Санкт-Петербургский Национальный Исследовательский Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №4

“Исследование протоколов, форматов обмена информацией и языков разметки документов”

По дисциплине

“Информатика”

Вариант 14

Выполнил:

Студент группы P3117

Пономарёв М. И.

Преподаватель:

Машина Е. А.

Санкт-Петербург
2022

Оглавление

Задание

4.1 Порядок выполнения работы

1. Определить номер варианта как остаток деления на 36 порядкового номера в списке группы в ISU. В случае, если в данный день недели нет занятий, то увеличить номер варианта на восемь.
2. Изучить форму Бэкуса-Наура.
3. Изучить особенности языков разметки/форматов JSON, YAML, XML.
4. Понять устройство страницы с расписанием для своей группы: http://itmo.ru/ru/schedule/0/P3110/schedule.htm
5. Исходя из структуры расписания конкретного дня, сформировать файл с расписанием в формате, указанном в задании в качестве исходного. При этом необходимо, чтобы в выбранном дне было не менее двух занятий (можно использовать своё персональное). В случае, если в данный день недели нет таких занятий, то увеличить номер варианта ещё на восемь.
6. Обязательное задание (позволяет набрать до 65 процентов от максимального числа баллов БаРС за данную лабораторную): написать программу на языке Python 3.x, которая бы осуществляла парсинг и конвертацию исходного файла в новый.
7. Нельзя использовать готовые библиотеки, в том числе регулярные выражения в Python и библиотеки для загрузки XML-файлов.
8. Дополнительное задание №1 (позволяет набрать +10 процентов от максимального числа баллов БаРС за данную лабораторную).
    a) Найти готовые библиотеки, осуществляющие аналогичный парсинг и конвертацию файлов.
    b) Переписать исходный код, применив найденные библиотеки. Регулярные выражения также нельзя использовать.
    c) Сравнить полученные результаты и объяснить их сходство/различие.
9. Дополнительное задание №2 (позволяет набрать +10 процентов от максимального числа баллов БаРС за данную лабораторную).
    a) Переписать исходный код, добавив в него использование регулярных выражений.

b) Сравнить полученные результаты и объяснить их сходство/различие.

10. Дополнительное задание №3 (позволяет набрать +10 процентов от максимального числа баллов БаРС за данную лабораторную).
    a) Используя свою исходную программу из обязательного задания, программу из дополнительного задания №1 и программу из дополнительного задания №2, сравнить стократное время выполнения парсинга + конвертации в цикле.
    b) Проанализировать полученные результаты и объяснить их сходство/различие.
11. Дополнительное задание №4 (позволяет набрать +5 процентов от максимального числа баллов БаРС за данную лабораторную).
    c) Переписать исходную программу, чтобы она осуществляла парсинг и конвертацию исходного файла в любой другой формат (кроме JSON, YAML, XML, HTML): PROTOBUF, TSV, CSV, WML и т.п.
    d) Проанализировать полученные результаты, объяснить особенности использования формата.

Основные этапы вычисления

getHTMLwithBs4.py

```python
from bs4 import BeautifulSoup
import requests


def get_html_with_bs4(url) -> str:
    html_page = requests.get(url)
    soup = BeautifulSoup(html_page.text, 'html.parser')
    schedule = soup.find("table", {"id": "2day"})
    schedule.prettify()
    return str(schedule)
```

HtmlToJson.py

```python
import html_to_json


def html_to_json_dict(html_string) -> dict:
    return html_to_json.convert(html_string)
```

JsonToYaml.py

```
def json_to_yaml(json_dict) -> str:
    output_string = "---\n" + parse_dict(json_dict) + "\n---"
    return output_string


def add_spacing(some_str) -> str:
    return "  " + some_str.replace("\n", "\n  ")


def parse_string(some_string) -> str:
    try:
        some_string = str(int(some_string))
    except ValueError:
        some_string = '"' + some_string + '"'
    if some_string.count("\n"):
        some_string = ">\n" + add_spacing(some_string)
    return some_string


def parse_list(main_list) -> str:
    output_string = ""
    for element in main_list:
        # output_string += "- "
        if type(element) == list:
            output_string += add_spacing(parse_list(element))
        elif type(element) == dict:
            output_string += add_spacing(parse_dict(element))
        else:
            output_string += "- " + parse_string(str(element))
        output_string += "\n"

    output_string = output_string.rstrip("\n").rstrip()
    return output_string


def parse_dict(main_dict) -> str:
    output_string = ""
    for key in main_dict.keys():
        # output_string += "- " + str(key) + ": "
        element = main_dict[key]
        if type(element) == list:
            output_string += "- " + str(key) + ":\n" +
add_spacing(parse_list(element))
        elif type(element) == dict:
            output_string += "- " + str(key) + ":\n" +
add_spacing(parse_dict(element))
        else:
            output_string += str(key) + ": " + parse_string(str(element))
        output_string += "\n"

    output_string = output_string.rstrip("\n").rstrip()
    return output_string
```

JsonToYamlLibrary.py

```
import yaml
import json


def json_to_yaml_with_library(json_str) -> str:
    return yaml.dump(json.loads(json_str), allow_unicode=True)
```

JsonToYamlRegex.py

```
import re


def json_to_yaml(json_dict) -> str:
    output_string = "---\n" + parse_dict(json_dict) + "\n---"
    return output_string


def add_spacing(some_str) -> str:
    return "  " + some_str.replace("\n", "\n  ")


def parse_string(some_string) -> str:
    if re.match("/\[[0-9]+\]/", some_string):
        some_string = str(int(some_string))
    else:
        some_string = '"' + some_string + '"'
    if some_string.count("\n"):
        some_string = ">\n" + add_spacing(some_string)
    return some_string


def parse_list(main_list) -> str:
    output_string = ""
    for element in main_list:
        # output_string += "- "
        if type(element) == list:
            output_string += add_spacing(parse_list(element))
        elif type(element) == dict:
            output_string += add_spacing(parse_dict(element))
        else:
            output_string += "- " + parse_string(str(element))
        output_string += "\n"

    output_string = output_string.rstrip("\n").rstrip()
    return output_string


def parse_dict(main_dict) -> str:
    output_string = ""
    for key in main_dict.keys():
        # output_string += "- " + str(key) + ": "
        element = main_dict[key]
        if type(element) == list:
            output_string += "- " + str(key) + ":\n" +
add_spacing(parse_list(element))
        elif type(element) == dict:
            output_string += "- " + str(key) + ":\n" +
add_spacing(parse_dict(element))
        else:
            output_string += str(key) + ": " + parse_string(str(element))
        output_string += "\n"

    output_string = output_string.rstrip("\n").rstrip()
    return output_string
```

main.py

```
from lab4.HtmlToJson import html_to_json_dict
from lab4.getHTMLwithBs4 import get_html_with_bs4
from lab4.JsonToYaml import json_to_yaml
from lab4.JsonToYamlLibrary import json_to_yaml_with_library

url = "https://itmo.ru/ru/schedule/0/P3117/schedule.htm"
data_html =
get_html_with_bs4("https://itmo.ru/ru/schedule/0/P3117/schedule.htm").replace
('"', "")
data_dict = html_to_json_dict(data_html)
data_str = str(data_dict).replace("'", '"')
with open(r"result/data.html", "w", encoding="utf-8") as html_file:
    html_file.write(data_html)
with open(r"result/data.json", "w", encoding="utf-8") as json_file:
    json_file.write(data_str)
with open(r"result/data.yaml", "w", encoding="utf-8") as yaml_file:
    yaml_file.write(json_to_yaml(data_dict))
with open(r"result/data_with_library.yaml", "w", encoding="utf-8") as
yaml_file:
    yaml_file.write(json_to_yaml_with_library(data_str))
```

compare.py

```
from lab4.HtmlToJson import html_to_json_dict
from lab4.getHTMLwithBs4 import get_html_with_bs4
from lab4.JsonToYaml import json_to_yaml
from lab4.JsonToYamlLibrary import json_to_yaml_with_library
import time


def run_my_yaml_parse():
    json_to_yaml(data_dict)


def run_library_yaml_parse():
    json_to_yaml_with_library(data_str)


def get_time(n, func) -> float:
    start_time = time.time()
    for i in range(n):
        func()
    return float(time.time() - start_time)


url = "https://itmo.ru/ru/schedule/0/P3117/schedule.htm"
data_html =
get_html_with_bs4("https://itmo.ru/ru/schedule/0/P3117/schedule.htm").replace
('"', "")
data_dict = html_to_json_dict(data_html)
data_str = str(data_dict).replace("'", '"')

time1, time2 = get_time(100, run_library_yaml_parse), get_time(100,
run_my_yaml_parse)
print(f"Парсинг через библиотеку: {time1} seconds")
print(f"Парсинг написанной функцией: {time2} seconds")
print(
    f"Разница составляет ~{int(max(time1 / time2, time2 / time1) * 100)}% или
{max(time1, time2) - min(time1, time2)} секунд")
```

Результат выполнения программы compare.py:

Парсинг через библиотеку: 4.575491189956665 seconds

Парсинг написанной функцией: 0.23781204223632812 seconds

Разница составляет ~1923% или 4.337679147720337 секунд

Результаты работы main.py содержаться в папке results/

Вывод

Во время выполнения работы я использовал библиотеки json, requests, bs4, yaml для перевода файлов между заданными форматами. Наглядно заметил читаемость и удобство формата yaml в сравнении с json.