2024-09-17

Обзор библиотек для сбора и анализа веб-данных средствами Python

Beautiful Soup 4

Документация: Beautiful Soup Documentation — Beautiful Soup 4.12.0 documentation

- предназначена для парсинга и обработки HTML- и XML-иерархии (дерева)
- забирает все тэги и позволяет работать с ними при помощи нативного Python-синтаксикса
- удобная навигация по дереву (как для вертикальных связей `родитель -> ребенок`, так и для горизонтальных связей `сиблинг <-> сиблинг`)
- надежна и достаточно стабильна, «ломающих» изменений давно не вносилось, срок поддержки длительный
- автоматически старается привести все к UTF-8, что бывает полезно в случае с не самыми современными сайтами
- !НО — изначально задумана, разработана и «заточена» под статические веб-ресурсы или веб-ресурсы с рендрингом на стороне сервера; не умеет выполнять JavaScript

Какие проекты используют BS4

(взято отсюда: Beautiful Soup: We called him Tortoise because he taught us.)

- "Movable Type", a work of digital art on display in the lobby of the New York Times building, uses Beautiful Soup to scrape news feeds.
- Jiabao Lin's DXY-COVID-19-Crawler uses Beautiful Soup to scrape a Chinese medical site for information about COVID-19, making it easier for researchers to track the spread of the virus. (Source: "How open source software is fighting COVID-19")
- Reddit uses Beautiful Soup to parse a page that's been linked to and find a representative image.
- Alexander Harrowell uses Beautiful Soup to track the business activities of an arms merchant.
- The developers of Python itself used Beautiful Soup to migrate the Python bug tracker from Sourceforge to Roundup.
- The Lawrence Journal-World uses Beautiful Soup to gather statewide election results.
- The NOAA's Forecast Applications Branch uses Beautiful Soup in TopoGrabber, a script for downloading "high resolution USGS datasets."

Почему «суп»? — Tag soup - Wikipedia

urllib3

Доки: https://urllib3.readthedocs.io/en/stable/

- одна из фундаментальных библиотек для взаимодействия с сайтами при помощи Python (как следствие — от нее зависят почти все остальные подобные пакеты)
- не связана с включенным в стандартную библиотеку Python модулем `urllib` (https://docs.python.org/3/library/urllib.html), но задумана и де-факто является расширением его функционала
- дает удобные функции для взаимодействия с протоколом HTTP/S (TSL/SSL)
- поддерживает загрузку (на сервер) и выгрузку (с сервера) файлов
- поддерживает переадресацию (редиректы) и нестандартные коды ответа севера (а 418 I'm a teapot - HTTP | MDN такое умеет обрабатывать и подавно)
- поддерживает прокси
- !НО — многое нужно делать «руками», т.к. библиотека по меркам экосистемы Python «низкоуровневая» (не в буквальном смысле!): не так много готовых методов, для решения одних и тех же задач придется либо копировать чьи-то готовые примеры, либо каждый раз писать достаточно много кода

requests

Доки: https://requests.readthedocs.io/en/latest/

- `urllib3` с человеческим лицом: предоставляет удобные методы для быстрого взаимодействия с веб-ресурсами без необходимости изобретения велосипедов
- как следствие, имеет возможностей не меньше, а взаимодействовать с ними гораздо проще

- хорошо работает с современными популярными стандартами WebAPI — REST/RESTful, GraphQL

(см. web scraping - Python requests post a query to graphql with variables - Stack Overflow; GET and POST Requests in GraphQL API using Python requests - GeeksforGeeks)

```
# POST Request
import requests

url = "https://fruits-api.netlify.app/graphql"

body = """
mutation {
  addFruit(
    id: 1
    scientific_name: "mangifera"
    tree_name: "mangifera indica"
    fruit_name: "Mango"
    family: "Anacardiaceae"
    origin: "India"
    description: "Mango is yellow"
    bloom: "Summer"
    maturation_fruit: "Mango"
    life_cycle: "100"
    climatic_zone: "humid"
  ) {
    id
    scientific_name
    tree_name
    fruit_name
    origin
  }
}
"""

response = requests.post(url=url, json={"query": body})
print("response status code: ", response.status_code)
if response.status_code == 200:
    print("response : ",response.content)
```

- поддерживает различные способы авторизации и аутентификации (например, OAuth-токены)
- сама по себе является одной из самых популярных зависимостей для многих веб-ориентированных модулей и фреймворков в экосистеме Python

```
import requests

date_iso = datetime.datetime.now().date.isoformat()
project_url = "https://en.wikipedia.org"
title = "Template:POTD_protected/" + date_iso

API_URL = f'{project_url}/w/api.php'
image_query_params = {
    "action": "query", "format": "json",
    "formatversion": "2", "prop": "images",
    "titles": title
}
potd_image_data = requests.get(API_URL, image_query_params).json()
```

lxml

Доки: https://lxml.de/

- ультимативный парсер HTML- и XML-разметки
- предоставляет огромное количество готовых методов по взаимодействию с иерархией узлов
- является одной из ключевых библиотек, поверх которых построена `BS4`
- может также дополнять `BS4` (BeautifulSoup Parser):

```
import lxml
from bs4 import UnicodeDammit              # BeautifulSoup 4
def decode_html(html_string):
    converted = UnicodeDammit(html_string)
    if not converted.unicode_markup:
        raise UnicodeDecodeError(
            f"Failed to detect encoding, tried [{',
'.join(converted.tried_encodings)}]")
    return converted.unicode_markup
root = lxml.html.fromstring(decode_html(tag_soup))
```

html5lib

Доки: Overview — html5lib 1.2-dev documentation

Исходный код: GitHub - html5lib/html5lib-python: Standards-compliant library for parsing and serializing HTML documents and fragments in Python

- библиотека для парсинга HTML в соответствии со стандартом WHATWG (WHATWG — Википедия)
- это позволяет парсить HTML так же, как это делают все современные крупные браузеры (т.к. именно стандарт WHATWG принят в них к реализации)
- все еще работает на 2-м Питоне (хотя лично вы давно уже не должны этого делать!)
- написана на чистом Питоне с минимумом зависимостей (`lxml`, `genshi`, `chardet` *могут* использоваться как опциональные зависимости для отдельных сценариев использования)
- оперирует концептом дерева элементов и полагается на алгоритмы обхода дерева, что делает ее достаточно эффективной для сложных документов
- есть отдельно модуль `sanitizer` для очистки ввода в соответствии с набором правил WHATWG Sanitization Rules

```
import html5lib
parser = html5lib.HTMLParser(tree=html5lib.getTreeBuilder("dom"))
minidom_document = parser.parse("<p>Hello World!")
```

```
from urllib.request import urlopen
import html5lib

with urlopen("http://example.com/") as f:
    document = html5lib.parse(f,
transport_encoding=f.info().get_content_charset())
```

Вне конкурса: ZenRows

Best Web Scraping Toolkit - ZenRows

Доки: Introduction to the Scraping API - ZenRows Docs

Web Scraping with Python in 2024 - ZenRows

- специализированный сервис для работы с API для крупномасштабных операций по сбору и анализу данных
- !НО — платный от начала и до конца
- явно заточен под дата-ориентированные бизнесы (например, тех, что полагаются на обучение больших нейросетевых моделей)
- зато есть несколько интересных услуг, например, сбор все, что только можно, с YouTube