Information Technologies

Работа с API веб-ресурсов с помощью XML ¶

In [ ]:

```
from bs4 import BeautifulSoup
```

API и XML

Анализируя веб-страницы и извлекая из них информацию мы пытаемся написать программу, которая бы действовала как человек. Это бывает непросто. К счастью, всё чаще разнообразные сайты предлагают информацию, которую может легко обрабатывать не только человек, но и другая программа. Это называется API — application program interface. Обычный интерфейс — это способ взаимодействия человека с программой, а API — одной программы с другой. Например, вашего скрипта на Python с удалённым веб-сервером.

Для хранения веб-страниц, которые читают люди, используется язык HTML. Для хранения произвольных структурированных данных, которыми обмениваются между собой программы, используются другие языки — в частности, язык XML, похожий на HTML. Вернее было бы сказать, что XML это *метаязык*, то есть способ описания языков. В отличие от HTML, набор тегов в XML-документе может быть произвольным (и определяется разработчиком конкретного диалекта XML). Например, если бы мы хотели описать в виде XML некоторую студенческую группу, это могло бы выглядеть так:

```
<group>
    <number>ПИ/б-18-1-о</number>
    <student>
        <firstname>Виталий</firstname>
        <lastname>Иванов</lastname>
    </student>
    <student>
        <firstname>Мария</firstname>
        <lastname>Петрова</lastname>
    </student>
</group>
```

Для обработки XML-файлов можно использовать тот же пакет *Beautiful Soup*, который мы уже использовали для работы с HTML. Единственное различие — нужно указать дополнительный параметр `features="xml"` при вызове функции `BeautifulSoup` — чтобы он не искал в документе HTML-теги.

Если параметр `features="xml"` приводит к ошибке, то нужно установить пакет `lxml` . Для этого нужно вызвать окно Anaconda Prompt и выполнить команду `pip install lxml`

In [ ]:

```
group = """<group>
<number>ПИ/б-18-1-о</number>
<student>
<firstname>Виталий</firstname>
<lastname>Иванов</lastname>
</student>
<student>
<firstname>Мария</firstname>
<lastname>Петрова</lastname>
</student>
</group>"""
```

In [ ]:

```
obj = BeautifulSoup(group, features="xml")
print(obj.prettify())
```

Вот так мы можем найти в нашем XML-документе номер группы:

In [ ]:

```
obj.group.number.string
```

Это значит «в объекте `obj` найти тег `group` в нём найти тег `number` и выдать в виде строки то, что в нём содержится.

А вот так можно перечислить всех студентов:

In [ ]:

```
for student in obj.group.findAll('student'):
    print(student.lastname.string, student.firstname.string)
```

Получаем список статей из категории в Википедии

Допустим, нам потребовалось получить список всех статей из некоторой категории в Википедии. Мы могли бы открыть эту категорию в браузере и дальше действовать теми методами, которые обсуждались выше. Однако в Википедии существует удобное API. Чтобы научиться с ним работать, придётся познакомиться с документацией (https://www.mediawiki.org/wiki/API:Main_page) (так будет с любым API), но это кажется сложным только в первый раз.

Итак, приступим. Взаимодействие с сервером при помощи API происходит с помощью отправки специальным образом сформированных запросов и получения ответа в одном из машинночитаемых форматов. Нас будет интересовать формат XML, хотя бывают и другие (позже мы познакомимся с JSONN). А вот такой запрос мы можем отправить:

https://en.wikipedia.org/w/api.php?action=query&list=categorymembers&cmtitle=Category:Physics&cmsort=timestamp&cmdir=desc&format=xmlf
(https://en.wikipedia.org/w/api.php?action=query&list=categorymembers&cmtitle=Category:Physics&cmsort=timestamp&cmdir=desc&format=xmlf

Строка `https://en.wikipedia.org/w/api.php` (до знака вопроса) — это *точка входа* в API. Всё, что идёт после знака вопроса — это, собственно, запрос. Он представляет собой что-то вроде словаря и состоит из пар «ключ=значение», разделяемых амперсандом `&`. Некоторые символы приходится кодировать специальным образом.

Например, в адресе выше сказано, что мы хотим сделать запрос (`action=query`), перечислить элементы категории `list=categorymembers`, в качестве категории, которая нас интересует, указана `Category:Physics` (`cmtitle=Category:Physics`) и указаны некоторые другие параметры. Если кликнуть по этой ссылке, откроется примерно такая штука:

```
<?xml version="1.0"?>
<api batchcomplete="">
  <continue cmcontinue="2015-05-30 19:37:50|1653925" continue="-||" />
  <query>
    <categorymembers>
      <cm pageid="24293838" ns="0" title="Wigner rotation" />
      <cm pageid="48583145" ns="0" title="Northwest Nuclear Consortium" />
      <cm pageid="48407923" ns="0" title="Hume Feldman" />
      <cm pageid="48249441" ns="0" title="Phase Stretch Transform" />
      <cm pageid="47723069" ns="0" title="Epicatalysis" />
      <cm pageid="2237966" ns="14" title="Category:Surface science" />
      <cm pageid="2143601" ns="14" title="Category:Interaction" />
      <cm pageid="10844347" ns="14" title="Category:Physical systems" />
      <cm pageid="18726608" ns="14" title="Category:Physical quantities" />
      <cm pageid="22688097" ns="0" title="Branches of physics" />
    </categorymembers>
  </query>
</api>
```

Мы видим здесь разные теги, и видим, что нас интересуют теги `<cm>`, находящиеся внутри тега `<categorymembers>`.

Давайте сделаем соответствующий запрос с помощью Python. Для этого нам понадобится уже знакомый модуль `requests`.

In [ ]:

```
import requests
```

In [ ]:

```
url = "https://en.wikipedia.org/w/api.php"
params = {
    'action':'query',
    'list':'categorymembers',
    'cmtitle': 'Category:Physics',
    'format': 'xml'
}

g = requests.get(url, params=params)
```

Как видно, список параметров мы передаем в виде обычного словаря. Посмотрим, что получилось.

In [ ]:

```
g.ok
```

Всё хорошо. Теперь используем *Beautiful Soup* для обработки этого XML.

In [ ]:

```
data = BeautifulSoup(g.text, features='xml')
```

In [ ]:

```
print(data.prettify())
```

Найдём все вхождения тега `<cm>` и выведем их атрибут `title`:

In [ ]:

```
for cm in data.api.query.categorymembers("cm"):
    print(cm['title'])
```

Можно было упростить поиск `<cm>`, не указывая «полный путь» к ним:

In [ ]:

```
for cm in data("cm"):
    print(cm['title'])
```

По умолчанию сервер вернул нам список из 10 элементов. Если мы хотим больше, нужно воспользоваться элементом `continue` — это своего рода гиперссылка на следующие 10 элементов.

In [ ]:

```
data.find("continue")['cmcontinue']
```

Нам пришлось использовать метод `find()` вместо того, чтобы просто написать `data.continue`, потому что `continue` в Python имеет специальный смысл.

Теперь добавим `cmcontinue` в наш запрос и выполним его ещё раз:

In [ ]:

```
params['cmcontinue'] = data.api("continue")[0]['cmcontinue']
```

In [ ]:

```
g = requests.get(url, params=params)
data = BeautifulSoup(g.text, features='xml')
for cm in data.api.query.categorymembers("cm"):
    print(cm['title'])
```

Мы получили следующие 10 элементов из категории. Продолжая таким образом, можно выкачать её даже целиком (правда, для этого потребуется много времени).

Аналогичным образом реализована работа с разнообразными другими API, имеющимися на разных сайтах. Где-то API является полностью открытым (как в Википедии), где-то вам потребуется зарегистрироваться и получить application id и какой-нибудь ключ для доступа к API, где-то попросят даже заплатить (например, автоматический поиск в Google стоит что-то вроде 5 долларов за 100 запросов). Есть API, которые позволяют только читать информацию, а бывают и такие, которые позволяют её править. Например, можно написать скрипт, который будет автоматически сохранять какую-то информацию в Google Spreadsheets. Всякий раз при использовании API вам придётся изучить его документацию, но это в любом случае проще, чем обрабатывать HTML-код. Иногда удаётся упростить доступ к API, используя специальные библиотеки.