

## Описание работы программы Mini readability

Программа состоит из 3 модулей:

- news\_parser.py
- news\_templates.py
- parsing\_scheme.py

Основным модулем и утилитой является файл news\_parser.py

Запускается он с обязательным аргументом *url* следующим образом:

*python news\_parser.py url*, где url это ссылка на статью.

news\_templates.py содержит шаблоны обработки новостных ресурсов.

parsing\_scheme.py содержит настройки форматирования текста

## Подробнее о каждом модуле

### news\_parser.py

Модуль содержит 3 класса

*ContentParser* - основной класс, получает, преобразует, сохраняет.

*TextTransform* - обрабатывает текст

*LinkCleaner* - обрабатывает ссылки

При запуске программы вызывается класс *ContentParser* и основной его метод *parse\_content*.

У класса есть 3 метода:

*get\_content* - получает контент сайта с помощью библиотеки *requests* и *bs4* и затем вызывается класс *NewsTemplates* который парсит содержимое в соответствии с шаблоном(как, описано ниже).

*parse\_content* - парсит ссылки с помощью класса *LinkCleaner*, а текст с помощью *TextTransform*

*save\_content* - сохраняет текст в файл

Класс *TextTransform* содержит базовый метод *base\_transform* который вызывает остальные методы, такие как *text\_upper*(перевод буквы в верхний регистр), *line\_break*(перенос строк), *ident\_line*(отступ строки)

Класс *LinkCleaner* содержит один метод обработки ссылок *link\_clean*.

Предполагается что в классе *ContentParser* новые методы добавляться не будут, тогда как классы *TextTransform* и *LinkCleaner* расширяемы, и при необходимости нового сценария обработки текста или ссылок можно будет добавить новых функций.

### Листинг news\_parser.py

```
import requests
```

```
import textwrap
```

```
import sys
```

```
import re
```

```
from bs4 import BeautifulSoup
```

```
from urllib.parse import urlparse
from parsing_scheme import *
from news_templates import NewsTemplates
from fake_useragent import UserAgent
```

```
HEADER = {'User-Agent': str(UserAgent().chrome)}
```

```
class LinkCleaner:
```

```
    def __init__(self, element):
        self.element = element
```

```
    def link_clean(self, position=LINK_PARSER):
```

```
        if self.element.a:
            links = self.element.findAll('a')
            for link in links:
                try:
                    index = self.element.index(link)
                    if position == 'after':
                        self.element.contents[index].replaceWith('{ } {}'.format(link.string, link["href"]))
                    elif position == 'before':
                        self.element.contents[index].replaceWith('[] {}'.format(link["href"], link.string))
                except ValueError:
                    pass
            return self.element
```

```
class TextTransform:
```

```
    def __init__(self, text, tag_name=None):
        self.text = text
        self.tag_name = tag_name
```

```
    def base_transform(self, ident=IDENT):
```

```
        if ident:
            self.ident_line()
            self.line_break()
        if self.tag_name in ('h1', 'h2', 'h3'):
            self.text_upper()
        return self.text
```

```
    def text_upper(self):
```

```
        self.text = self.text.upper()
```

```
    def line_break(self, width=LINE_BREAK_WIDTH):
```

```
        wrap = textwrap.wrap(self.text, width=width)
        self.text = '\n'.join(wrap) + '\n'
```

```
    def ident_line(self):
```

```
if self.tag_name not in ('h1', 'h2', 'h3'):
    self.text = '\t' + self.text
```

**class** ContentParser:

```
regex_link = re.compile('(/[S]*)/([a-z\d_]*)')
```

```
def __init__(self, url, regex_link=regex_link):
    self.url = url
    self.domain = urlparse(url).hostname
    self.path = re.search(regex_link, urlparse(url).path).group(1).replace('/', '\\')
    self.file_name = re.search(regex_link, urlparse(url).path).group(2) or 'index'
```

```
def get_content(self):
    response = requests.get(self.url, headers=HEADER).text
    soup = BeautifulSoup(response, 'html.parser')
    content = NewsTemplates(self.domain, soup).get_content_from_template()
    return content
```

```
def parse_content(self, link=LINK):
    clean_content = []
    for element in self.get_content():
        if link:
            element = LinkCleaner(element).link_clean()
            text = TextTransform(element.text, element.name).base_transform()
            clean_content.append(text)
    self.save_content(clean_content)
```

```
def save_content(self, content):
    path = '{}{}'.format(CUR_DIR, self.path)
    file = '{}{}'.format(self.file_name, '.txt')
    if not os.path.exists(path + file):
        try:
            os.makedirs(path)
        except FileExistsError:
            pass
    with open(path + file, 'w+', encoding='utf-8') as out:
        for line in content:
            out.write(line + '\n')
```

```
if __name__ == '__main__':
    try:
        site = ContentParser(sys.argv[1])
        site.parse_content()
    except IndexError:
        print('Укажите обязательный аргумент в виде URL')
```

## news\_templates.py

Модуль содержит один класс *NewsTemplates*, в котором формируется шаблон обработки web страницы в соответствии с сайтом. Данная необходимость возникла из за уникального контекста на каждом ресурсе. На одном ресурсе параграф с текстом может быть обрамлен в *<div>*, на другом в *<p>*, а так же могут быть разные классы.

Класс очень простой, в базовом методе *get\_content\_from\_template* есть условия проверяющие доменное имя из *url*, если в условии присутствует ресурс на котором располагается статья, то вызывается его метод, если нет то вызывается базовый метод. Базовый метод собирает много мусора. Для точной очистки содержимого нужно создавать шаблон ресурса.

## Листинг news\_templates.py

```
import re
```

```
class NewsTemplates:
```

```
    def __init__(self, domain, soup):
        self.domain = domain
        self.soup = soup
```

```
    def get_content_from_template(self):
        if self.domain == 'ria.ru':
            return self.ria_news()
        elif self.domain == 'news.rambler.ru':
            return self.rambler_news()
        elif self.domain == 'www.gazeta.ru':
            return self.gazeta_news()
        elif self.domain == 'lenta.ru':
            return self.lenta_news()
        else:
            return self.base_news()
```

```
    def ria_news(self):
        content = self.soup.find_all(['div', 'h1', 'h2', 'h3'], {'class': re.compile('article__(text|title)')})
        return content
```

```
    def rambler_news(self):
        content = self.soup.find_all(['div', 'h1', 'h2', 'h3'], {'class': re.compile('__(paragraph|title)')})
        return content
```

```
    def gazeta_news(self):
        try:
            content = self.soup.find(['article'], {'class': re.compile('article-text')})
            content = content.find_all(['p', 'h1', 'h2', 'h3'])
```

```

except AttributeError:
    content = self.base_news()
return content

def lenta_news(self):
    content = self.soup.find(['div'], {'class': re.compile('topic__content')})
    content = content.find_all(['p', 'h1'])
    return content

def base_news(self):
    content = self.soup.find_all(['p', 'h1', 'h2', 'h3'])
    return content

```

## parsing\_scheme.py

Настройка форматирования контента, оформленного в виде глобальных переменных которые импортируются главным модулем *news\_parser.py*.

Можно включить форматирование ссылок, можно отключить(их вообще в тексте не будет), можно указать другую директорию сохранения файлов или указать ширину строки для переноса и пр.

При необходимости всегда можно что то добавить.

## Листинг parsing\_scheme.py

```

import os

# Парсить ли ссылки, принимает 2 значения 0 -выкл 1 - вкл
LINK = 1

# Позиционирование ссылки. После текста ссылки или до. Может принимать значения 'after' или 'befor'
LINK_PARSER = 'after'

# Ширина строки для переноса по словам. Принимает возможное количество символов в строке.
LINE_BREAK_WIDTH = 80

# Домашняя директория для сохранения файла.
CUR_DIR = os.getcwd()

# Отступ у параграфа, принимает 2 значения 0 -выкл 1 - вкл
IDENT = 0

```

**Рекомендации для дальнейшего развития системы и прочее комментарии от автора.**

Не знаю правильно ли я понял про утилиту командной строки, но есть мысль, возможно так и надо было сделать, хотел завернуть программу в скрипт устанавливающий virtualenv и все зависимости и вызывающий базовый метод программы, для того чтобы простой пользователь не парился с установкой и настройкой проекта. Немного не успел.

По рекомендациям.

Добавлять больше шаблонов для обработки ресурсов, больше настроек для обработки текста. Возможно сделать графический интерфейс, а лучше веб-приложение. Добавить автоматическое открытие готового файла после выполнения программы. Добавить больше валидаций для приема данных. Подумать над тем как сделать супер универсальный алгоритм для всех ресурсов(я пока не смог). Да и совершенству и фантазиям нет предела...