Midterm Project Report

Advanced Computer Programming

Student Name : M. Fadhlan A. Harashta
Student ID : 112021222
Teacher : DINH-TRUNG VU

2025-04

Chapter 1 Introduction

1.1 Github

1) **Personal Github Account**: fadhlanharashta
2) **Group Project Repository**: https://github.com/fadhlanharashta/ACP---Group-3

1.2 Overview

In my mid term project, I develop a web scraper using Scrapy and Pyhthon web crawling framework, I use several useful libraries which includes:

- CrawlSpider
- Regular Expression
- CSS and XPath Selectors
- Conditional Logic and String Manipulation

In this project I use a few libraries, Scrapy which is the main scrapy module, LinkExtractor which helps extract links that match a patterm, and CrawlSpider to lets the code to follow links based on rules.

While some part of the code works as intended, such as the code abilitry to capture the repositories link, title, and number of commit and language , it is somehow unable to get the last updated.

The captured data then put on an XML file and also to make it easier to check, I also make a JSON file.

Chapter 2 Implementation

2.1 Class 1: GithubSpider

This class inherits form CrawlSpider, a specialized spider in Scrappy used for crawling websites using Rules. It is designed to scrape Github repository data froma specific user.

2.1.1 Fields

Field	Description
name	Unique name to identify spider: “github_spider”
allowed_domains	A list that limits the spider crawling to github: “github.com”
Start_urls	Starting URL for the spider’s to begin crawling “https://github.com/fadhlanharashta?tab=repositories"
rules	Some rules that tells the spider how to follow links and which callback to use. In this case, it follows repository links and cals “parse_repo”

2.1.2 Methods

Parse Repo

The main callback that extract data from each repository page. It handles both empty and not empty repositories and collect all the needed datas such as link, name, number of commit, language, and last update.

```
    def parse_repo(self, response):
        url = response.url
        repo_name = url.rstrip('/').split('/')[-1]

        # About section
        about = response.css('p.f4.my-3::text, div.BorderGrid p.f4::text').get()
        about = about.strip() if about else None

        # Check for empty repo
        is_empty = response.css('div.Box.mt-3 h3::text').re_first(r'This repository is (.*)')

        if is_empty:
            yield {
                "url": url,
                "about": about if about else repo_name,
                "last_updated": None,
                "languages": None,
                "number_of_commits": None,
            }
        else:
            last_updated = response.css('div[data-testid="latest-commit-details"] relative-time::attr(datetime)').get()
            if not last_updated:
                last_updated = response.xpath('//relative-time/@datetime').get()

            #Language
            languages = response.css('ul.list-style-none .d-inline .color-fg-default::text').getall()
            if not languages:
                languages = response.css('.language-color + span::text').getall()
            languages = [lang.strip() for lang in languages if lang.strip()]
            languages_str = ", ".join(languages) if languages else None
            # Commit
            commit_text = response.css('div[data-component="text"] span.fgcolor-default::text').re_first(r'(\d+)\s+commits?')
            number_of_commits = int(commit_text) if commit_text else None



            yield {
                "url": url,
                "about": about if about else repo_name,
                "last_updated": last_updated,
                "languages": languages_str,
                "number_of_commits": number_of_commits,
            }

```

2.1.3 Functions

- **Extract Repositories**

  Extract current repository and then parse the repositories name from the URL string. The spider only crawls github.com and starts from repositories tab. It has a rule: the program will extract links for githubname/reponame but no more slashes. Since it only follow links to individual repositories from the profile's repositories page, it only take from anchor tags (<a>) that match the given CSS selector. And then each link call the parse repo method. Lastly it wont follow links from the individual repo page.

```
class GithubSpider(CrawlSpider):
    name = "github_spider"
    allowed_domains = ["github.com"]
    start_urls = ["https://github.com/fadhlanharashta?tab=repositories"]

    #rules
    rules = (
        Rule(
            LinkExtractor(
                allow=r'/fadhlanharashta/[^/]+$',
                restrict_css='a[itemprop="name codeRepository"]'
            ),
            callback='parse_repo',
            follow=False
        ),
    )


    def parse_repo(self, response):
        url = response.url
        repo_name = url.rstrip('/').split('/')[-1]

```

- **About**

  Uses CSS selector to get the repository description if available. It extract the repository description.

```
        # About section
        about = response.css('p.f4.my-3::text, div.BorderGrid p.f4::text').get()
        about = about.strip() if about else None
```

- **Empty Repo check**

  Uses regex to check if the repository is empty or not. If repo is empty yield as follows:

```
        # Check for empty repo
        is_empty = response.css('div.Box.mt-3 h3::text').re_first(r'This repository is (.*)')

        if is_empty:
            yield {
                "url": url,
                "about": about if about else repo_name,
                "last_updated": None,
                "languages": None,
                "number_of_commits": None,
            }
        else:
```

- **Last Update**

  Scrapes the <relative-time> tag to get the latest update timestamp.

```
            last_updated = response.css('div[data-testid="latest-commit-details"] relative-time::attr(datetime)').get()
            if not last_updated:
                last_updated = response.xpath('//relative-time/@datetime').get()
```

- **Language**

  Extract programing languages used in the repository using css selector.

```
            #Language
            languages = response.css('ul.list-style-none .d-inline .color-fg-default::text').getall()
            if not languages:
                languages = response.css('.language-color + span::text').getall()
            languages = [lang.strip() for lang in languages if lang.strip()]
            languages_str = ", ".join(languages) if languages else None
```

- **Number of Commit**

  Uses CSS Selector to find and parse the number of commits.

```

            if commit_text:
                commit_count = commit_text.strip().replace(",", "")
                if "Commit" not in commit_count:
                    commit_count += " Commits"
                commits = commit_count
            else:
                commits = "None"
```

- **Yield**

  Each yield returns a dictionary for each repository. The data of which we want to crawl

```
            yield {
                "url": url,
                "about": about if about else repo_name,
                "last_updated": last_updated,
                "languages": languages_str,
                "number_of_commits": number_of_commits,
            }
```

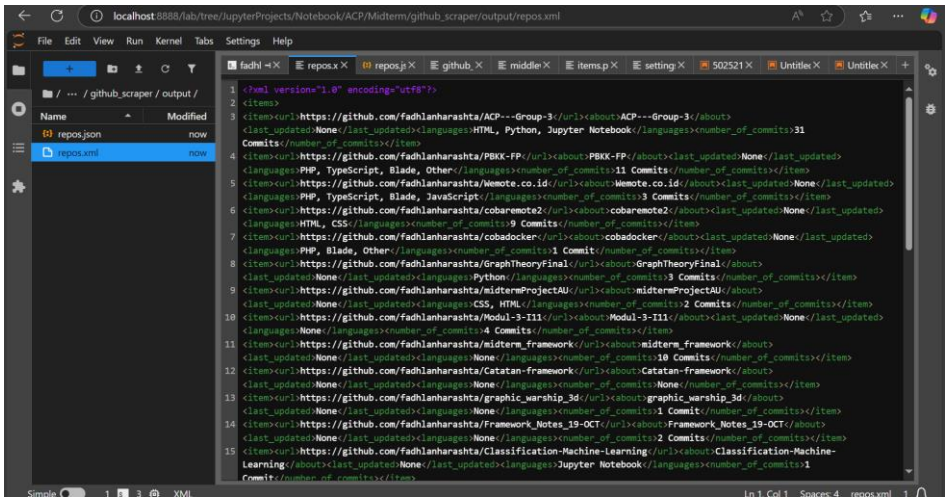
2.2 Class 2: Setting

```
BOT_NAME = "github_scraper"

SPIDER_MODULES = ["github_scraper.spiders"]
NEWSPIDER_MODULE = "github_scraper.spiders"

FEEDS = {
    "output/repos.xml": {
        "format": "xml",
        "encoding": "utf8",
        "overwrite": True,
    }
}
# Crawl responsibly by identifying yourself (and your w
#USER_AGENT = "github_scraper (+http://www.yourdomain.c

# Obey robots.txt rules
USER_AGENT = "Mozilla/5.0"
ROBOTSTXT_OBEY = False
```

Name	Description
BOT_NAME	Identifies the scappy bot, used internally.
SPIDER_MODULE	Module path for spider definition.
NEWSPIDER_MODULE	Default path for spider created via command line
FEEDS	Specifies export format and file path XML file exported to output/repos.xml
USER_AGENT	Overrides default user-agent string to mimic a real browser.
ROBOTSTXT_OBEY	Sets to false to ignore robot.txt riles and allow full crawling
TWISTED_REACTOR	Specifies the event loop reactor for asynchronous processing.
FEED_EXPORT_ENCODING	Sets UTF-8 Encoding for exported file.

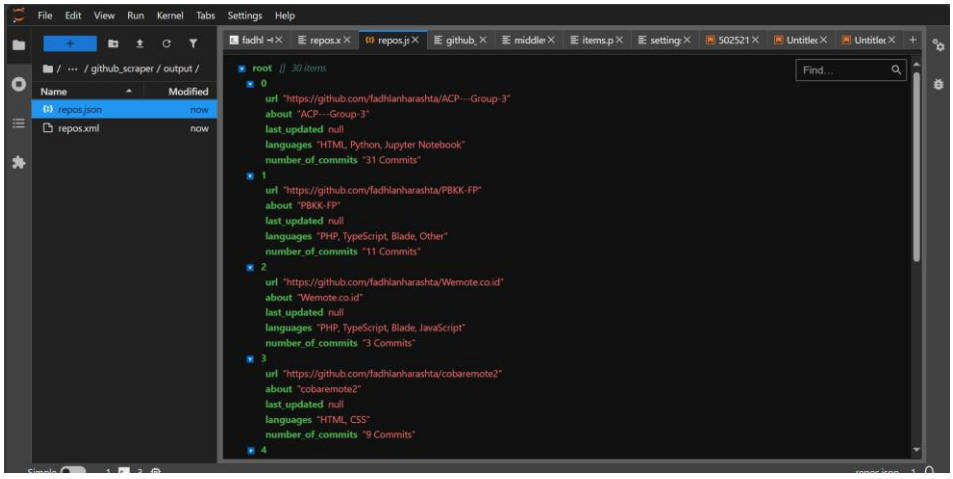
Chapter 3 Results

3.1 Result 1: XML File



As per requirement the result of the scrapping is put on XML file. This is the result I get from the scrapping.

3.2 Result 2: Json File



To make the debugging process much easier, I make a Json file since its easier to read when debugging.

Chapter 4 Conclusions

While I successfully crawl some data inside my repositories, there are some data that I stil unable to get. Data like last updated still unable to be crawled. The problem is that my code is unable to recognize the last updated date from the HTML. While I have try to inspect the HTML manually to find the number of commit and the last update detail, I still unable to get the data and put it into the XML file. As a result, the data im able to put on the XML file which is not none is links, about, number of commit, and languages, while the last update and number of commit remain none.