



Методы сбора и обработки данных при помощи Python

HTML, DOM, XPath

На этом уроке

1. Познакомимся с HTML-разметкой.
2. Узнаем, что такое DOM.
3. Научимся работать с Xpath.

Оглавление

[HTML](#)

[Теги](#)

[Структура страницы HTML](#)

[Атрибуты тегов и их значения](#)

[Атрибут class](#)

[Атрибут id](#)

[DOM](#)

[Уровни DOM](#)

HTML

HTML (HyperText Markup Language) — язык гипертекстовой разметки, на котором написано большинство веб-страниц. HTML интерпретируется браузерами, в результате чего форматированный текст отображается на экране. HTML-страницы передаются от сервера к клиенту по протоколу HTTP в виде обычного либо зашифрованного текста. Такие страницы обычно имеют расширения html или htm.

Актуальная версия HTML — 5.2. Она представлена 14 декабря 2017 года. HTML5 оживил всемирную паутину, в нём появились новые элементы: например, тег `<video>` для вставки видеопотока и `<audio>` для аудиопотока. Большинство украшающих тегов удалили (например, `<center>` и ``), так как функции украшения HTML-страницы на себя взяли каскадные таблицы стилей (CSS). HTML5 также предоставляет доступ к API, с которыми можно работать при помощи языка JavaScript.

Теги

HTML-теги — основа HTML. Теги используются для разграничения начала и конца элементов в разметке. Каждый HTML-документ состоит из дерева HTML-элементов и текста. Каждый HTML-элемент обозначается начальным (открывающим) `<>` и конечным (закрывающим) `</>` тегом. Открывающий и закрывающий теги содержат имя тега.

Все HTML-элементы делятся на пять типов:

- **пустые элементы** — `<area>`, `<base>`, `
`, `<col>`, `<embed>`, `<hr>`, ``, `<input>`, `<link>`, `<menuitem>`, `<meta>`, `<param>`, `<source>`, `<track>`, `<wbr>`;

- элементы с неформатированным текстом — `<script>`, `<style>`;
- элементы, выводющие неформатированный текст — `<textarea>`, `<title>`;
- элементы из другого пространства имён — MathML и SVG;
- обычные элементы — все остальные элементы.

Важно! Внутри тегов хранится большинство необходимой информации, поэтому важно не только находить данные правильно, но и верно определять, в каких тегах они находятся.

Структура страницы HTML

Перейдём на сайт [Books to Scrape](#) — это один из двух сайтов, созданных специально для того, чтобы их парсили. Вы можете самостоятельно познакомиться со вторым сайтом на домашней странице [Toscrape.com](#). Второй сайт предоставляет дополнительные возможности для погружения в мир парсинга, например, отключение JavaScript или бесконечный скролл страницы.

Итак, мы на главной странице [Books to Scrape](#). Теперь нажимаем правой кнопкой мыши и выбираем «Исследовать» — открывается окно с несколькими вкладками. Например, вкладка «Сеть» позволяет посмотреть, какие запросы были отправлены сайту. Обновим страницу и увидим — мы можем фильтровать запросы по их типу. Нажмём на HTML и поймём, что был отправлен один GET-запрос. Пролистав ниже, увидим заголовки. Во вкладке «Куки» увидим куки, на вкладке «Запрос» — параметры, которые были отправлены сайту (здесь пусто), во вкладке «Ответ» — непосредственно сам сайт.

```

<!DOCTYPE html>
<!--[if lt IE 7]> <html lang="en-us" class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
<!--[if IE 7]> <html lang="en-us" class="no-js lt-ie9 lt-ie8"> <![endif]-->
<!--[if IE 8]> <html lang="en-us" class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!-->
<html class="no-js" lang="en-us">
  <!--<![endif]-->
  <head>
    <title>All products | Books to Scrape - Sandbox</title>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <meta name="created" content="24th Jun 2016 09:29">
    <meta name="description" content="">
    <meta name="viewport" content="width=device-width">
    <meta name="robots" content="NOARCHIVE,NOCACHE">
    <!--Le HTML5 shim, for IE6-8 support of HTML elements-->
    <!--[if lt IE 9]> <script src="//html5shim.googlecode.com/svn/trunk/html5.js"></script> <![endif]-->
    <link rel="shortcut icon" href="static/oscar/favicon.ico">
    <link rel="stylesheet" type="text/css" href="static/oscar/css/styles.css">
    <link rel="stylesheet" href="static/oscar/js/bootstrap-datetimepicker/bootstrap-datetimepicker.css">
    <link rel="stylesheet" type="text/css" href="static/oscar/css/datetimepicker.css">
  </head>
  <body id="default" class="default">
    <header class="header container-fluid">
      <div class="container-fluid page">
        <::before>
          <div class="page_inner">
            <::before>
              <ul class="breadcrumb">
                <div class="row">
                  <::before>
                    <aside class="sidebar col-sm-4 col-md-3">
                      <div class="col-sm-8 col-md-9">
                        <div class="page-header action">
                          <div id="messages">

```

Нас интересует вкладка «Инспектор», где и находится вся разметка сайта. Это хороший сайт, тут все теги удобно и понятно названы.

Разберём эту страницу построчно:

- **<!DOCTYPE html>** — здесь мы объявляем, что это документ HTML версии 5. Для 4-й версии HTML эта строка будет следующей:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

- **html** — элемент, включающий всю веб-страницу;
- **head** — головная часть HTML-страницы: в ней обычно описываются метатеги документа, подключаются дополнительные таблицы стилей и JavaScript, объявляется title страницы (текст, который будет выведен в заголовке окна или вкладки браузера) — всё, что касается мета- и описательной информации HTML-страницы;
- **body** — тег, внутри которого описывается основное содержимое страницы;
- **header** — именованный тег, определяющий заглавие страницы.
- внутри тега **div** с классами **container-fluid page** находится весь основной контент сайта.

Полный список тегов с удобной группировкой — в руководстве: [HTML 5.2. теги: HTML5BOOK.RU](http://HTML5BOOK.RU).

Атрибуты тегов и их значения

У тега могут быть свойства — *атрибуты*. Они дают дополнительные возможности форматирования текста. Записываются в виде сочетания имени атрибута и значения, причём текстовые значения заключаются в кавычки.

Например, можно выделить фрагмент текста определённым шрифтом, используя тег ``, и указав в этом теге название шрифта **face** и желаемый размер **size**:

```
<font face="Times, Arial, Courier" size=4> Оформляемый текст </font>
```

Важно! Обращайте внимание на атрибуты: они помогают идентифицировать нужные вам теги и часто делают их уникальными, упрощая их поиск.

Атрибут class

С помощью классов часто задают стили для одного и более элементов, объединённых логическим смыслом (например, на нашем сайте это блоки с книгами). Для этого обычно используют один класс. Внутри тега `article` с классом `product_pod` находится информация о книгах:

```

::before
▼ <li class="col-xs-6 col-sm-4 col-md-3 col-lg-3">
  ▶ <article class="product_pod"> ... </article>
</li>
▼ <li class="col-xs-6 col-sm-4 col-md-3 col-lg-3">
  ▶ <article class="product_pod"> ... </article>
</li>
▼ <li class="col-xs-6 col-sm-4 col-md-3 col-lg-3">
  ▶ <article class="product_pod"> ... </article>
</li>
▼ <li class="col-xs-6 col-sm-4 col-md-3 col-lg-3">
  ▶ <article class="product_pod"> ... </article>
</li>

```

Важно! Один элемент может иметь **несколько** атрибутов класса! В этом случае их значения перечисляются через пробел. Например, тег `article` вложен в тег `li`, класса у которого целых четыре. В адаптивной вёрстке эти классы указывают на то, как будет выглядеть страница при изменении размера экрана.

Атрибут id

Идентификатор однозначно определяет **один** конкретный элемент. Значение `id` должно быть уникальным, на одной странице может встречаться только один раз. Благодаря этому атрибуту мы можем однозначно точно выбрать именно нужный нам элемент на странице. При этом класс у элемента может быть такой же, как и у других элементов на странице.

```

▼ <div class="col-sm-8 col-md-9"> прокручивание
  ▶ <div class="page-header action"> ... </div>
  ▶ <div id="messages"> ... </div>
  ▶ <div id="promotions"> ... </div>

```

Помимо этого, у тегов могут быть другие атрибуты, например, ссылка (`href`) или значение (`value`). По этим атрибутам также можно осуществлять парсинг, однако с ними надо быть особо внимательными. Значения, как и ссылка, могут меняться, поэтому, если вы ищете ссылку с содержанием определённого текста, будьте уверены, что этот текст не меняется при загрузке или обновлении страницы.

DOM

DOM (от англ. *Document Object Model* — объектная модель документа) — не зависящий от платформы и языка программный интерфейс, позволяющий программам и скриптам получить доступ к содержимому HTML-, XHTML- и XML-документов, а также менять содержимое, структуру и оформление таких документов.

Модель DOM не накладывает ограничений на структуру документа. Любой документ известной структуры с помощью DOM может быть представлен в виде дерева узлов, каждый узел которого может быть:

- элементом,
- атрибутом,
- текстовым, графическим или любым другим объектом.

Узлы связаны между собой отношениями «родительский-дочерний».

Изначально различные браузеры имели собственные модели документов (DOM), несовместимые с остальными. Для обеспечения взаимной и обратной совместимости специалисты международного консорциума W3C классифицировали эту модель по уровням, для каждого из которых была создана своя спецификация. Все эти спецификации объединены в общую группу — W3C DOM.

Уровни DOM

У модели DOM несколько уровней. Каждый новый расширяет функционал предыдущего:

Уровень 1 включает в себя поддержку XML 1.0 и HTML (управление деревом, перемещение по дереву) без пространства имён (этот уровень существовал ещё до разделения модели на уровни).

Уровень 2 поддерживает пространства имён XML и CSS.

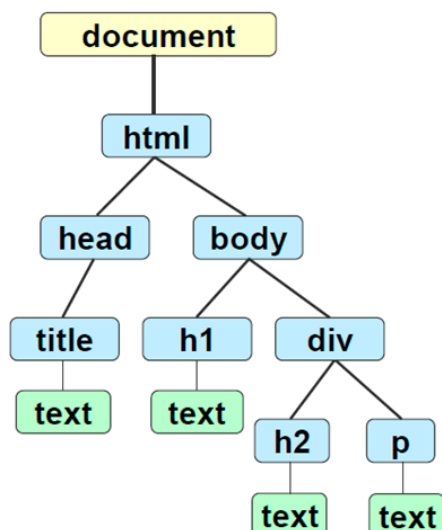
Уровень 3 состоит из 6 спецификаций:

- a. DOM Level 3 Core;
- b. DOM Level 3 Load and Save;
- c. DOM Level 3 XPath;
- d. DOM Level 3 Views and Formatting;
- e. DOM Level 3 Requirements;
- f. DOM Level 3 Validation.

Для лучшего понимания DOM рассмотрим пример простой HTML-страницы:

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>PAGE</title>
  </head>
  <body>
    <h1>
      PAGE
    </h1>
    <div>
      <h2>Раздел</h2>
      <p>Параграф</p>
    </div>
  </body>
</html>
```

В виде DOM она будет представлена так:



В представленном дереве два типа узлов:

- **теги** — узлы, за счёт которых выстроена структура дерева, его вложенность;
- **текст** — узлы, обозначенные как text, содержат исключительно строку текста.

При построении модели DOM учитываются и переносы строк, и пробелы.

XPath

XPath или XML Path — это язык запросов, который можно использовать для поиска узлов (элементов) в документах XML (Extensible Markup Language).

Языки разметки HTML и XML следуют аналогичным правилам структуры и формата. Поэтому XPath также можно использовать для запросов к HTML-документам.

Проще говоря, для поиска и обработки элементов в HTML-документах мы можем использовать особый синтаксис XPath, чтобы следовать структуре и иерархии страницы. Мы познакомимся с XPath поближе, когда будем говорить о Scrapy и Selenium — познакомимся и научимся использовать. В этом уроке у нас в основном теория.

В XPath есть семь типов узлов: элементы, атрибуты, текст, пространство имён, инструкции по обработке, комментарии и узлы документов. XML-документы рассматриваются как деревья узлов. Самый верхний элемент дерева называется корневым элементом.

Есть несколько видов связей между узлами: Parent, Children, Siblings, Ancestors, Descendant. Но обычно выделяют две главные связи:

- **Связь Parent** (родитель) — у каждого элемента и атрибута есть один родитель. Так, в следующем примере элемент `<article>` является родителем по отношению к элементам `<div>`, `<p>`, `<h3>` и `<div>`;

- **Связь Children** (ребёнок) — узлы элементов могут иметь ноль, одного или нескольких дочерних элементов. Так, элементы `<div>`, `<p>`, `<h3>` и `<div>` являются дочерними элементами элемента `<article>`.

```

▼ <li class="col-xs-6 col-sm-4 col-md-3 col-lg-3">
  ▼ <article class="product_pod">
    ▶ <div class="image_container"> ... </div>
    ▶ <p class="star-rating Three"> ... </p>
    ▶ <h3> ... </h3>
    ▶ <div class="product_price"> ... </div>
  </article>
</li>

```

При этом между собой эти элементы являются братьями, то есть Siblings.

Чтобы выбрать узлы в XML-документе, XPath использует выражения пути. Узел выбирается исходя из заданного пути. Наиболее полезные выражения пути:

Выражение	Результат
<i>имя узла</i>	Выбирает все узлы с именем «имя_узла»
/	Выбирает от корневого узла
//	Выбирает узлы от текущего узла, соответствующего выбору, независимо от их местонахождения
.	Выбирает текущий узел
..	Выбирает родителя текущего узла
@	Выбирает атрибуты

Давайте рассмотрим на примере. Возьмем все элементы с тегом `article` и из них достанем все названия книг, их обложку, цену и наличие с сайта [Books to Scrape](https://books.toscrape.com).

```

from pprint import pprint
from lxml import html
import requests
r = requests.get('https://books.toscrape.com')

root = html.fromstring(r.content)
articles = root.xpath («//article»)

for article in articles:
    title = article.xpath («//h3/a/@title») [0]
    image = article.xpath («//div[@class='image_container']/a/img/@src») [0]
    price = article.xpath («//p[@class='price_color']/text()») [0]
    instock = article.xpath ("//p[contains(@class, 'instock')]/text()")
    print(title, image, price, instock) [0]

```


Обратите внимание, что XPath возвращает список, поэтому мы забираем первый элемент в каждом списке. В уроках по Scrapy и Selenium мы подробнее рассмотрим синтаксис XPath. Если вы используете библиотеку Requests для парсинга, удобнее затем получать необходимую информацию с помощью библиотеки BeautifulSoup.

Обратите внимание: чтобы получить атрибут тега (например, атрибут title или src), мы обращаемся к нему через @. Когда же нам нужен текст тега, пишем текст и ставим круглые скобки. Также мы можем указывать название класса, которые содержится в классе тега. Например, в теге, который содержит информацию о наличии книги в магазине, есть два класса: instock availability. Мы указали, что нам нужен тег, который содержит класс instock. Для этого мы использовали метод contains, а внутри указали, что класс содержит название instock. Этот же метод contains можно использовать и для текста, например, когда вам нужен тег с определённым текстом внутри. В таком случае надо будет прописать: текст, круглые скобки, запятая, пробел и кусок текст, который должен быть внутри искомого тега.

Xpath очень удобен. Подробнее почитать о его использовании можно в статье — [«Руководство по поиску элементов с использованием XPath в Selenium Python»](#).