Assignment 1

Repo address: <https://github.com/krystenng/SMU-IS459-assignment-.git>

**Navigation to codes:**

hardwarezone/

scrapy.cfg <-- Configuration file

result.json <-- The results from the data crawled

hardwarezone/

items.py <-- Model of the item to scrap

middlewares.py <-- Scrapy processing hooks

pipelines.py <-- What to do with the scraped item

settings.py <-- Settings file

spiders/ <-- Directory of the spider

\_\_init\_\_.py

spider.py <--- Containing xpath codes to extract the data

**Creating spider in the terminal using visual studio code:**

>>> scrapy startproject hardwarezone

>>> cd hardwarezone

>>> scrapy genspider spider forums.hardwarezone.com.sg/forums/pc-gaming.382/   
# connecting to the webpage

>>> scrapy crawl spider <-- to run the spider

>>> scrapy runspider spider.py -o result.json <-- in order to get the output (data) crawled from

the website

**Using ubuntu (20.04) to dump the records to mongoDB:**

>>> clone your git repo to ubuntu

>>> cd SMU-IS459(assignment)

>>> cd Scrapy

>>> cd hardwarezone

>>> scrapy crawl spider <-- ensure that the pipelines.py and settings.py have the codes to

connect to the localhost port 27017

>>> used robo 3T to check whether the records have been dumped into mongoDB

Explanation of code (HighLighted):

Spider.py

import scrapy

class SpiderSpider(scrapy.Spider):

    name = 'spider'

    allowed\_domains = ['forums.hardwarezone.com.sg/forums/pc-gaming.382']

    start\_urls = ['http://forums.hardwarezone.com.sg/forums/pc-gaming.382//']

    base\_url = 'https://forums.hardwarezone.com.sg' #to access the threads next page

    def parse(self, response):

        all\_the\_threads = response.xpath('//div[starts-with(@class, "structItem structItem--thread js-inlineModContainer js-threadListItem-")]') #get all the threads first

        for thread in all\_the\_threads:

            thread\_url = self.base\_url + thread.xpath('.//div[@class="structItem-title"]/a/@href').extract\_first()

            yield scrapy.Request(thread\_url, callback=self.parse\_thread,dont\_filter=True) #to access the different threads on the page

        next\_page\_partial\_url = response.xpath('//li[@class="pageNav-page pageNav-page--later"]/a/@href').extract\_first()

        next\_page\_url = self.base\_url + next\_page\_partial\_url

        yield scrapy.Request(next\_page\_url, callback=self.parse, dont\_filter=True) #to access the threads in all pages

    def parse\_thread(self, response):

        next\_page\_partial\_url = response.xpath('//li[@class="pageNav-page pageNav-page--later"]/a/@href').extract\_first()

        if next\_page\_partial\_url != None:

            threads = response.xpath('//article[@class="message message--post js-post js-inlineModContainer  "]') #to check if next page exist for the individual thread

            for thread in threads:

                name = response.xpath('//div/h1/text()').extract\_first()

                content\_list = thread.xpath('.//div[@class="bbWrapper"]/text()').extract()

                content = ''.join(content\_list) #to join the content of one thread

                author = thread.xpath('.//h4/a/text()').extract\_first()

                if author == None:

                    author = thread.xpath('.//h4/a/span/text()').extract\_first() #this is for the first author on the page that has different xpath

                yield {

                    'Name': name, #topic of the poster

                    'Author': author, #author of the poster

                    'content': content #content of the poster

                }

            next\_page\_url = self.base\_url + next\_page\_partial\_url

            yield scrapy.Request(next\_page\_url, callback=self.parse\_thread, dont\_filter=True)

        else:

            threads = response.xpath('//article[@class="message message--post js-post js-inlineModContainer  "]')

            for thread in threads:

                name = response.xpath('//div/h1/text()').extract\_first()

                content\_list = thread.xpath('.//div[@class="bbWrapper"]/text()').extract()

                content = ''.join(content\_list)

                # print(content)

                author = thread.xpath('.//h4/a/text()').extract\_first()

                if author == None:

                    author = thread.xpath('.//h4/a/span/text()').extract\_first()

                # print(author)

                yield {

                    'Name': name,

                    'Author': author,

                    'content': content

                }

Validity of records going in MongoDB:

>>> Type “mongo” into the ubuntu terminal and can check whether the database is created

Calendar

Description automatically generated

>>> Check if the records are in MongoDB using Robo 3T

Graphical user interface, text

Description automatically generated