

Привет всем любознательным! В данной пасхалке будет находиться дополнительный материал по парсеру **Beautiful Soup**. На самом деле, это не совсем парсер. Просто высокоуровневая абстракция, которая позволяет более удобно работать с низкоуровневыми библиотеками. Например, `xpath` там вообще как таковой не нужен, а нужная информация в `html` ищется буквально в пару строчек. Вот взять тот же `lxml`, например, — он довольно древний, а разработчики, почему-то, пренебрегли шаблоном проектирования ФАБРИКА (надеюсь, помните его), из-за чего приходится порой сильно напрягать мозги. В общем прелюдия окончена. Милости просим на скринкаст и дальнейший материал.

[Видеоматериал](#)

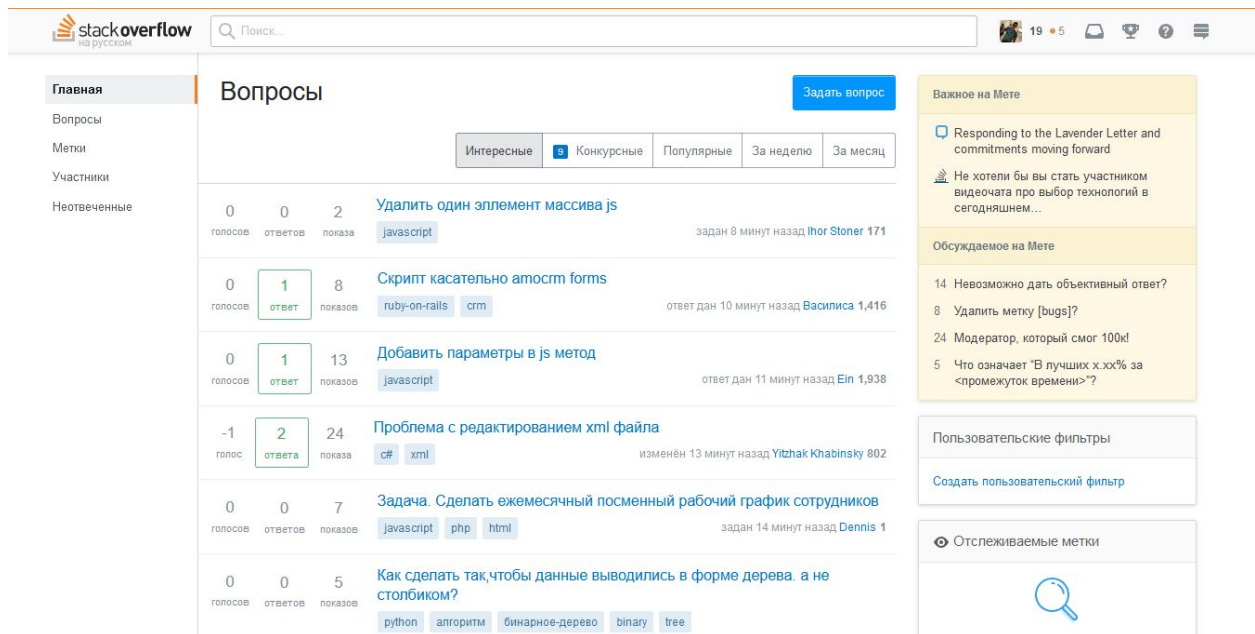
В данном конкретном примере мы напишем парсер, который будет собирать все заголовки вопросов и ссылки на них с ресурса *stackoverflow*.

Для написания парсеров в *Python* есть специальная библиотека — *Beautiful Soup*
4. Эта библиотека помогает избежать прямой работы с другими более сложными библиотеками, такими как *html.parser* или *lxml*. Давайте же установим эту библиотеку. Первое, что нам надо сделать, это ввести в терминале:

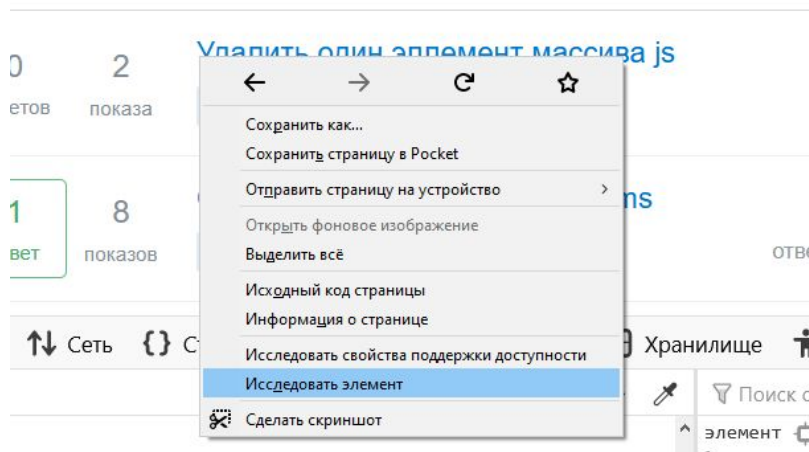
```
pip3 install beautifulsoup4  
pip3 install lxml
```

Вторая библиотека нам нужна, т. к. в ней хранятся сами алгоритмы поиска в *HTML*. Без библиотеки *lxml beautifulsoup4* работать не будет.

Перед тем как мы перейдём в редактор давайте взглянем на страничку, которую мы сегодня будем парсить. Этой страничкой будет главная страница *stackoverflow*, как уже говорилось ранее.



Мы уже открывали с вами консоль разработчика, и сегодня она нам также понадобится. Но чтобы не искать долго нужный код элемента, достаточно нажать правой кнопкой мыши по списку вопросов и выбрать пункт «Исследовать элемент» (если ваш браузер на английском языке, то вам надо будет нажать «Inspect element»).



Теперь при нажатии у нас откроется консоль разработчика с html-кодом, которая направит нас на конкретный тег, который нам нужен.

```
▼ <div id="mainbar">
  ▶ <div class="grid"> ... </div> flex
  ▶ <div class="grid ai-center mb16"> ... </div> flex
  ▼ <div id="qlist-wrapper" class="flush-left">
    ▼ <div id="question-mini-list">
      ▼ <div>
        ▼ <div id="question-summary-1190866" class="question-summai
          ▶ <div class="cp" onclick="window.location.href='/questic
            %bd%d1%82-%d0%bc%d0%b0%d1%81%d1%81%d0%b8%d0%b2%d0%b0-j:
          ▼ <div class="summary">
            ▼ <h3>
              <a class="question-hyperlink" href="/questions/1190
                %d1%82%d1%8...b5%d0%bd%d1%82-%d0%bc%d0%b0%d1%81%d1%81
```

Как мы видим, все вопросы хранятся в контейнере (тег `<div>`, в дальнейшем они ещё несколько раз будут названы как «контейнеры») с `id="question-mini-list"`.

Давайте попробуем получить html-код данного конкретного контейнера. Перейдём уже наконец в редактор:

```
from bs4 import BeautifulSoup
import requests

base = 'https://ru.stackoverflow.com' # выносим базовую ссылку
# в отдельную переменную, она нам потом
# понадобится
html = requests.get(base).content # с помощью уже знакомой нам
# библиотеки requests получаем html код странички
# целиком
soup = BeautifulSoup(html, 'lxml') # создаём объект супа.
# Первый аргумент в конструкторе - это весь html
# код. Второй аргумент - сама библиотека для
# парсинга. В нашем случае lxml
div = soup.find('div', id='question-mini-list') # находим с помощью
# метода find() нужный нам див уточняя id

print(div) # пишем в консоль то что нашли
```

В результате после запуска спустя какое-то время появится следующий вывод в консоли:

Здесь нам вывелся весь html-код данного контейнера.

Не смотря на то что нам и вывелся html-код, сам объект, возвращающийся с метода *find* не является html-кодом (т. е. строкой). Это точно такой же объект *BeautifulSoup*, только в нём хранится только нужный нам html, который мы искали.

0

2

Удалить один элемент массива js

1

8

Скрипт касательно amocrm forms

ответов

показа

javascript

ответ

показов

ruby-on-rails

crm

Навигатор

↕ Сеть

{ } Стили

🔍 Профайлер

🧠 Память

📁 Хранилище

+

🔍

🔍 Поис

1190866" class="question-summary narrow"> flex

"window.location.href='/questions/1190866/%d0%a3%d0%b4%d0%b0...5%d01%81d1%81d0%b8d0%b2d0%b0-js'"> ...</div> event flex

hyperlink" href="/questions/1190866/%d0%a3%d0%b4%d0%b0%d0%bb%d0%b8%d1%82-%d0%bc%d0%b0d1%81d1%81d0%b8d0%b2d0%b0-js">

т массива js

"question-hyperlink" href="/questions/1190866/%d0%a3%d0%b4%d0%b0%d0%bb%d0%b8...b5%d0%bd%d1%82-%d0%bc%d0%b0d1%81d1%81d0%b8d0%b2d0%b0-js">

дин элемент массива js

элемент

{

.question

{

for

}

.question

Как мы видим, нас интересует тег `<a>` с классом `"question-hyperlink"`, давайте найдём все теги `<a>` в данном контейнере с этим классом. Для этого будем использовать

метод `find_all()`, который возвращает нам список объектов *beautifulsoup*, все из которых удовлетворяют нашим критериям.

```
from bs4 import BeautifulSoup
import requests

base = 'https://ru.stackoverflow.com'
html = requests.get(base).content
soup = BeautifulSoup(html, 'lxml')
div = soup.find('div', id='question-mini-list')

a = div.find_all('a', class_='question-hyperlink') # ищем только
нужные нам объекты тегов <a>
print(a)
```

Обратите внимание, что поскольку слово `class` зарезервировано в *Python*, то при указании аргументов нужно добавить нижнее подчёркивание.

В результате выполнения скрипта в консоли должно напечататься нечто следующего вида:

1%83%87%b0%b9%bd%be%81%82%8c-%d0%be%b4-%d0%bc%be%89%bd%be%81%82%8c-%d1%80%b8%82%8c-%d0%a5%b8-%d0%ba%b2">Проверить ЛКГ адрат критерий, <a class="question-hyperlink" href="/question%8c-%d0%b2%8b%b2%be%b4-%d0%bd%b5%81%80%ba%b%b8%b9-%d0%b2-%d1%82%b0%b1%bb%b8%86%83-s5QIite3 and Python?,

Если приглядеться, можно увидеть, что мы получили то, что хотели. Ну почти. :) Осталось разделить данный тег и отдельно напечатать в консоль название вопроса и ссылку на него. И да, обратите внимание, что в данном случае у нас выводится список.

```
from bs4 import BeautifulSoup
import requests

base = 'https://ru.stackoverflow.com'
```



```

html = requests.get(base).content
soup = BeautifulSoup(html, 'lxml')
div = soup.find('div', id='question-mini-list')

a = div.find_all('a', class_='question-hyperlink') # ищем только
нужные нам объекты тегов <a>

for _ in a:
    # метод getText() возвращает текст внутри тега.
    Это и будет название вопроса
    # чтобы получить атрибут тега (ссылка лежит в
    атрибуте href) можно обратиться к нему как к
    словарю через [] или же использовать метод .get()
    print(_.getText(), base + _.get('href')) # печатаем название
    вопроса и ссылку на него в консоль

```

Теперь в консоли у нас должно вывестись:

```

0%bc%d0%b8%d1%87%d0%b5%d1%81%d0%ba%d0%b8-%d0%bc%d0%b5%d0%bd%
80%d0%b5%d0%bc%d0%b5%d0%bd%d0%bd%d0%be%d0%b9-%d0%bf%d0%be%d0%
d0%b2%d0%b5%d1%80%d0%b0
Обход блокировки playmarket из-за санкций https://ru.stackove
%d0%be%d0%ba%d0%b8%d1%80%d0%be%d0%b2%d0%ba%d0%b8-playmarket-%
Интернет-магазин на чистом PHP для портфолио PHP-разработчика
5%d1%80%d0%bd%d0%b5%d1%82-%d0%bc%d0%b0%d0%b3%d0%b0%d0%b7%d0%b
%bb%d1%8f-%d0%bf%d0%be%d1%80%d1%82%d1%84%d0%be%d0%bb%d0%b8%d0%
%ba%d0%b0

```

И теперь с этими данными можно делать всё, что угодно. Например, сохранять в файл, excel таблицы и т. д.

Нередко также бывает, что надо найти родительский тег (т. е. тег, внутри которого находится искомый элемент). Для этого можно воспользоваться методом *find_parent()*, который работает также, как и *find()*, но возвращает тег родителя. Слегка поменяем наш код.

```

from bs4 import BeautifulSoup
import requests

base = 'https://ru.stackoverflow.com'

```

```

html = requests.get(base).content
soup = BeautifulSoup(html, 'lxml')
div = soup.find('div', id='question-mini-list')

a = div.find('a', class_='question-hyperlink') # попробуем найти
первый тег a
parent = a.find_parent() # находим первого родителя тега
a.
print(parent) # Печатаем тег родитель. В данном случае
это должен быть тег <h3>

```

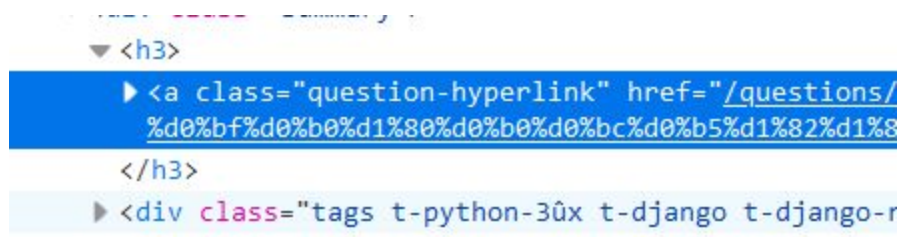
Как мы видим, вывелось примерно следующее:

```

C:\Users\skavi\Documents\repositories\education\bs>C:/Users
documents/repositories/education/bs/parsing.py
<h3><a class="question-hyperlink" href="/questions/1190910/
-win-crt-heap-l1-1-0-dll-%d1%87%d1%82%d0%be-%d1%81-%d1%8d%
-win-crt-heap-l1-1-0.dll. что с этим делать?</a></h3>

```

Т. е. мы увидели тег <h3>. Давайте посмотрим, так ли это. Перейдем обратно в консоль на сайте и посмотрим.



```

▼ <h3>
  ► <a class="question-hyperlink" href="/questions/
    %d0%bf%d0%b0%d1%80%d0%b0%d0%bc%d0%b5%d1%82%d1%8
  </h3>
  ► <div class="tags t-python-3 t-django t-django-r

```

Да, действительно, наш родительский тег это h3. Получили то, что и хотели.

На этих базовых методах и строится 95% работы всех парсеров. Однако, порой бывает так, что объекты генерируются по мере просмотра с помощью динамических скриптов. В этом случае используются более сложные инструменты, такие как веб-драйвер. Но об этом, возможно, в другой раз.