GeekBrains

Методы сбора и обработки данных при помощи Python

Парсинг HTML, XPath

На этом уроке

1. Узнаем, что такое открытые данные, для чего они нужны и как используются.
2. Научимся работать с CSV в Python и самостоятельно их создавать.

Оглавление

Введение

XPath (XML Path Language) — язык запросов к элементам XML-документа. Разработан для организации доступа к частям документа XML в файлах трансформации XSLT, стандарт консорциума W3C. XPath призван реализовать навигацию по DOM в XML. В нём используется компактный синтаксис, отличный от принятого в XML. В 2007 году завершилась разработка версии 2.0, которая теперь входит в состав языка XQuery 1.0. В декабре 2009 года началась разработка версии 2.1,

которая использует XQuery 1.1.

Сейчас самая популярная версия — XPath 1.0, потому что открытые библиотеки не поддерживают XPath 2.0. В частности, речь идёт о LibXML, от которой зависит поддержка языка в браузерах, с одной стороны, и поддержка со стороны серверного интерпретатора — с другой.

Основы

XML имеет древовидную структуру. В самостоятельном XML-документе всегда есть один корневой элемент (инструкция <?xml version="1.0"?> к дереву элементов отношения не имеет), в котором допустим ряд вложенных элементов, некоторые из них, в свою очередь, могут содержать вложенные элементы. Также могут встречаться текстовые узлы, комментарии и инструкции. Можно считать, что XML-элемент содержит массив вложенных в него элементов и массив атрибутов.

У элементов дерева бывают элементы-предки и элементы-потомки (у корневого элемента предков нет, а у тупиковых элементов (листьев дерева) нет потомков). Каждый элемент дерева находится на определённом уровне вложенности (далее — «уровень»). Элементы упорядочены так же, как расположены в тексте XML, и поэтому можно говорить об их предыдущих и следующих элементах. Это очень похоже на организацию каталогов в файловой системе.

Строка XPath описывает способ выбора нужных элементов из массива, которые могут содержать вложенные элементы. Начинается отбор с переданного множества элементов, на каждом шаге пути отбираются элементы, соответствующие выражению шага, и в результате оказывается отобрано подмножество элементов, соответствующих данному пути.

Для примера возьмём следующий HTML-документ:

```
1 <html>
2  <body>
3     <div>Первый слой
4       <span>блок текста в первом слое</span>
5     </div>
6     <div>Второй слой</div>
7     <div>Третий слой
8       <span class="text">первый блок в третьем слое</span>
9       <span class="text">второй блок в третьем слое</span>
10      <span>третий блок в третьем слое</span>
11     </div>
12     <span>четвёртый слой</span>
13     <img />
14  </body>
15 </html>
```

XPath-путь **/html/body/*/span[@class]** будет соответствовать в нём двум элементам исходного документа:

```
<span class="text">первый блок в третьем слое</span>
```

и

```
<span class="text">второй блок в третьем слое</span>
```

Элементы пути преимущественно пишутся в XPath в краткой форме. Полная форма приведённого выше пути имеет вид /child::html/child::body/child::*/child::span[attribute::class].

Чтобы выбрать узлы или наборы узлов в документе, XPath использует выражения пути. Они очень похожи на те, что используются в традиционных файловых системах. Путь состоит из шагов адресации, которые разделяются символом «косая черта»: /.

Каждый шаг адресации состоит из трёх частей:

- ось (по умолчанию child::, ось элементов). Кроме отбора по оси вложенных элементов, можно отбирать по другим осям элементов и по оси атрибутов (attribute::, она же обозначается символом @) (см. ниже);
- выражение, определяющее отбираемые элементы (в примере отбор делается по соответствию элементов документа именам html, body, span, и используется символ *, который отберёт все элементы оси);
- предикаты (в нашем примере это attribute::class) — дополнительные условия отбора. Их может быть несколько. Каждый предикат заключается в квадратные скобки и подразумевает логическое выражение для проверки отбираемых элементов. Если предиката нет, то отбираются все подходящие элементы.

Анализ пути ведётся слева направо и начинается в контексте первого элемента корневого узла. Если в начале XPath указан символ /, то анализ производится в контексте со всеми корневыми элементами переданного XML по оси child::. В первом случае (в примере это элемент html), по оси child:: будут вложенные в него элементы (в примере это один элемент body), что удобно при обработке обычного XML-документа с одним корневым узлом. Во втором случае, в примере, это будет один элемент html. На каждом шаге адресации в текущем контексте отбираются элементы, подходящие под указанные условия. Их перечень берётся как контекст для следующего шага или как возвращаемый результат.

Шаг 1. /child::html явным образом делает текущим контекстом для следующего шага перечень из одного элемента html, что было бы и так сделано неявно, если бы этот шаг не был обозначен.

Шаг 2. В нашем примере (шаг child::body) контекст — перечень из одного элемента html. Ось child:: говорит, что необходимо смотреть на имена вложенных элементов в текущем контексте. Условие проверки body говорит, что в формируемый набор элементов нужно включить те узлы, у которых имя body. Таким образом, в ходе второго шага адресации получаем набор узлов, состоящий всего из одного элемента body, который и становится контекстом для третьего шага.

Шаг 3. child::* . Ось child:: содержит всех непосредственных потомков элемента body, а условие проверки * говорит, что в формируемый перечень нужно включить элементы основного типа с любым именем. В ходе этого шага получаем перечень, состоящий из трёх элементов div, одного span и одного img — итого пять элементов.

Шаг 4. child::span/@class. Его контекст — перечень из пяти элементов, поэтому исходящий перечень создаётся в пять проходов (за пять итераций).

- при первой итерации узлом контекста становится первый div. Согласно заданной оси child:: и правилу проверки span, в набор должны включаться непосредственные потомки этого div, имя которых равно span. В данном случае такой один;
- при второй итерации в набор ничего добавляться не будет, так как у второго div нет потомков;
- третья итерация увидит сразу три элемента span;
- четвёртая ничего не увидит, так как у элемента span нет потомков span, а то, что он сам span — не важно, ведь просматриваются именно потомки;
- пятая тоже ничего не увидит, у элемента img нет потомков span. Итак, в ходе проверки мы могли бы получить набор узлов, состоящий из четырёх элементов span. Это было бы контекстом для последующей обработки, не будь на этом шаге указан предикат.

Так как предикат на четвёртом шаге есть, по мере выполнения каждого из пяти проходов отбираемые элементы пройдут дополнительную фильтрацию. В данном случае у предиката ось attribute:: говорит о необходимости проверить, есть ли у отбираемого узла атрибуты. Условие class требует оставить лишь те узлы, у которых задан атрибут с именем class. Поэтому на первой итерации единственный найденный span фильтрацию предикатом не пройдёт, на третьей итерации фильтрацию пройдут два элемента из трёх. В итоге, несмотря на то, что фильтрация происходит за пять итераций, в окончательный набор попадают только два элемента span.

Узлы

Всего в XPath выделяется семь типов узлов:

- элемент;
- атрибут;
- текст;
- пространство имён;
- инструкция обработки;
- комментарий;
- узлы документа.

Документ XML рассматривается как дерево узлов. Элемент, находящийся на самом верху этого дерева, называется корневым.

В качестве примера используем XML-документ:

```
<messages>
   <note id="1">
      <to>Джону</to>
      <from>Дженни</from>
      <heading>Напоминание</heading>
      <body>Купи хлеба!</body>
   </note>
</messages>
```

Здесь:

- <messages> — узел корневого элемента;
- <to>Джону</to> — узел элемента;
- note id="1" — узел атрибута;

Атомарные значения или узлы

Узлы, у которых нет родителей и потомков, называются атомарными значениями.

Пример:

```
Джону
"1"
```

Атомарные значения или узлы называются элементами данных.

Отношения узлов

Родитель

У каждого элемента есть один родитель.

В следующем примере элемент note — родитель to, from, heading и body:

```
<note>
   <to>Джону</to>
   <from>Дженни</from>
   <heading>Напоминание</heading>
   <body>Купи хлеба!</body>
</note>
```

Прямые потомки

У элемента могут присутствовать или отсутствовать прямые потомки.

В следующем примере элементы to, from, heading и body — прямые потомки элемента note:

```
<note>
   <to>Джону</to>
   <from>Дженни</from>
   <heading>Напоминание</heading>
   <body>Купи хлеба!</body>
</note>
```

Элементы данных одного уровня

Узлы, у которых один и тот же родитель, будут элементами данных одного уровня.

В следующем примере to, from, heading и body — элементы данных одного уровня:

```
<note>
   <to>Джону</to>
   <from>Дженни</from>
   <heading>Напоминание</heading>
   <body>Купи хлеба!</body>
</note>
```

Предки

Предками называются родители узлов, родители родителей и т. д.

В следующем примере предки элемента to — note и messages:

```
<messages>
   <note>
      <to>Джону</to>
      <from>Дженни</from>
      <heading>Напоминание</heading>
      <body>Купи хлеба!</body>
   </note>
</messages>
```

Потомки

Потомками называются прямые потомки узлов, прямые потомки прямых потомков и т. д.

В следующем примере потомки элемента messages — note, to, from, heading и body:

```
<messages>
   <note>
      <to>Джону</to>
      <from>Дженни</from>
      <heading>Напоминание</heading>
      <body>Купи хлеба!</body>
   </note>
</messages>
```

Синтаксис Xpath

Выбор узлов

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

Выражение	Результат
имя узла	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

В следующей таблице приводятся некоторые выражения XPath, позволяющие сделать выборки по демонстрационному XML-документу:

Выражение XPath	Результат
messages	Выбирает все узлы с именем messages
/messages	Выбирает корневой элемент сообщений **Примечание**: Если путь начинается с косой черты (/), то он всегда представляет абсолютный путь к элементу!
messages/note	Выбирает все элементы note — потомки элемента messages
//note	Выбирает все элементы note независимо от того, где в документе они находятся
messages//note	Выбирает все элементы note — потомки элемента messages независимо от того, где относительно него они находятся
//@date	Выбирает все атрибуты с именем date

Предикаты

Предикаты позволяют найти конкретный узел или узел с конкретным значением.

Предикаты всегда заключаются в квадратные скобки.

В следующей таблице приводятся некоторые выражения XPath с предикатами, позволяющими сделать выборки по демонстрационному XML-документу:

Выражение XPath	Результат
/messages/note[1]	Выбирает первый элемент note, который является прямым потомком элемента messages. **Примечание**: В IE 5,6,7,8,9 первым узлом будет [0], однако согласно W3C это должен быть [1]. Чтобы решить эту проблему в IE, нужно установить опцию SelectionLanguage в значение XPath. В JavaScript: xml.setProperty("SelectionLanguage","XPath");
/messages/note[last()]	Выбирает последний элемент note — прямой потомок элемента messages.
/messages/note[last()-1]	Выбирает предпоследний элемент note, который является прямым потомком элемента messages.
/messages/note[position()=2]	Выбирает второй по счету элемент note, который является прямым потомком элемента messages.
//heading[@date]	Выбирает все элементы heading, у которых есть атрибут date
//heading[@date="10/01/2008"]	Выбирает все элементы heading, у которых есть атрибут date со значением 10/01/2008

Выбор неизвестных заранее узлов

Чтобы найти неизвестные заранее узлы XML-документа, XPath позволяет использовать специальные символы.

Спецсимвол	Описание
*	Соответствует любому узлу элемента
@*	Соответствует любому узлу атрибута
node()	Соответствует любому узлу любого типа

В следующей таблице приводятся некоторые выражения XPath со спецсимволами, позволяющие сделать выборки по демонстрационному XML-документу:

Выражение XPath	Результат

/messages/*	Выбирает все элементы — прямые потомки элемента messages
//*	Выбирает все элементы в документе
//heading[@*]	Выбирает все элементы heading, у которых есть по крайней мере один атрибут любого типа

Выбор нескольких путей

Использование оператора | в выражении XPath позволяет делать выбор по нескольким путям.

В следующей таблице приводятся некоторые выражения XPath, позволяющие сделать выборки по демонстрационному XML-документу:

Выражение XPath	Результат
//note/heading | //note/body	Выбирает все элементы heading и body из всех элементов note
//heading | //body	Выбирает все элементы heading и body во всем документе

Прочие обозначения в XPath

Обозначение	Описание
*	Обозначает *любое* имя или набор символов по указанной оси, например: * — любой дочерний узел; @* — любой атрибут
$name	Обращение к переменной. name — имя переменной или параметра
[]	Дополнительные условия выборки (или предикат шага адресации). Должен содержать логическое значение. Если содержит числовое, считается, что это порядковый номер узла, что эквивалентно приписыванию перед этим числом выражения position()=
{}	Если применяется внутри тега другого языка (например, HTML), то XSLT-процессор рассматривает содержимое фигурных скобок как XPath
/	Определяет уровень дерева, т. е. разделяет шаги адресации
|	Объединяет результат. Т. е., в рамках одного пути можно написать несколько путей разбора через знак |, и в результат такого выражения войдёт всё, что будет найдено любым из этих путей

Оси XPath

Ось определяет отношение узлового набора по отношению к текущему узлу.

Название оси	Результат
ancestor	Выбирает всех предков текущего узла
ancestor-or-self	Выбирает всех предков текущего узла и сам текущий узел
attribute	Выбирает все атрибуты текущего узла
child	Выбирает всех прямых потомков текущего узла
descendant	Выбирает всех потомков текущего узла
descendant-or-self	Выбирает всех потомков текущего узла и сам текущий узел
following	Выбирает все элементы в документе после закрывающего тега текущего узла
following-sibling	Выбирает все элементы одного уровня после текущего узла
namespace	Выбирает все узлы пространства имен текущего узла
parent	Выбирает родителя текущего узла
preceding	Выбирает все узлы, которые появляются перед текущим узлом, за исключением предков, узлов атрибутов и пространства имен
preceding-sibling	Выбирает все элементы одного уровня до текущего узла
self	Выбирает текущий узел

Выражение пути выборки

Путь выборки может быть абсолютным или относительным. Абсолютный путь выборки начинается с символа косой черты (/), а относительный — сразу с шага. В обоих случаях такой путь содержит один или несколько шагов, разделённых косой чертой:

Абсолютный путь:

/шаг/шаг/...

Относительный путь:

шаг/шаг/...

Каждый шаг вычисляется по отношению к узлам в текущем узловом наборе.

Шаг содержит:

- ось (определяет древовидное отношение между выбранными узлами и текущим узлом);
- проверку узла (определяет узел внутри оси);
- ноль или больше предикатов (для более точного указания выбранного узлового набора).

Синтаксис для шага выборки имеет следующий вид:

имяОси::проверкаУзла[предикат]

Примеры

Пример	Результат
child::note	Выбирает все узлы note — прямые потомки текущего узла
attribute::date//ul/li[@class='list__item']/a/text()	Выбирает атрибут date текущего узла
child::*	Выбирает всех прямых потомков текущего узла
attribute::*	Выбирает все атрибуты текущего узла
child::text()	Выбирает все текстовые узлы текущего узла
child::node()	Выбирает всех прямых потомков текущего узла
descendant::note	Выбирает всех потомков note текущего узла
ancestor::note	Выбирает всех предков note текущего узла
ancestor-or-self::note	Выбирает всех предков note текущего узла, а также сам текущий узел, если это узел note
child::*/child::heading	Выбирает всех прямых потомков прямых потомков («внуков») heading текущего узла

Рассмотрим использование XPath на практике. В нашем примере будем использовать результат поисковой выдачи Яндекса по запросу «шляпа». Причём нас будут интересовать именно «живые» ссылки, без Яндекс.Картинок и Яндекс.Маркета. Исследуя элементы на странице, увидим, что сами ссылки на ресурсы находятся под тегом «a». Возьмём два класса: link_cropped_no и organic__url_type_multilane. Используем эти данные для составления программы. Сначала подключим в Python библиотеки pprint, lxml (метод html) и request, затем создадим запрос, используя выбранные классы для XPath. После исполнения кода программа выдаст нам все соответствующие результаты со страницы Яндекса.

```
1 from pprint import pprint
2 from lxml import html
3 import requests
4 import time
```

```
5 header = {'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.130 Safari/537.36'}
6 def request_to_yandex(str):
7         try:
8             time.sleep(1)
9             response = requests.get('https://yandex.ru/search/',
10                          params={'text':str},
11                          headers = header
12                         )
13             root = html.fromstring(response.text)
14             result = root.xpath(
15                 "//a[contains(@class,'link_cropped_no')]/@href |
//a[contains(@class,'organic__url_type_multiline')]/@href"
16                 )
17          return result
18          except:
19          print('Ошибка запроса')
20 for i in range(50):
21     result = request_to_yandex('Шляпа')
22 pprint(result)
```

Теперь сравним программы сбора данных с использованием Beautiful Soup (рассматривался в уроке №2) и с использованием XPath. Возьмём страницу https://www.kinopoisk.ru/afisha/new/city/458/. Сначала подключим те же библиотеки, что и в примере с Яндексом. Список фильмов содержится внутри div-контейнера с классом filmsListNew js-rum-hero. В нём находятся div-контейнеры с тегом «a», с атрибутом href, в которых прописаны сами ссылки на фильмы. У каждого фильма есть класс item, по нему мы будем фильтровать. Фактически, код работает так: он проходит по списку фильмов ("//div[@class='item']") и берёт значение (".//div[@class='name']/a/@href"). Обратите внимание, что в начале выражения стоит точка, это сделано, чтобы поиск не выходил за пределы текущего каталога. Добавим в код вывод атрибутов: name, rating, genre. После запуска на исполнение программа выдаст список из 10 фильмов с указанием названия, ссылки на фильм, рейтинга и жанра.

```
1 from pprint import pprint
2 from lxml import html
3 import requests
4 main_link = 'https://www.kinopoisk.ru'
5 response = requests.get(main_link+'/afisha/new/city/458/')
6 root = html.fromstring(response.text)
7 films = root.xpath("//div[@class='item']")
8 for film in films:
9             href = film.xpath(".//div[@class='name']/a/@href")
10             name = film.xpath(".//div[@class='name']/a/text()")
11             genre =
film.xpath('.//div[@class="gray"][last()]/text()')[2].replace(' ','')
12                         )
13             root = html.fromstring(response.text)
14             result = root.xpath(
15                 "//a[contains(@class,'link_cropped_no')]/@href |
//a[contains(@class,'organic__url_type_multiline')]/@href"
16          rating = film.xpath('.//div[@class="rating"]/span/text()')
17          print(name, href, rating, genre)
```


Глоссарий

XPath (XML Path Language) — язык запросов к элементам XML-документа. Разработан для организации доступа к частям документа XML в файлах трансформации XSLT.

Предикат — выражение, использующее одну или более величину с результатом булева типа.

Абсолютный путь выборки — отсчитывается от корневого узла дерева.

Относительный путь выборки — отсчитывается от контекстного узла.

Дополнительные материалы

1. Синтаксис XPath.
2. Спецификация XPath на русском языке.
3. XPath примеры - шпаргалка для разбора страниц.

Домашнее задание

1. Написать приложение, которое собирает основные новости с сайтов

- https://news.mail.ru,
- https://lenta.ru,
- https://yandex.ru/news.

Для парсинга использовать XPath. Структура данных должна содержать:

- название источника;
- наименование новости;
- ссылку на новость;
- дата публикации.

2. Сложить собранные данные в БД

Используемая литература

1. Википедия (XPath).
2. Книга «Изучаем XPath».
3. Синтаксис XPath