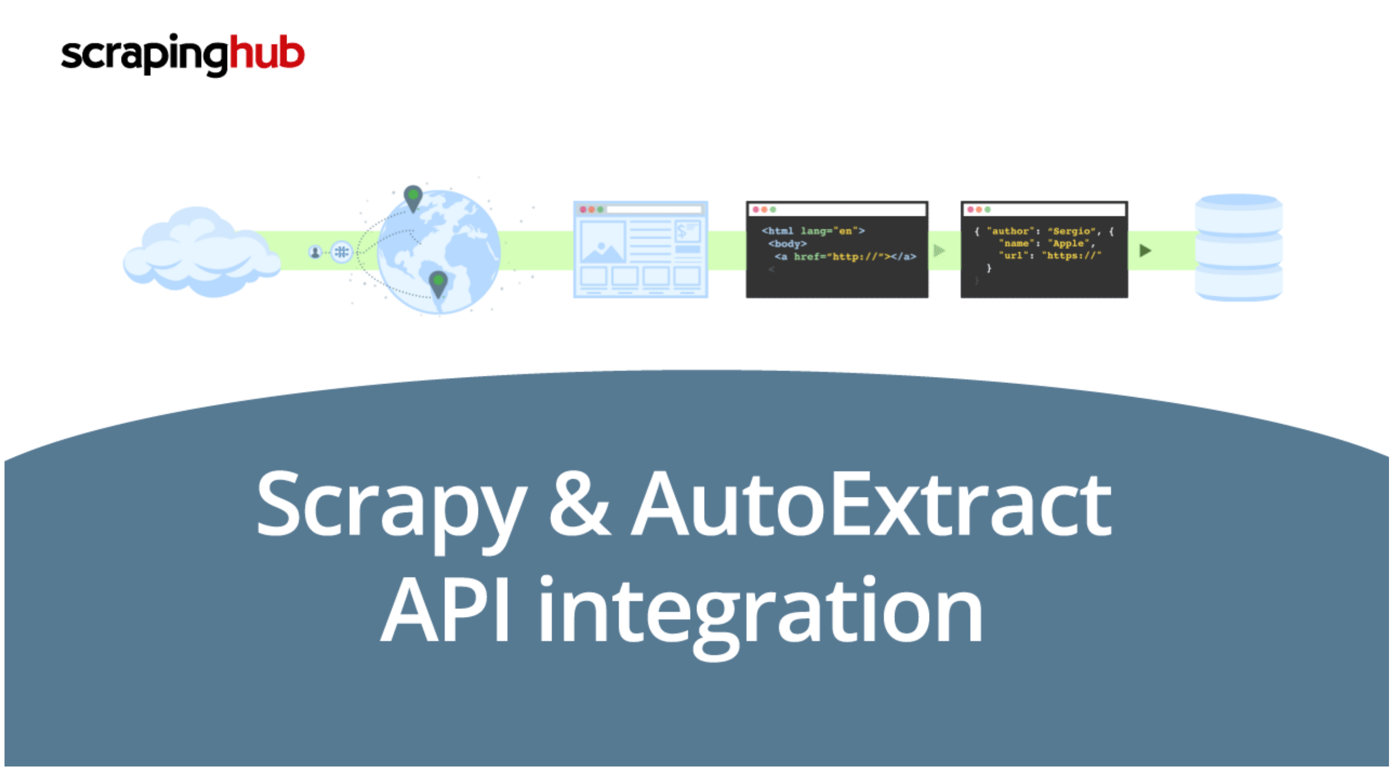
THE SCRAPINGHUB BLOG

Turn Web Content Into Useful Data



SCRAPY & AUTOEXTRACT API INTEGRATION

October 15, 2019 Attila Tóth 0 Comments

We've just released a new open-source Scrapy middleware which makes it easy to integrate AutoExtract into your existing Scrapy spider. If you haven't heard about AutoExtract yet, it's an AI-based web scraping tool which automatically extracts data from web pages without the need to write any code. Learn more about AutoExtract here.

Installation

This project uses Python 3.6+ and pip. A virtual environment is strongly encouraged.

```
$ pip install git+https://github.com/scrapinghub/scrapy-autoextract
```

Configuration

Enable middleware

```
DOWNLOADER_MIDDLEWARES = {
    'scrapy_autoextract.AutoExtractMiddleware': 543,
}
```

This middleware should be the last one to be executed so make sure to give it the highest value.

AutoExtract settings

Mandatory

These settings must be defined in order for AutoExtract to work.

- AUTOEXTRACT_USER: your AutoExtract API key
- AUTOEXTRACT_PAGE_TYPE: the kind of data to be extracted (current options: "product" or "article")

Optional

- AUTOEXTRACT_URL: AutoExtract service url (default: autoextract.scrapinghub.com)
- AUTOEXTRACT_TIMEOUT: response timeout from AutoExtract (default: 660 seconds)

Spider

AutoExtract requests are opt-in and they must be enabled for each request, by adding:

```
meta['autoextract'] = {'enabled': True}
```

If the request was sent to AutoExtract, inside your Scrapy spider you can access the AutoExtract result through the meta attribute:

```
def parse(self, response):
    yield response.meta['autoextract']
```

Example

In the Scrapy settings file:

```
DOWNLOADER_MIDDLEWARES = {
'scrapy_autoextract.AutoExtractMiddleware': 543,
}

# Disable AutoThrottle middleware
AUTHTHROTTLE_ENABLED = False

AUTOEXTRACT_USER = 'my_autoextract_apikey'
AUTOEXTRACT_PAGE_TYPE = 'article'
```

In the spider:

```
class ExampleSpider(Spider):
    name = 'example'
    start_urls = ['example.com']

    def start_requests(self):
        yield scrapy.Request(url, meta={'autoextract': {'enabled': True}}, self.p

    def parse(self, response):
        yield response.meta['autoextract']
```

Example output:

```
[{
      "query":{
         "domain":"example.com",
         "userQuery":{
            "url":"https://www.example.com/news/2019/oct/15/lorem-dolor-sit",
            "pageType":"article"
         },
         "id":"1570771884892-800e44fc7cf49259"
      },
      "article":{
         "articleBody":"Lorem ipsum dolor sit amet, consectetur adipiscing elit,
         "description":"Duis aute irure dolor in reprehenderit in voluptate velit
         "probability":0.9717171744215637,
         "inLanguage":"en",
         "headline":"'Lorem Ipsum Dolor Sit Amet",
         "author":"Attila Toth",
         "articleBodyHtml":"<article>\n\n<p>Lorem ipsum...",
         "images":["https://i.example.com/img/media/12a71d2200e99f9fff125972b88f
         "mainImage":"https://i.example.com/img/media/12a71d2200e99f9fff125972b88
}]
```

Limitations

- The incoming spider request is rendered by AutoExtract, not just downloaded by Scrapy, which can change the result - the IP is different, headers are different, etc.
- Only GET requests are supported
- Custom headers and cookies are not supported (i.e. Scrapy features to set them don't work)
- Proxies are not supported (they would work incorrectly, sitting between Scrapy and AutoExtract, instead of AutoExtract and website)
- AutoThrottle extension can work incorrectly for AutoExtract requests, because AutoExtract timing can be much larger than the time required to download a page, so it's best to use AUTOTHROTTLE_ENABLED=False in the settings.
- Redirects are handled by AutoExtract, not by Scrapy, so these kinds of middlewares might have no effect
- Retries should be disabled, because AutoExtract handles them internally (use RETRY_ENABLED=False in the settings) There is an exception, if there are too many requests sent in a short amount of time and AutoExtract returns HTTP code 429. For that case it's best to use RETRY_HTTP_CODES=[429].

Check out the middleware on Github or learn more about AutoExtract!

Share this:

Tweet Share **Like** 28 **Share**

Related

Extracting clean article HTML with News API
March 12, 2020
In "Autoscraping" , "data extraction" , "AutoExtract" , "News Data Extraction"

Job Postings Beta API: Extract Job Postings at Scale
March 05, 2020
In "Web Scraping" , "Autoscraping" , "data extraction" , "Developer API" , "AutoExtract" , "Jobs Data"

Building spiders made easy: GUI For Your Scrapy Shell
March 03, 2020
In "Open source" , "Scrapy" , "spider"

Scrapy, AutoExtract

‹NEXT
Web Scraping Questions & Answers Part II

PREVIOUS ›
Web Scraping Questions & Answers Part I

Leave a Reply

First Name*

Last Name

Email*

Website

Comment*

☐ I would like to receive email updates from Scrapinghub on blog posts, products, news, events and offers. You may unsubscribe at any time.

☐ I have read and agree to Scrapinghub's **Privacy Policy** and **Cookie Policy**. *

protected by **reCAPTCHA**
Privacy - Terms

SUBMIT COMMENT

KEEP UP TO DATE WITH WEB SCRAPING AND DATA TIPS...

Email*

protected by **reCAPTCHA**
Privacy - Terms

SIGN ME UP

Story of the Month

HOW TO SCRAPE THE WEB WITHOUT GETTING BLOCKED

Getting data from publicly available websites should not be a problem. In reality though, it’s not that easy. We can make it easy.

Search …

CRAWL WEB DATA AT SCALE WITHOUT BOTTLENECKS OR SLOWDOWNS.

Start your Free Trial

FOLLOW US

POPULAR POSTS

Learn how to configure and utilize proxies with Python Requests module

An Introduction to XPath: How to Get Started

Handling JavaScript in Scrapy with Splash

How to Build your own Price Monitoring Tool

How to Crawl the Web Politely with Scrapy

RECENT POSTS

- Transitioning to Remote Working as a Company
- A Practical Guide to Web Data QA Part I: Validation Techniques
- COVID-19: Handling the Situation as a Fully Remote Company
- Extracting clean article HTML with News API
- Job Postings Beta API: Extract Job Postings at Scale

CATEGORIES

Select Category ▼

ARCHIVES

Select Month ▼