
Основы веб- скрапинга и работы с API





Олег Булыгин

IT-аудитор, ментор и наставник

Аккаунты в соц.сетях

 fb.com/obulygin91

 vk.com/obulygin91

 obulygin91@ya.ru

 fb.com/obulygin91

 [@obulygin91](https://t.me/obulygin91)



Содержание

1

Что такое веб-скрапинг?

2

Клиент-серверное взаимодействие и HTTP

3

HTML

4

Библиотека Requests

5

Библиотека BeautifulSoup

6

Работа с API



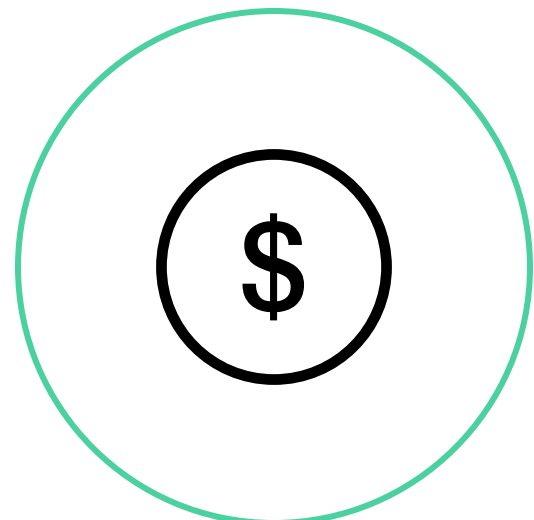
Web-scraping

Web scraping — это процесс извлечения информации из Интернета.

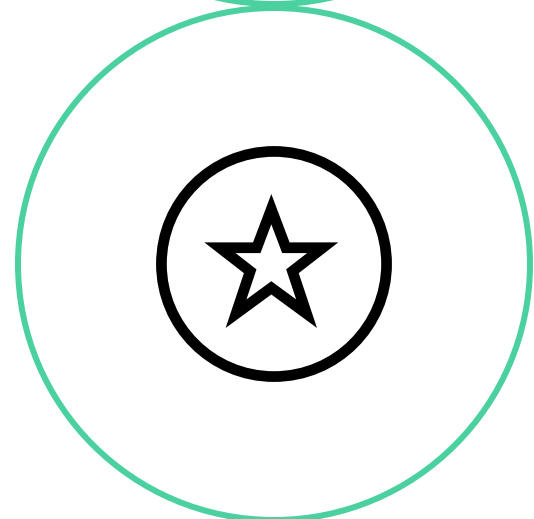
Когда мы говорим “**web scraping**”, подразумевается именно автоматизация этого процесса.



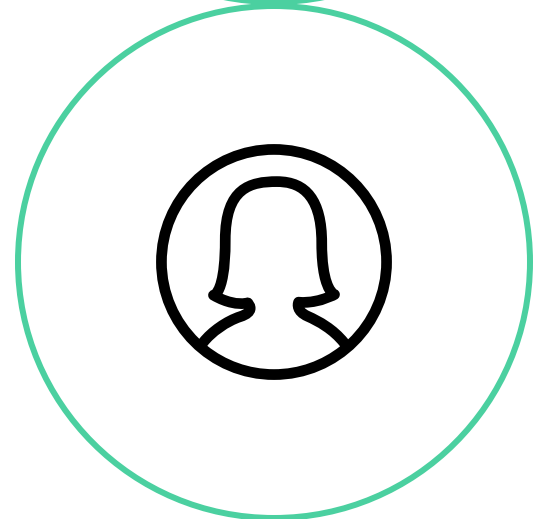
Зачем нужен web scraping?



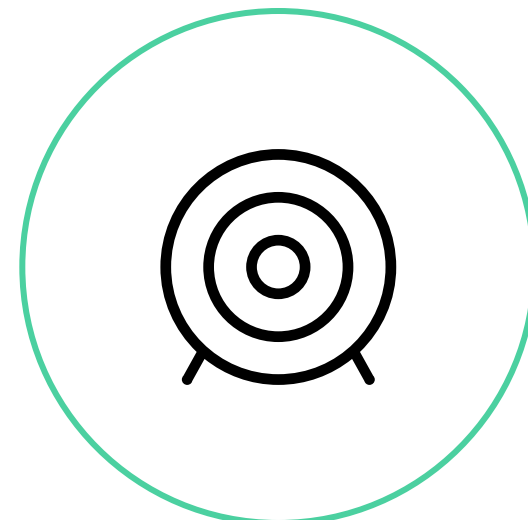
можно собирать цены на товары конкурентов для оптимизации своей стратегии ценообразования



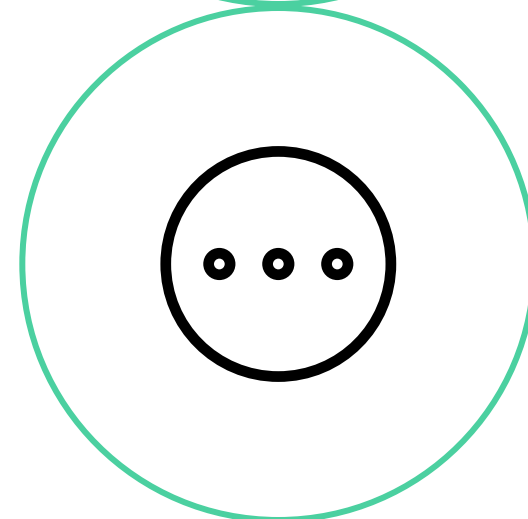
собирать отзывы на свои товары/услуги на различных площадках для анализа



собирать контактные данные пользователей из соцсетей или форумов для дальнейшего взаимодействия с ними



следить за трендами обсуждений в соцмедиа



и еще много всего



Когда мы открываем интернет-страницу — то, что мы видим это заслуга браузера. Как браузер получает эту информацию и как отображает?

Чтобы это понять, нужно ознакомиться со следующими понятиями:



HTTP



HTML



HTTP

HyperText Transfer Protocol

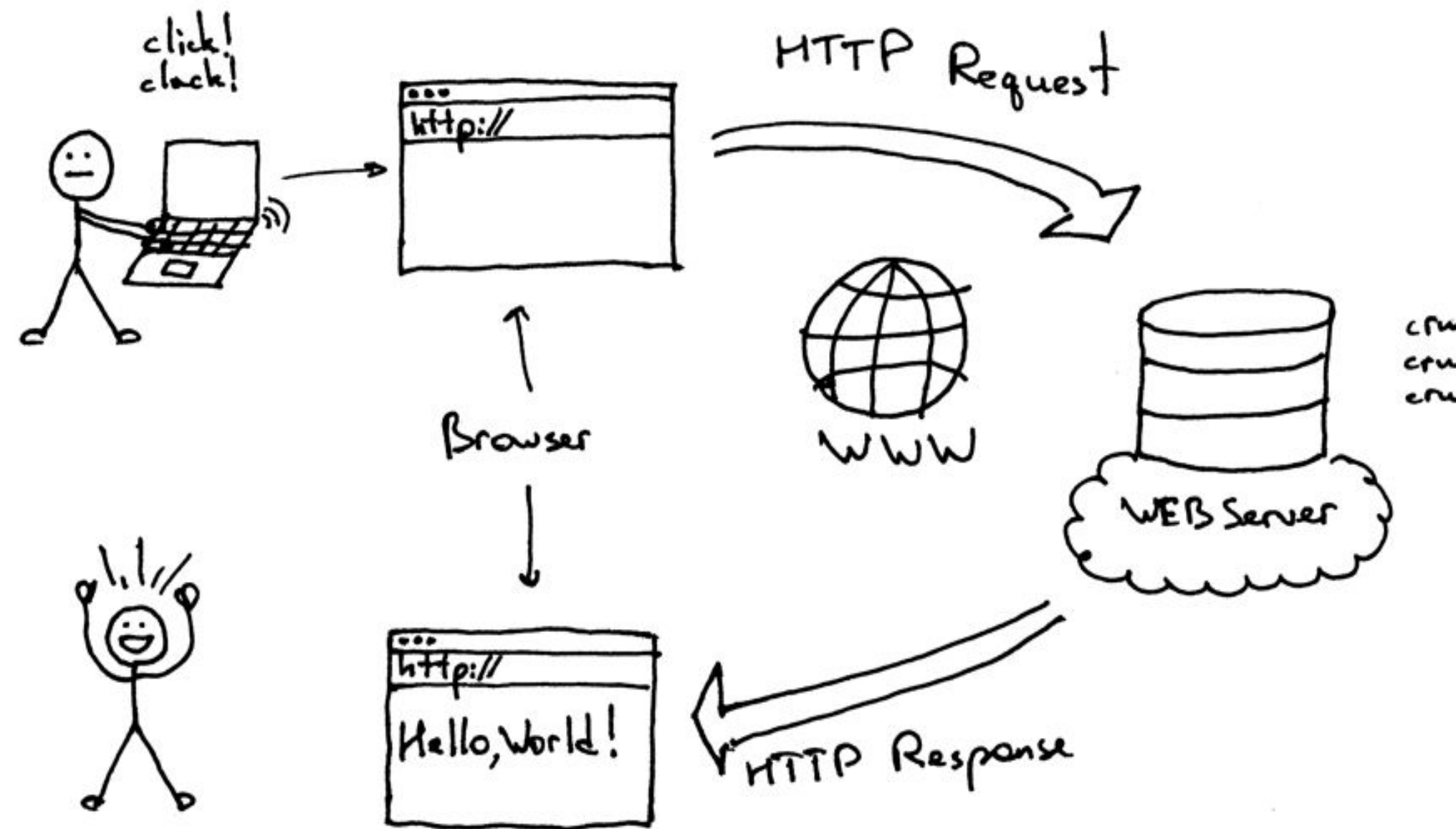
HTTP — это протокол клиент-серверного взаимодействия, позволяющий получать различные ресурсы, например HTML-документы. Протокол HTTP лежит в основе обмена данными в интернете.



HTTP

Клиенты и серверы взаимодействуют, обмениваясь одиночными сообщениями (а не потоком данных).

Сообщения, отправленные клиентом (например, веб-браузером), называются **запросами**, а сообщения, отправленные сервером – **ответами**.



Методы HTTP



GET



PUT



POST



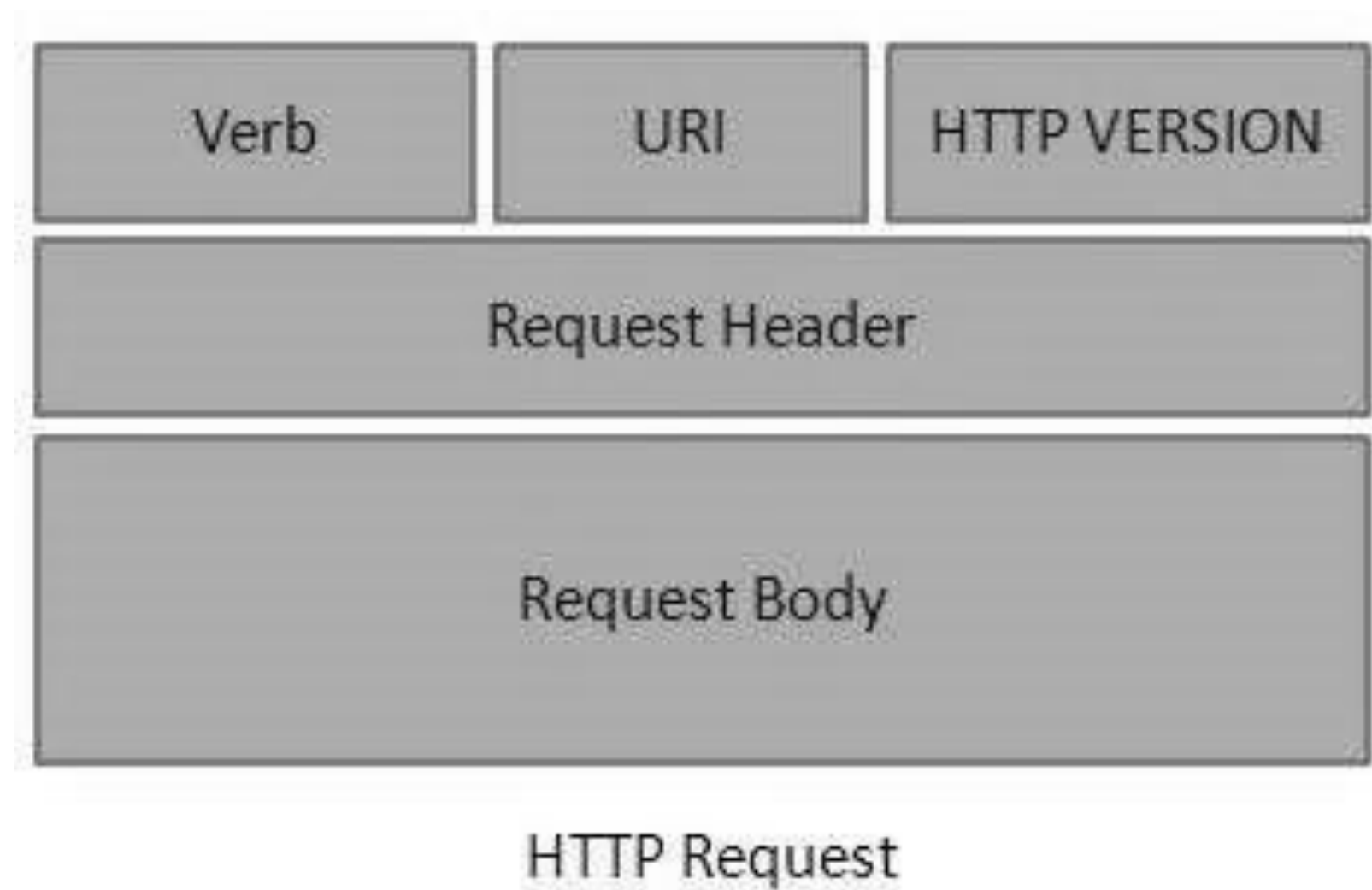
DELETE



Подробнее про методы HTTP можно почитать [здесь](#)



Структура HTTP-запроса



HTML

HyperText Markup Language

HTML — стандартизированный язык разметки документов в интернете. Большинство веб-страниц содержат описание разметки на языке HTML. Язык HTML интерпретируется браузерами; полученный в результате интерпретации текст отображается на экране монитора компьютера или мобильного устройства.



HTML

Теги

Вся информация о форматировании документа сосредоточена в его фрагментах - **тегах** - заключенных между знаками «<» и «>».

Большинство HTML-тегов — **парные**, например, <head> </head>

Вложенные элементы

Можно вкладывать элементы внутрь других элементов — это называется **вложенностью**.

Атрибуты

Атрибуты содержат дополнительную информацию о том, как браузер должен обработать текущий тег.


Например, атрибут **class** позволяет дать элементу идентификационное имя, которое в дальнейшем может быть использовано для обращения к элементу с информацией о стиле и прочими вещами.



HTTP-запросы и HTML

А где это можно посмотреть?

Для этих целей можно использовать **Developer Tools**. Его можно открыть следующими способами:

- **F12**
-  -> Inspect / Просмотреть код
- Сочетание клавиш, например: **Ctrl + Shift + I**



Библиотека Requests

Requests — это библиотека, которую вы можете использовать для отправки всех видов HTTP-запросов. У нее много функций, начиная от передачи параметров в URL-адресах до отправки пользовательских заголовков и проверки SSL.

Документация:

<https://requests.readthedocs.io/en/master/user/quickstart/>



Библиотека BeautifulSoup

Beautiful Soup — это библиотека для извлечения данных из HTML и XML. С ее помощью можно извлечь из сложной структуры разметки нужную информацию.

Документация:

<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>



Проблемы веб-скрапинга

- 1 Огромное многообразие сайтов и у каждого сайта своя структура
- 2 Сайты постоянно меняются
- 3 Контент сайта может формироваться динамически (тогда его содержимое из HTML не достать)
- 4 Сайты могут использовать различные средства для защиты от скрапинга.



А может можно обойтись без скрапинга?

Да, если сайт предоставляет **API** (**Application programming interface**) – специальный интерфейс для удобного извлечения информации программами, а не человеком.

Это набор готового функционала, который открыт для работы извне, и которым можно воспользоваться для решения своих задач.



Проблемы при работе с API

1 Все API абсолютно разные

2 Функционал строго ограничен тем, какой заложили в него разработчики

3 Может подразумевать очень много ограничений, которые определили разработчики

4 Могут быть платными

5

Для работы со многими API требуется наличие специальных ключей доступа, выдачу которых полностью контролирует сам сервис

6

Документация к конкретному API может быть очень низкого качества (а может вообще отсутствовать)



Вопросы?

Основы веб- скрапинга и работы с API

Олег Булыгин

Python для анализа данных

 **НЕТОЛОГИЯ**