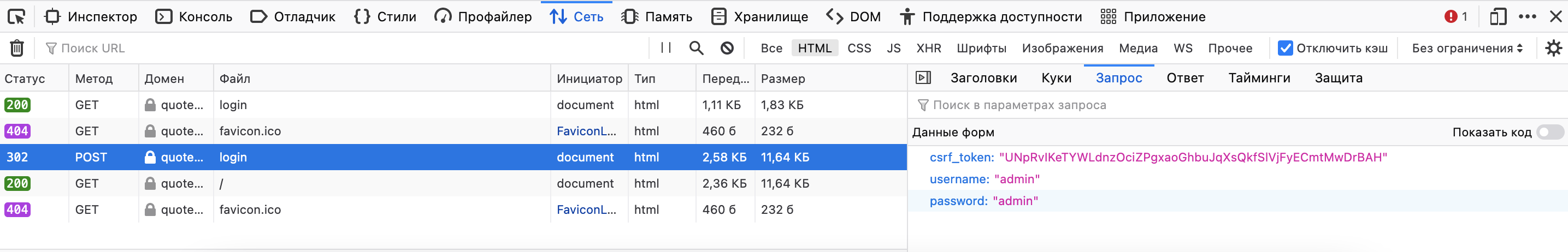
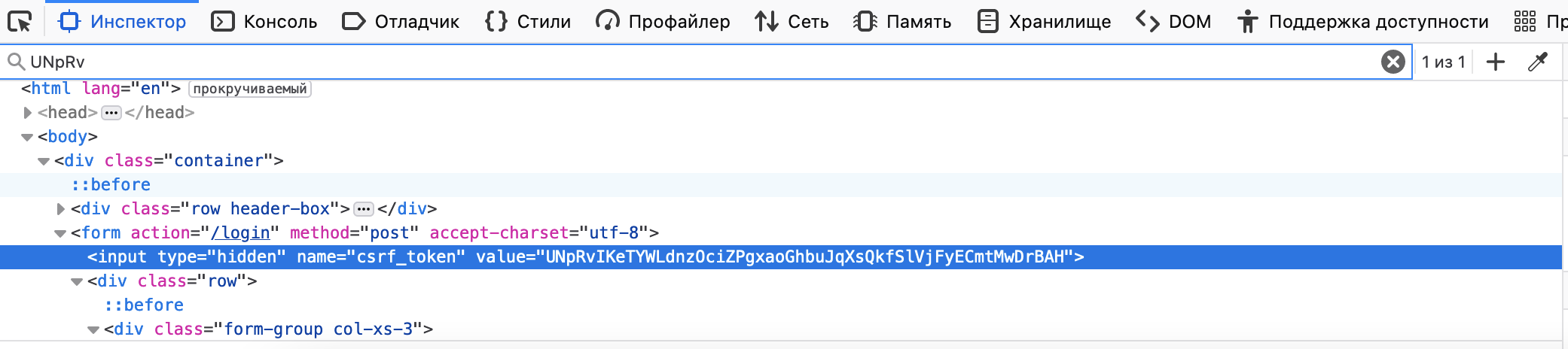
Методы сбора и обработки данных при помощи Python

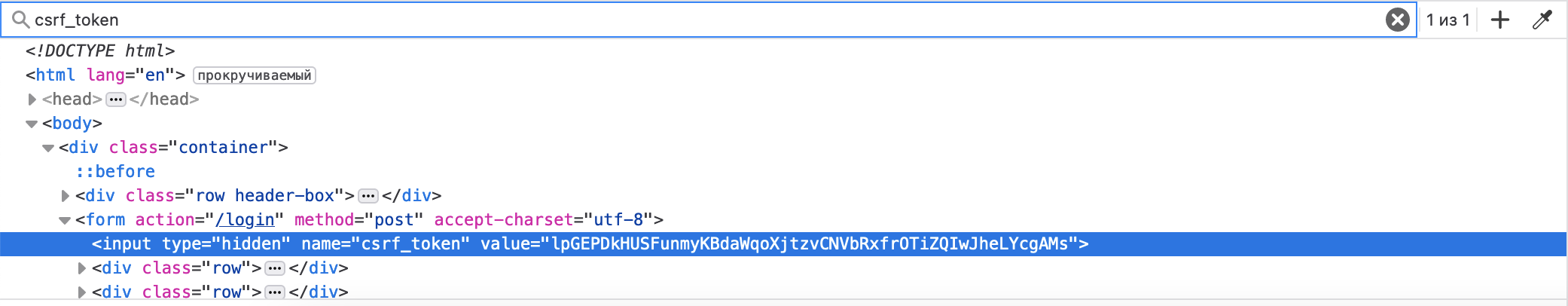


На этом уроке

1. Научимся заполнять формы логина в scrapy.
2. Научимся отправлять API запросы в Scrapy

На прошлом уроке мы научились лггиниться на сайт с использованием селениума. Теперь давайте изучим, как это делать при помощи скрейпи. Переходим на наш сайт [quotes.toscrape.com/login](http://quotes.toscrape.com/login). Открываем инспектор и переходим на вкладку «Сеть». Заполняем поля admin/admin и нажимаем login. Смотрите, у нас появился запрос с ответом 302, то есть перенаправление, и названием логин. Открываем запрос, в заголовках вы видите адрес по которому был сделан запрос, метод - Post, нас интересует вкладка запрос.

Как видите, вместе с запросом был отправлен массив, в котором есть такие ключи, как username, password и csrf\_token. Откуда у нас юзернейм и пароль - понятно. Но давайте посмотрим, где получить этот токен. Снова открываем страницу с логином, проще всего вернуться назад. Смотрим разметку, попробуем ввести первые символы токена.

Так, у нас нашелся тег с именем csrf\_token. Отлично, это то, что нужно. Давайте еще проверим, появляется ли этот тег с токеном при отключенном JavaScript’е. F1, отключить джаваскрипт, обновляем страницу и ищем csrf\_token.

Отлично, значит, мы можем использовать скрейпи без сплеша. Создадим новый проект

**scrapy startproject login**

**cd login**

**scrapy genspider quotes [quotes.toscrape.com](http://quotes.toscrape.com)**

Открываем файли паука. Первое, что надо сделать - импортировать библиотеку, которая позволит нам отправить post запрос.

**from scrapy import FormRequest**

Меняем start\_urls

**start\_urls = ['https://quotes.toscrape.com/login']**

Теперь в методе parse ищем токен

**csrf\_token = response.xpath(‘//input[@name="csrf\_token"]/@value').get()**

И после того, как токен найден, отправляем запрос с помощью FormRequest

**yield FormRequest.from\_response(**

**response,** - на какую ссылку отправлять запрос

**formxpath=‘//form',** - находим форму по xpath, так как у нас всего одна форма на странице, сделаем это так

**formdata={**

**'csrf\_token': csrf\_token,**

**'username': 'admin',**

**'password': 'admin'**

**},** - формируем данные, которые надо отправить

**callback=self.after\_login** - после отправки запроса переходим к выполнению функции after\_login

**)**

Давайте в функции after\_login соберем все цитаты с первой страницы, то есть всего 10 штук. Не будем углубляться в сбор текстов и авторов.

**def after\_login(self, response):**

**quotes = response.xpath("//div[@class='quote']")**

**print(f'Scrapy crawled {len(quotes)} quotes’)**

Запускаем паука - всё верно. Но может быть такая ситуация, что токен появляется только после выполнения джава скрипта и сам логин выполняется при помощи javascript’а. Тогда вам надо использовать Splash. Давайте представим, что наш сайт именно такой и напишем еще один скрипт.

Создадим нового паука

**scrapy genspider js\_login**

Импортируем библиотеки

**from scrapy\_splash import SplashRequest, SplashFormRequest**

Удаляем start\_urls. Теперь напишем скрипт для открытия страницы, точно такой же, какой мы писали на предыдущих уроках

**script = '''**

**function main(splash, args)**

**assert(splash:go(args.url))**

**assert(splash:wait(0.5))**

**return splash:html()**

**end**

**‘''**

затем создадим метод start\_requests и внутри него вызовем сплеш реквест

**def start\_requests(self):**

**yield SplashRequest(**

**url='https://quotes.toscrape.com/login',**

**endpoint='execute',**

**args = {**

**'lua\_source': self.script**

**},**

**callback=self.parse**

**)**

Всё то же самое, что мы уже делали. Затем переходим к методу parse. Сначала находим наш токен.

**csrf\_token = response.xpath(‘//input[@name="csrf\_token"]/@value').get()**

Затем отправляем форму при помощи класса SplashFormRequest

**yield SplashFormRequest.from\_response(**

**response,**

**formxpath='//form',**

**formdata={**

**'csrf\_token': csrf\_token,**

**'username': 'admin',**

**'password': 'admin'**

**},**

**callback=self.after\_login**

**)**

По сути, кроме класса отправка формы ничем не отличается от предыдущего скрипта. И наконец метод after\_login, который мы просто скопируем из предыдущего скрипта.

**def after\_login(self, response):**

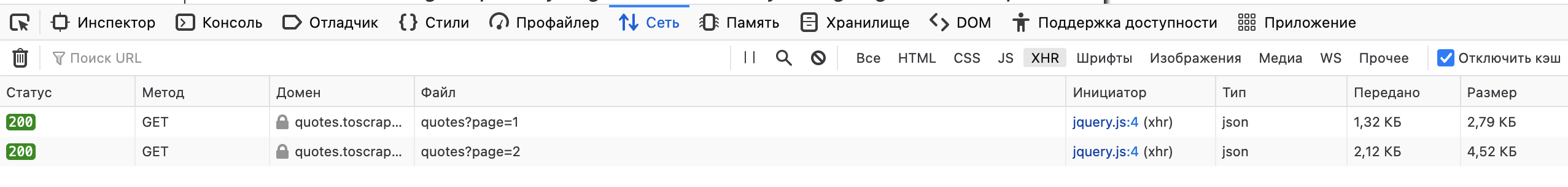
**quotes = response.xpath("//div[@class='quote']")**

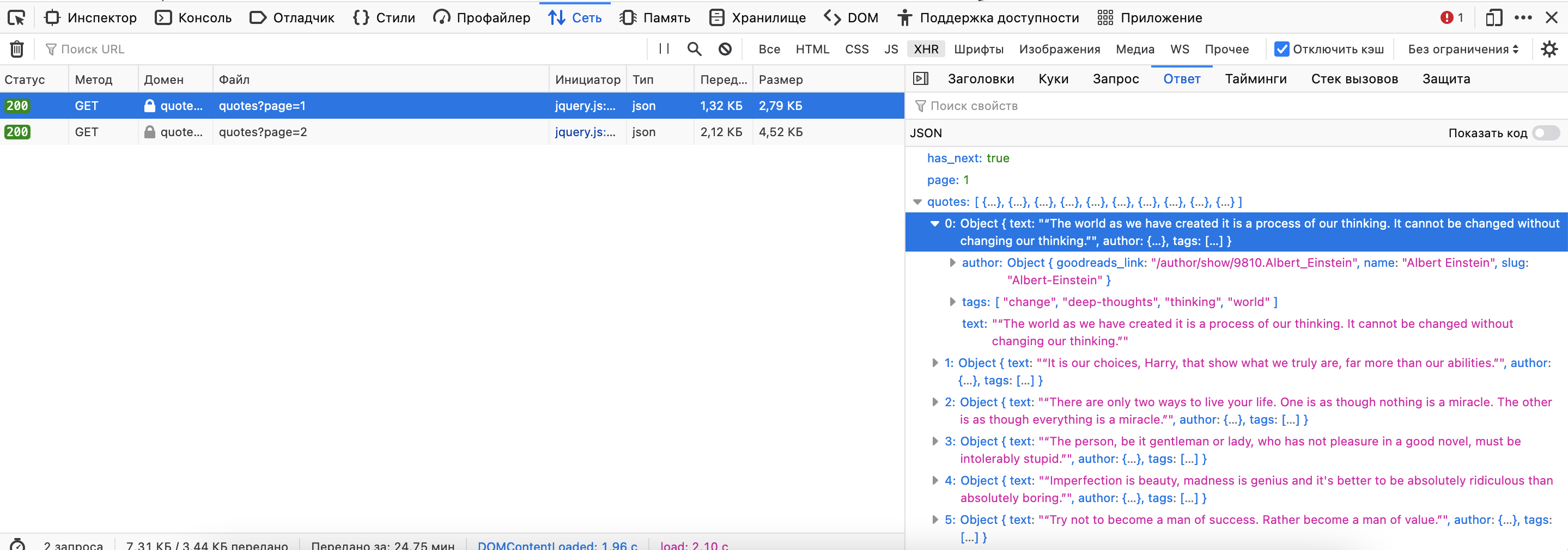
**print(f'Scrapy crawled {len(quotes)} quotes’)**

Вот таким образом мы можем логиниться на сайт при помощи скрейпи.

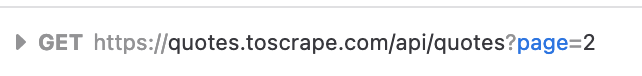
Последнее, что нам осталось затронуть - это работу с API при помощи Scrapy. Помните, на прошлом уроке, когда мы разбирались с бесконечным скролом в селениуме и отправляли команды в консоль, у нас там появлялся метод api? Сейчас вам еще раз покажу.

Открываем <https://quotes.toscrape.com/scroll> и вкладку сеть в инспекторе. Включаем XHR, чтобы посмотреть только апи запросы. Вообще апи позволяет легко и просто получать нужные данные легальным способом. Всегда сначала проверяйте эту вкладку при работе с сайтом, а уже потом пробйуте писать свой парсер. Бывает так, что по апи отдаются не все нужные данные, приходится переходить по ссылкам дальше, собирать айдишники и искать новые запросы. Но даже это проще, чем писать парсер html.

Пролистаем страницу чуть вниз, чтобы подгрузились новые данные и видим, что у нас есть GET запросы

Давайте посмотрим какой-нибудь. Видим, что в ответ приходит json, в котором есть несколько ключей, в том числе ключ quotes, в котором лежат словари с данными о цитатах, авторах и тегах.

Отлично, теперь осталось понять, что меняется в запросе. Мы видим, что первый и второй запрос отличаются номером страницы в конце ссылкиpasted-image.png



Отлично, давайте создадим новый проект

**scrapy startproject api**

**cd api**

**scrapy genspider quotes [quotes.toscrape.com](http://quotes.toscrape.com)**

Открываем файл паука. В начале импортируем json, так как нам придется с ним работать, когда мы получим ответ от api.

**import json**

Затем изменим ссылку в start\_urls на первую ссылку апи запроса

**start\_urls = [‘http://quotes.toscrape.com/api/quotes?page=1']**

Теперь извлечем json из ответа

**resp = json.loads(response.body)**

Получим цитаты

**quotes = resp.get(‘quotes')**

И вытащим данные из цитат.

**for quote in quotes:**

Давайте посмотрим. Автор у нас лежит по ключу автор, а затем ключ имя

**yield {**

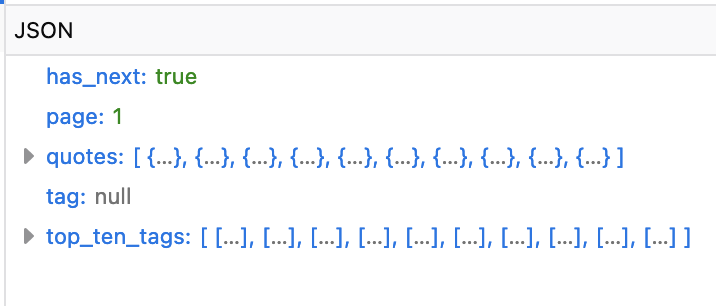
**'author': quote.get('author').get('name'),**

**'tags': quote.get('tags'),**

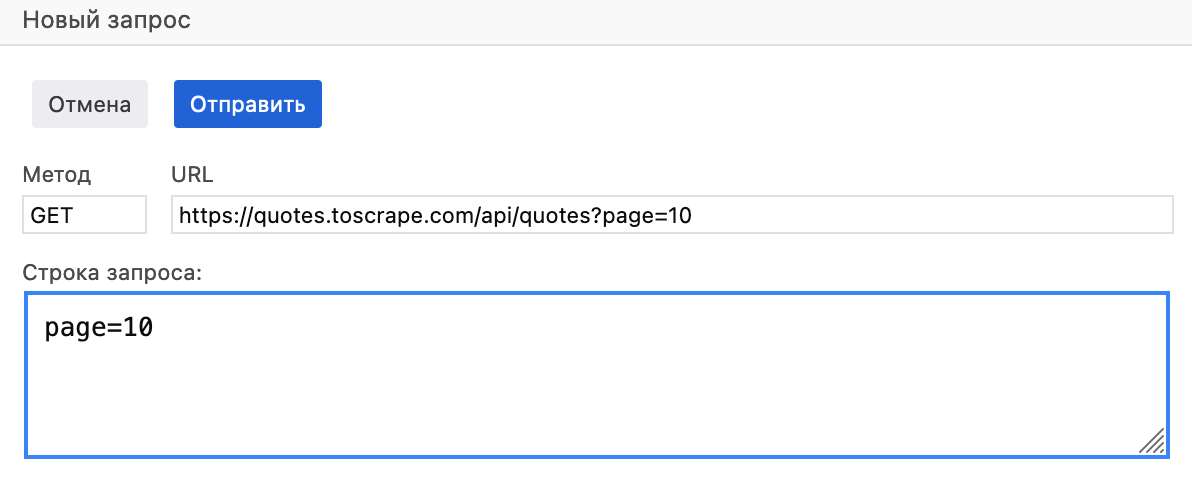
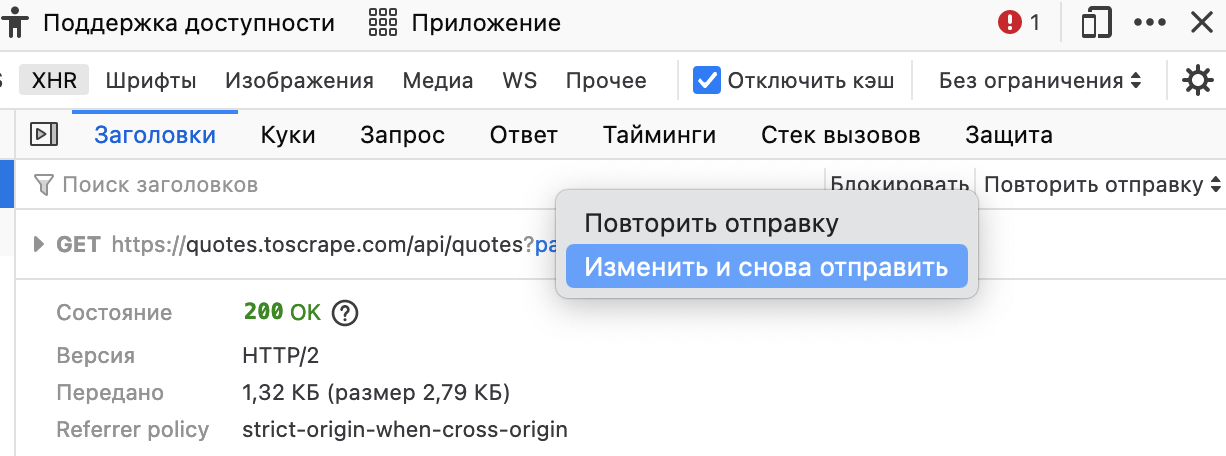
**'quote\_text': quote.get('text')**

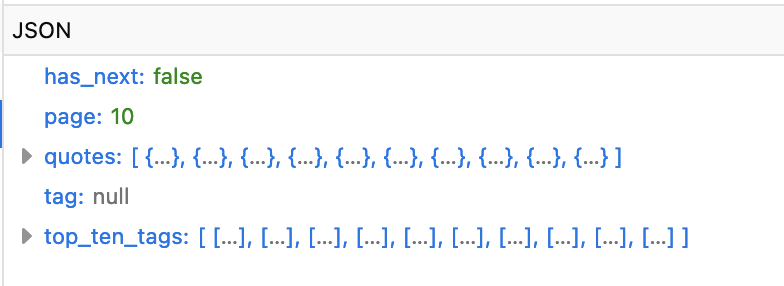
**}**

Теги у нас лежат по ключу tags. Сама цитата лежит в ключе text.

Теперь нам надо понять, каким образом мы получим все цитаты. Посмотрим внимательнее на ключи в ответе апи.

Тут есть ключ has\_next. Возможно, на последней странице этот ключ поменяет свое значение на false. Давайте изменим наш запрос. Добавим 10 страницу, так как из предыдущих уроков мы знаем, что страниц всего 10.

В обычной жизни, когда вы не знаете точное количество страниц, вам бы, скорее всего, пришлось писать цикл while True и проверять этот ключ до тех пор, пока он не станет false. И еще надеяться, что разработчики не забыли и поменяли этот ключ на последней странице. У нас же с вами идеальные условия, так что меняем запрос и смотрим.

Видим, что тут значение ключа has\_next изменилось

Отлично, давайте писать код. Сначала создадим переменную has\_next и получим для нее значение

**has\_next = resp.get(‘has\_next’)**

Если эта переменная равна true, то делаем запрос по апи еще раз, только меняя номер страницы. Номер страницы мы будем так же получать из нашего ответа.

**if has\_next:**

**next\_page\_number = resp.get('page') + 1**

**yield scrapy.Request(**

**url=f'http://quotes.toscrape.com/api/quotes?page={next\_page\_number}',**

**callback=self.parse**

**)**

Вот и всё. Запускаем пука. Ждем. В item\_scraped\_count собралась тысяча документов. Прекрасно. Мы с вами научились работать с апи.

На этом наш курс подходит к концу. Я надеюсь, что вам понравился парсинг. Этих основ более чем достаточно для самостоятельно дальнейшего погружения в мир автоматического сбора данных. Ну, а при любых трудностях можете смело обращаться к Интернету, StackOverflow и ко мне =)

Домашнее задание

Залогиниться на [сайте](https://www.scrapethissite.com/pages/advanced/?gotcha=csrf), используя Scrapy. Вывести сообщение, которое появляется после логина (связка логин/пароль может быть любой).