BASF Working Student Assignment

Shankar Anbalagan
shankanb09@gmail.com

Source code to my solution: https://github.com/ShankarAnbalagan/basf-coding-challenge

Website 1

Scrapy Spider

website1_spider.py

website1 > website1_spider.py > Website1Spider > parse_project

```python
import scrapy
from urllib.parse import urlparse, parse_qs
import time

import website1_scrape
from utils import save_data

class Website1Spider(scrapy.Spider):
    name = "website1"

    def start_requests(self):
        urls = list((f"https://www.uvp-verbund.de/freitextsuche?rstart=0&currentSelectorPage={page_number}" for page_number in range(1, 6)))
        for url in urls:
            yield scrapy.Request(url=url, callback=self.parse)
            time.sleep(5)

    def parse(self, response):
        parsed_url = urlparse(response.url)
        base_url = f"{parsed_url.scheme}://{parsed_url.netloc}"
        project_links = website1_scrape.get_project_links(base_url, response.body)
        for index, link in enumerate(project_links):
            yield scrapy.Request(url=link, callback=self.parse_project, meta={'index': index})
            time.sleep(5)

    def parse_project(self, response):
        project_index = response.meta['index']

        project_url = response.url
        parsed_url = urlparse(response.url)
        base_url = f"{parsed_url.scheme}://{parsed_url.netloc}"
        page_num = parse_qs(parsed_url.query)['currentSelectorPage'][0]

        project_data, project_html = website1_scrape.extract_data(base_url, response.body)
        save_data.save('website1\output', project_data, project_html, page_num, project_index)
```

BeautifulSoup

website1_scrape.py M

website1 > website1_scrape.py > extract_data

```python
from bs4 import BeautifulSoup

def get_project_links(base_url, html):
    soup = BeautifulSoup(html, features="lxml")

    results_div = soup.find('div', id='results')
    data_divs = results_div.find_all('div', class_='data')

    links = []
    for data_div in data_divs:
        links.append(f'{base_url}{data_div.find('div').find('a')['href']}')
    print(links)
    return links

def extract_data(base_url, html):
    data = {}
    soup = BeautifulSoup(html, features="lxml")

    data['title'] = soup.title.string

    description_title_element = soup.find('h3', string='Allgemeine Vorhabenbeschreibung')
    description = description_title_element.find_next_sibling('p').string
    data['description'] = description

    date_title_element = soup.find('h4', string='Datum der Entscheidung')
    data['date'] = date_title_element.find_next_sibling('p').string if not date_title_element else ''

    download_link = soup.find('a', title='Alle Dokumente als ZIP-Datei herunterladen')['href']
    data['download_link'] = download_link

    print(data)
    return data, soup.prettify()
```

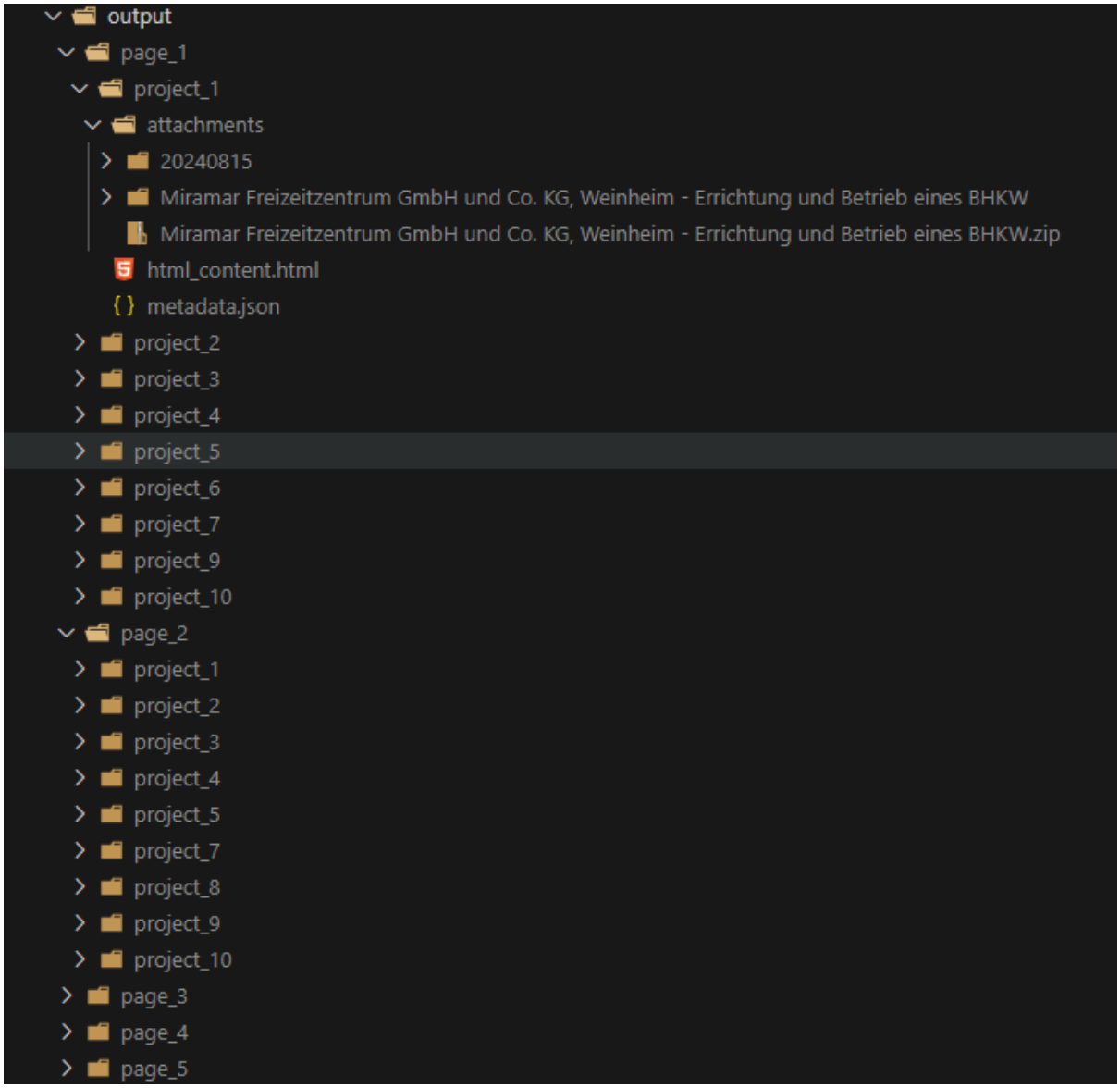
Utility to save metadata and download attachments

save_data.py

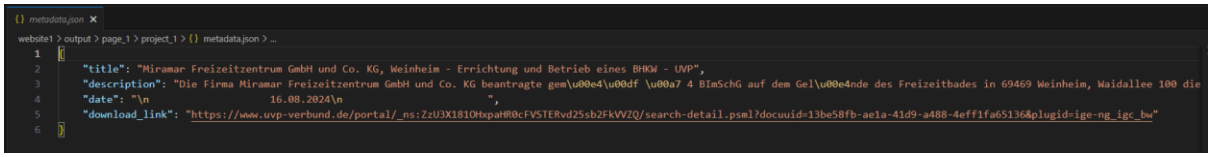
website1 > utils > save_data.py > ...

```python
import os
import json
import urllib.request
import zipfile

def save(output_dir, data, html, page_num, project_index):
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    page_dir = os.path.join(output_dir, f'page_{page_num}')
    project_dir = os.path.join(page_dir, f'project_{project_index + 1}')

    if not os.path.exists(page_dir):
        os.makedirs(page_dir)

    if not os.path.exists(project_dir):
        os.makedirs(project_dir)

    with open(os.path.join(project_dir, 'metadata.json'), 'w') as file:
        json.dump(data, file, indent=4)

    with open(os.path.join(project_dir, 'html_content.html'), 'w') as file:
        file.write(html)

    if(data['download_link']):
        attachment_dir = os.path.join(project_dir, 'attachments')
        if not os.path.exists(attachment_dir):
            os.makedirs(attachment_dir)

        download_file_path = ''
        with urllib.request.urlopen(data['download_link']) as response:
            content_disposition = response.headers.get('Content-Disposition')
            if content_disposition:
                filename = content_disposition.split('filename=')[-1].strip('\"')
                download_file_path = os.path.join(attachment_dir, filename)

                if(len(download_file_path) > 256):
                    extra_characters = len(download_file_path) - 256
                    filename = filename.strip('.zip')[:-extra_characters] + '.zip'
                    download_file_path = os.path.join(attachment_dir, filename)

                with open(download_file_path, 'wb') as file:
                    file.write(response.read())

        if('.zip' in download_file_path):
            os.makedirs(download_file_path.strip('.zip'))
            with zipfile.ZipFile(download_file_path, 'r') as zip_file:
                zip_file.extractall(attachment_dir)
```

Output folder structure



Metadata.json



Observations/Challenges:

After initial inspection of the html behind the web page, I decided to implement the task using scrapy and BeautifulSoup. This website blocks very frequent requests which prompted me to add a delay between requests. Unfortunately, this slows down the complete extraction process. When downloading attachments, some failed initially because the name of the zip file would be greater than 256 characters (character limit for path in windows os), so I had to trim the name of zip file in case the path character count was more than 256.

Website 2

Main.py

main.py

website2 > main.py > main

```python
from SeleniumDriver import SeleniumDriver
from utils import save_data

def main():
    url = r'https://edl.doe.gov.my/discovery?query=+where+category%3d+%2527EIA+Report%2527&category=EIA%20Report&main=Digital'
    selenium_driver = SeleniumDriver()
    selenium_driver.open(url)

    for page in range(1,6):
        selenium_driver.navigate_to_page_number(page)
        projects = selenium_driver.get_projects()

        for index, project in enumerate(projects):
            selenium_driver.navigate_to_project(index)
            data, pretty_html = selenium_driver.extract_data()
            save_data.save('website2/output', data, pretty_html, page, index)
            selenium_driver.open(url)

    selenium_driver.close()


if __name__ == "__main__":
    main()
```

SelenuimDriver

SeleniumDriver.py M

website2 > SeleniumDriver.py > ...

```python
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.support import expected_conditions
from bs4 import BeautifulSoup
from urllib.parse import urlparse, parse_qs

class SeleniumDriver:
    def __init__(self):
        chrome_options = webdriver.ChromeOptions()
        chrome_options.add_experimental_option("detach", True)
        service = webdriver.ChromeService(executable_path=r'website2\chromedriver.exe')
        self._driver = webdriver.Chrome(service=service, options=chrome_options)

    def close(self):
        self._driver.close()

    def open(self, url):
        self._driver.get(url)

    def navigate_to_page_number(self, page_num):
        id = f'content_body_rptPaging_lbPaging_{page_num - 1}'
        WebDriverWait(self._driver, 30).until(expected_conditions.presence_of_element_located((By.ID, id)))
        next_page_a = self._driver.find_element(By.ID, id)
        self._driver.execute_script('arguments[0].click();', next_page_a)

    def navigate_to_project(self, project_index):
        print(f'content_body_rptResult_lbtnOverview2_{project_index}')
        element = self._driver.find_element(By.ID, f'content_body_rptResult_lbtnOverview2_{project_index}')
        self._driver.execute_script('arguments[0].click();', element)

    def get_projects(self):
        project_elements = self._driver.find_elements(By.CLASS_NAME, 'lbltitle_')
        return project_elements

    def get_page_html(self):
        return self._driver.page_source

    def _get_data_by_id(self, soup, id):
        return soup.find('textarea', id=id).string

    def extract_data(self):
        WebDriverWait(self._driver, 30).until(expected_conditions.presence_of_element_located((By.ID, 'content_body_txtTitle')))

        data = {}
        soup = BeautifulSoup(self._driver.page_source, features="lxml")

        data['title'] = self._get_data_by_id(soup, 'content_body_txtTitle')
        data['category'] = self._get_data_by_id(soup, 'content_body_txtsubcategory')
        data['release_date'] = self._get_data_by_id(soup, 'content_body_txtreleased_eia')
        data['consultant'] = self._get_data_by_id(soup, 'content_body_txtconsultant_eia')
        data['project_developer'] = self._get_data_by_id(soup, 'content_body_txtconsultant_eia')
        data['location'] = self._get_data_by_id(soup, 'content_body_txtlocation_eia')
        data['post_date'] = soup.find('input', id='content_body_txtdatepost_eia')['value']


        # https://edl.doe.gov.my/images/items/42681/attachment/20241908131633413.pdf
        page_url = self._driver.current_url
        parsed_url = urlparse(page_url)
        download_base_url = f"{parsed_url.scheme}://{parsed_url.netloc}/images/items/"
        bib_number = parse_qs(parsed_url.query)['bibnumber'][0]

        download_links = []
        download_a_tags = soup.find_all('a', class_='lbtn')
        for tag in download_a_tags:
            download_links.append(f'{download_base_url}{bib_number}/attachment/{tag.string}')
        data['download_links'] = download_links

        print(data)
        return data, soup.prettify()
```

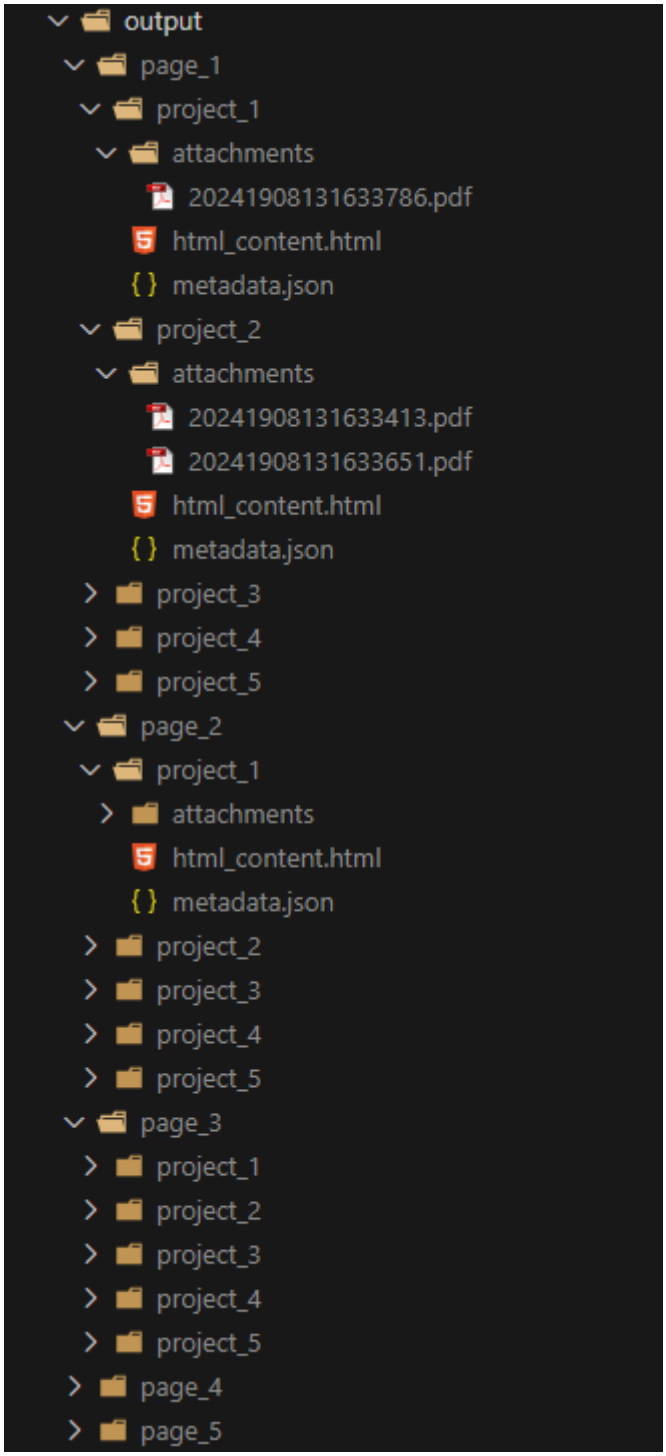
Utility to save metadata and download attachments

save_data.py

website2 > utils > save_data.py > save

```python
import os
import json
import urllib.request
import zipfile

def save(output_dir, data, html, page_num, project_index):
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    page_dir = os.path.join(output_dir, f'page_{page_num}')
    project_dir = os.path.join(page_dir, f'project_{project_index + 1}')

    if not os.path.exists(page_dir):
        os.makedirs(page_dir)

    if not os.path.exists(project_dir):
        os.makedirs(project_dir)

    with open(os.path.join(project_dir, 'metadata.json'), 'w') as file:
        json.dump(data, file, indent=4)

    with open(os.path.join(project_dir, 'html_content.html'), 'w') as file:
        file.write(html)

    if(data['download_links']):
        attachment_dir = os.path.join(project_dir, 'attachments')
        if not os.path.exists(attachment_dir):
            os.makedirs(attachment_dir)

        for download_link in data['download_links']:
            download_file_path = ''
            with urllib.request.urlopen(download_link) as response:
                filename = download_link.split('/')[-1]
                download_file_path = os.path.join(attachment_dir, filename)

                with open(download_file_path, 'wb') as file:
                    file.write(response.read())
```

Output folder structure



Metadata.json

Observations/Challenges:

On inspection on the html code behind website 2, I realized it is a ASP.NET web page. This made navigation and actions difficult with scrapy, so I decided to use Selenium for this. Downloading attachments was a challenge too as the download link was not directly available.