    Write-Host "Operation cancelled by user."
```

```
}
```