SourceFilesToSinglePDF

Extensions extracted

.py
.html
.css

```
############# Parameters for the user
extensions = ['.py', '.html', '.css']
#Example of extensions supported : `extensions = ['.txt', '.csv', '.py',
'.json', '.xml', '.html', '.css', '.md', '.log', '.yaml', '.ini', '.cfg',
'.bat', '.sh', '.sql', '.php', '.java', '.cpp', '.c', '.h', '.js', '.jsx',
'.ts', '.tsx', '.rb', '.pl', '.r', '.go', '.swift', '.vb']`
folder_path = r'C:\Users\Utilisateur\Desktop\SourceFilesToSinglePDF'
#Example (Windows) : folder_path = r'C:\Users\YourUsername\Documents\MyFiles'
output_path = r'C:\Users\Utilisateur\Desktop\dezsfrgt\pdf_generator.pdf'
#Example (Windows) : output_path = r'C:\Users\YourUsername\Desktop\MyPDF.pdf'
# Once the parameters set, execute this code (/!\ If the PDF already exists, be
sure that the file is closed during the execution)
#############
if __name__ == "__main__":
    from pdf_generator import generate_pdf_from_folder
    generate_pdf_from_folder(folder_path, output_path)
```

```
from reportlab.lib.pagesizes import A4
from reportlab.pdfgen import canvas
from reportlab.lib.units import cm
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.pdfbase.pdfmetrics import stringWidth
import os
import textwrap
from PyPDF2 import PdfWriter, PdfReader
from title_page import generate_title_page
from pdf_utils import draw_footer, draw_header
from table_of_contents import generate_table_of_contents
from config_and_launch import extensions, folder_path, output_path
def generate_pdf_from_folder(folder_path, output_path):
    page_width, page_height = A4
    margin = 2 * 28.3465  # Margin in points (2cm)
    y_start = page_height - margin - 70  # Adjust the starting position as 
needed
    y_threshold = margin + 70  # Adjust the threshold as needed
    width = page_width - 2 * margin  # Width of the text
    file_pages = []  # List for storing the page numbers of each file
    # Temporary output paths for the three passes
    temp_output_path_title = output_path.replace('.pdf', '_temp_title.pdf')
    title_canvas = canvas.Canvas(temp_output_path_title, pagesize=A4)
    temp_output_path_code = output_path.replace('.pdf', '_temp_code.pdf')
    # First pass to generate the code pages and record the page numbers
    c = canvas.Canvas(temp_output_path_code, pagesize=A4)
    total_page_num = 1
    for root, dirs, files in os.walk(folder_path):
        for file in files:
            if any(file.endswith(ext) for ext in extensions):
                file_path = os.path.join(root, file)
                relative_path = os.path.relpath(file_path, folder_path)
                full_path = os.path.join(os.path.basename(folder_path), 
relative_path)
                with open(file_path, 'r', encoding='utf-8') as f:
                    content = f.read()
                file_page_start = total_page_num
                file_page_num = 1
                # Header
                draw_header(c, margin, page_width, page_height, folder_path, 
full_path, file_page_num)
                c.setFont("Courier", 10)
                # Draw file content
                lines = content.split('\n')
                y = y_start
                for line in lines:
                    # Wrap the line
                    wrapper = textwrap.TextWrapper(width=int(width / 6))  # 6
approximates average character width in points
                    wrapped_lines = wrapper.wrap(line)
                    for wrapped_line in wrapped_lines:
                        if y <= y_threshold:
                            # Footer
                            draw_footer(c, margin, page_width, total_page_num)
                            # Create new page and add header
                            c.showPage()
                            file_page_num += 1
                            total_page_num += 1
                            draw_header(c, margin, page_width, page_height,
folder_path, full_path, file_page_num)
                            c.setFont("Courier", 10)
                            y = y_start
                        c.drawString(margin, y, wrapped_line)
                        y -= 14  # Adjust the line spacing as needed
                # Footer
                draw_footer(c, margin, page_width, total_page_num)
                c.showPage()
                file_pages.append((full_path, (file_page_start,
total_page_num)))
                total_page_num += 1
    c.save()
    # Generate the title page
    generate_title_page(title_canvas, folder_path, extensions, page_width,
page_height, margin)
    title_canvas.save()
    # Generate the table of contents
    toc_file = generate_table_of_contents(folder_path, page_width, page_height,
margin, file_pages)
    # Create a PdfWriter object
    writer = PdfWriter()
    # Add the title page at the beginning
    reader = PdfReader(temp_output_path_title)
    for i in range(len(reader.pages)):
        writer.add_page(reader.pages[i])
    # Add the ToC pages
    reader = PdfReader(toc_file)
    for i in range(len(reader.pages)):
        writer.add_page(reader.pages[i])
    # Add the previously generated code pages
    reader = PdfReader(temp_output_path_code)
    for i in range(len(reader.pages)):
        writer.add_page(reader.pages[i])
    # Write the final output PDF file
    with open(output_path, 'wb') as f:
        writer.write(f)
    # Delete the temporary files
    os.remove(temp_output_path_title)
    os.remove(temp_output_path_code)
    os.remove(toc_file)
```

```
from reportlab.lib.pagesizes import A4
from reportlab.pdfgen import canvas
import os
def draw_footer(c, margin, page_width, total_page_num, is_toc=False):
    c.setFont("Helvetica", 12)
    if not is_toc:  # Don't draw the page number if it's the Table of Contents
        c.drawRightString(page_width - margin, margin, f"Doc Page:
{total_page_num}")
    c.line(margin, margin + 30, page_width - margin, margin + 30)
def draw_header(c, margin, page_width, page_height, folder_path, file_path,
file_page_num, is_toc=False):
    c.setFont("Helvetica-Bold", 14)
    c.drawString(margin, page_height - margin, f"Project:
{os.path.basename(folder_path)}")
    c.setFont("Helvetica", 14)
    if is_toc:  # If it's the Table of Contents
        c.drawString(margin, page_height - margin - 20, 'Table of Contents')
        c.setFont("Helvetica", 12)
        c.drawRightString(page_width - margin, page_height - margin - 20, f"ToC
Page: {file_page_num}")
    else:  # If it's a code page
        c.drawString(margin, page_height - margin - 20, f"Path: {file_path}")
        c.setFont("Helvetica", 12)
        c.drawRightString(page_width - margin, page_height - margin - 20, f"File
Page: {file_page_num}")
    c.line(margin, page_height - margin - 30, page_width - margin, page_height -
margin - 30)
```

```
from reportlab.lib.pagesizes import A4
from reportlab.pdfgen import canvas
from reportlab.lib.units import cm
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.pdfbase.pdfmetrics import stringWidth
import os
import textwrap
from pdf_utils import draw_footer, draw_header
def generate_table_of_contents(folder_path, page_width, page_height, margin,
file_pages):
    toc_canvas = canvas.Canvas("toc.pdf", pagesize=A4)
    toc_page_num = 1
    # Header
    draw_header(toc_canvas, margin, page_width, page_height, folder_path, 'Table
of Contents', 1, is_toc=True)  # use toc_canvas instead of c
    draw_footer(toc_canvas, margin, page_width, toc_page_num, is_toc=True)
    toc_canvas.setFont("Helvetica", 12)
    # Draw the ToC
    y_start = page_height - margin - 70
    y_threshold = margin + 70
    y = y_start
    for directory in sorted(set(os.path.dirname(full_path) for full_path, _ in
file_pages)):
        if y <= y_threshold:
            # Create new page and add header/footer
            toc_canvas.showPage()
            toc_page_num += 1
            draw_header(toc_canvas, margin, page_width, page_height,
folder_path, 'Table of Contents', toc_page_num, is_toc=True)
            draw_footer(toc_canvas, margin, page_width, toc_page_num,
is_toc=True)
            toc_canvas.setFont("Helvetica", 12)
            y = y_start
        toc_canvas.setFont("Helvetica-Bold", 12)
        folder_text = toc_canvas.beginText(margin, y)
        folder_text.textLine(directory)
        toc_canvas.drawText(folder_text)
        toc_canvas.line(margin, y - 1, margin + stringWidth(directory,
"Helvetica-Bold", 12), y - 1)
        y -= 14  # Adjust the line spacing as needed
        toc_canvas.setFont("Helvetica", 12)
        for full_path, page_range in file_pages:
            if os.path.dirname(full_path) == directory:
                if y <= y_threshold:
                    # Create new page and add header/footer
                    toc_canvas.showPage()
                    toc_page_num += 1
                    draw_header(toc_canvas, margin, page_width, page_height,
folder_path, 'Table of Contents', toc_page_num, is_toc=True)
                    draw_footer(toc_canvas, margin, page_width, toc_page_num,
is_toc=True)
                    toc_canvas.setFont("Helvetica", 12)
                    y = y_start
                if page_range[0] == page_range[1]:
                    page_info = f'{page_range[0]}'
                else:
                    page_info = f'{page_range[0]} - {page_range[1]}'
                # The indent, filename and dots
                toc_canvas.drawString(margin + 20, y, full_path)
                page_info_width = stringWidth(page_info, "Helvetica", 12)
                dots_width = page_width - margin - (margin + 20 +
stringWidth(full_path, "Helvetica", 12) + page_info_width)
                toc_canvas.drawString(margin + 20 + stringWidth(full_path,
"Helvetica", 12), y, '.' * int(dots_width / stringWidth('.', "Helvetica", 12)))
                toc_canvas.drawRightString(page_width - margin, y, page_info)
                y -= 14  # Adjust the line spacing as needed
    toc_canvas.save()
    return "toc.pdf"
```

```
from reportlab.lib.pagesizes import A4
from reportlab.pdfgen import canvas
from reportlab.lib.units import cm
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.pdfbase.pdfmetrics import stringWidth
import os
import textwrap
def generate_title_page(c, folder_path, extensions, page_width, page_height,
margin):
    y_start = page_height - margin
    width = page_width - 2 * margin
    title_font_size = 44
    subtitle_font_size = 28
    extensions_font_size = 14
    extension_limit = (page_height - 2*margin) / (extensions_font_size * 1.5)
    # Calculate the total block height
    title_height = title_font_size * 1.5
    subtitle_height = subtitle_font_size * 1.5
    total_block_height = title_height + subtitle_height
    # Centered positions
    y_position = page_height / 2 + total_block_height / 2
    c.setFont("Helvetica-Bold", title_font_size)
    folder_name = os.path.basename(folder_path)
    title_lines = textwrap.wrap(folder_name, width=int(width /
(title_font_size/2)))
    for title_line in title_lines:
        c.drawCentredString(page_width/2, y_position, title_line)
        y_position -= title_font_size * 1.5
    c.setFont("Helvetica", subtitle_font_size)
    subtitle = "Extension extracted" if len(extensions) == 1 else "Extensions
extracted"
    c.drawCentredString(page_width/2, y_position, subtitle)
    y_position -= subtitle_font_size * 1.5
    c.setFont("Courier", extensions_font_size)
    for i, ext in enumerate(extensions):
        if y_position <= margin:
            c.showPage()  # Add a new page for the remaining extensions
            c.setFont("Courier", extensions_font_size)
            y_position = page_height - margin
        extension_line = ', '.join(textwrap.wrap(ext, width=int(width /
(extensions_font_size/2))))
        c.drawCentredString(page_width/2, y_position, extension_line)
        y_position -= extensions_font_size * 1.5
    c.showPage()
```