goose3 Documentation

Release 3.1.12

maintainers

Oct 28, 2022

Contents:

CHAPTER 1

Goose3 API

1.1 Goose3

class goose3.**Goose**(*config: Union[goose3.configuration.Configuration, dict] = None*)
Extract most likely article content and aditional metadata from a URL or previously fetched HTML document

Parameters **config** (Configuration, *dict*) – A configuration file or dictionary representation of the configuration file

Returns An instance of the goose extraction object

Return type *Goose*

close()
Close the network connection and perform any other required cleanup

Note: Auto closed when using goose as a context manager or when garbage collected

extract(*url: str = None*, *raw_html: str = None*) → goose3.article.Article
Extract the most likely article content from the html page

Parameters

- **url** (*str*) – URL to pull and parse
- **raw_html** (*str*) – String representation of the HTML page

Returns Representation of the article contents including other parsed and extracted metadata

Return type *Article*

shutdown_network()
Close the network connection

Note: Auto closed when using goose as a context manager or when garbage collected

1.2 Configuration

Configuration options to change how and what goose3 extracts and parses.

class goose3.**Configuration**

available_parsers
A list of all possible parser values for the parser_class

Note: Not settable

Type list(str)

browser_user_agent
Browser user agent string to use when making URL requests

Note: Defaults to *Goose/{goose3.__version__}*

Examples

Using the non-standard browser agent string is advised when pulling frequently

```
>>> config.browser_user_agent = 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_
↪2)'
>>> config.browser_user_agent = 'AppleWebKit/534.52.7 (KHTML, like Gecko)'
>>> config.browser_user_agent = 'Version/5.1.2 Safari/534.52.7'
```

enable_image_fetching
Turn on or off image extraction

Note: Defaults to *False*

Type bool

get_parser() → Union[goose3.parsers.Parser, goose3.parsers.ParserSoup, any]
Retrieve the current parser class to use for extraction

Returns The parser to use

Return type Parser

http_auth
Authentication class and information to pass to the requests library

See also:

Requests Authentication

Type tuple

http_headers
Custom headers to pass directly to the supporting *requests* object

See also:

Requests Custom Headers

Type dict

http_proxies
Proxy information to pass directly to the supporting *requests* object

See also:

Requests Proxy Support

Type dict

http_timeout
The time delay to pass to *requests* to wait for the response in seconds

Note: Defaults to 30.0

Type float

imagemagick_convert_path
Path to the convert program that is part of imagemagick

Note: Defaults to *"/opt/local/bin/convert"*

Warning: Currently not used / implemented

Type str

imagemagick_identify_path
Path to the identify program that is part of imagemagick

Note: Defaults to *"/opt/local/bin/identify"*

Warning: Currently not used / implemented

Type str

images_min_bytes
Minimum number of bytes for an image to be evaluated to be the main image of the site

Note: Defaults to 4500 bytes

Type int

keep_footnotes
Specify if footnotes should be kept or not in the cleaned_text output

Note: Defaults to *True*

Type bool

known_author_patterns
The tags to search to find the likely published date

Note: Each entry must be a dictionary with the following keys: *attribute*, *value*, and *content*.

Type list

known_context_patterns
The context patterns to search to find the likely article content

Note: Each entry must be a dictionary with the following keys: *attr* and *value* or just *tag*

Type list

known_publish_date_tags
The tags to search to find the likely published date

Note: Each entry must be a dictionary with the following keys: *attribute*, *value*, and *content*.

Type list

local_storage_path
The local path to store temporary files

Note: Defaults to the value of *os.path.join(tempfile.gettempdir(), 'goose')*

Type str

parse_headers
Specify if headers should be pulled or not in the cleaned_text output

Note: Defaults to *True*

Type bool

parser_class
The key of the parser to use

Note: Defaults to *lxml*

Type str

pretty_lists
Specify if lists should be pretty printed in the cleaned_text output

Note: Defaults to *True*

Type bool

stopwords_class
The StopWords class to use when analyzing article content

Note: Defaults to the english stop words

Note: Current stop words available in *goose3.text* include:

StopWords, *StopWordsChinese*, *StopWordsArabic*, and *StopWordsKorean*

Type StopWords

strict
Enable *strict mode* and throw exceptions instead of swallowing them.

Note: Defaults to *True*

Type bool

target_language
The default target language if the language is not extractable or if use_meta_language is set to False

Note: Default language is 'en'

Type str

use_meta_language
Determine if language should be extracted from the meta tags or not. If this is set to *False* then the target_language will be used. Also, if extraction fails then the target_language will be utilized.

Note: Defaults to *True*

Type bool

1.2.1 Configuration Helper Classes

class goose3.configuration.**ArticleContextPattern**(*, *attr=None*, *value=None*, *tag=None*, *domain=None*)

Help ensure correctly generated article context patterns

Parameters

- **attr** (*str*) – The attribute type: class, id, etc
- **value** (*str*) – The value of the attribute
- **tag** (*str*) – The type of tag, such as *article* that contains the main article body
- **domain** (*str*) – The domain to which this pattern pertains (optional)

Note: Must provide, at a minimum, (attr and value) or (tag)

---

class goose3.configuration.**AuthorPattern**(*, *attr=None*, *value=None*, *content=None*, *tag=None*, *subpattern=None*)

Ensures that the author patterns are correctly formed for use with the *known_author_patterns* of configuration

Parameters

- **attr** (*str*) – The attribute type: class, id, etc
- **value** (*str*) – The value of the attribute
- **content** (*str*) – The name of another attribute (of the element) that contains the value
- **tag** (*str*) – The type of tag, such as *author* that contains the author information
- **subpattern** (*str*) – A subpattern for elements within the main attribute

class goose3.configuration.**PublishDatePattern**(*, *attr=None*, *value=None*, *content=None*, *subcontent=None*, *tag=None*, *domain=None*)

Ensure correctly formed publish date patterns; to be used in conjuntion with the configuration *known_publish_date_tags* property

Parameters

- **attr** (*str*) – The attribute type: class, id, etc
- **value** (*str*) – The value of the attribute
- **content** (*str*) – The name of another attribute (of the element) that contains the value
- **subcontent** (*str*) – The name of a json object key (optional)
- **tag** (*str*) – The type of tag, such as *time* that contains the publish date
- **domain** (*str*) – The domain to which this pattern pertains (optional)

Note: Must provide, at a minimum, (attr and value) or (tag)

1.3 Article

The result of a goose3 extraction is to return an Article object that contains the results of the parsing process.

class goose3.**Article**

additional_data
A property bucket for consumers of goose3 to store custom data extractions

Note: Read only

Type dict

authors
A listing of authors as parsed from the meta tags

Note: Read only

Type list(str)

canonical_link
The canonical link of the article if found in the meta data

Note: Read only

Type str

cleaned_text
Cleaned text of the article without HTML tags; most commonly desired property

Note: Read only

Type str

doc
lxml document that is being processed

Note: Read only

Type etree

domain
Domain of the article parsed

Note: Read only

Type str

final_url
The URL that was used to pull and parsed; *None* if raw_html was used and no url element was found.

Note: Read only

Type str

infos
The summation of all data available about the extracted article

Note: Read only

Type dict

link_hash
The hash of the final url to be used for various identification tasks

Note: Read only

Type str

links
A listing of URL links within the article

Note: Read only

Type list(str)

meta_description
Contents of the meta-description field from the HTML source

Note: Read only

Type str

meta_encoding
Contents of the encoding/charset field from the HTML source

Note: Read only

Type str

meta_favicon
Contents of the meta-favicon field from the HTML source

Note: Read only

Type str

meta_keywords
Contents of the meta-keywords field from the HTML source

Note: Read only

Type str

meta_lang
Contents of the meta-lang field from the HTML source

Note: Read only

Type str

movies
A listing of all videos within the article such as YouTube or Vimeo

Returns See more information on the goose3.Video class

Return type list(*Video*)

Note: Read only

Type list(*Video*)

opengraph
All opengraph tag data

Note: Read only

Type dict

publish_date
The date the article was published based on meta tag extraction

Note: Read only

Type str

publish_datetime_utc
The date time version of the published date based on meta tag extraction in the UTC timezone, if timezone information is known

Note: Read only

Type datetime.datetime

raw_doc
Original, uncleaned, and untouched lxml document to be processed

Note: Read only

Type etree

raw_html
The HTML represented as a string

Note: Read only

Type str

schema
All schema tag data

Note: Read only

Type dict

tags
List of article tags (non-metadata tags)

Note: Read only

Type list(str)

title
Title extracted from the HTML source

Note: Read only

Type str

top_image
The top image object that likely represents the article

Returns See more information on the goose3.Image class

Return type *Image*

Note: Read only

Type *Image*

top_node
The top Element that is a candidate for the main body of the article

Note: Read only

Type etree

tweets
A listing of embeded tweets in the article

Note: Read only

Type list(str)

1.4 Image

class goose3.**Image**

bytes
The size of the image in bytes

Note: Read only

Type int

confidence_score
The confidence score that this is the main image

Note: Read only

Type float

extraction_type
The extraction type used

Note: Read only

Type str

height
The image height in pixels

Note: Read only

Type int

src
Source URL for the image

Note: Read only

Type str

top_image_node
The most likely top image element node

Note: Read only

Type etree

width
The image width in pixels

Note: Read only

Type int

1.5 Video

class goose3.**Video**
Video object

embed_code
The embed code of the video

Note: Read only

Type str

embed_type
The type of embeding such as embed, object, or iframe

Note: Read only

Type str

height
The video height in pixels

Note: Read only

Type int

provider
The video provider

Note: Read only

Type str

src
The URL source of the video

Note: Read only

Type str

width
The video width in pixels

Note: Read only

Type int

CHAPTER 2

Quickstart

2.1 Install

Goose3 is a python3 fork of the python-goose library. To use goose3, one must run everything using python3. All python commands assume the usage of the correct python version.

2.1.1 Using pip

The easiest way to install goose3 is to use pip:

```
$ pip install goose3
```

2.1.2 From source

To install from source simply clone the repository on GitHub, then run the following command from the extracted folder:

```
$ python setup.py install
```

2.2 Setup

Setting up Goose3 using the standard configuration is fairly straight forward:

```
from goose3 import Goose

g = Goose()
article = g.extract(url='http://this-url.html')
print(article.cleaned_text)
g.close()
```

For extracting lots of HTML files or URLs, one can also use it as a context manager:

```
from goose3 import Goose

urls = [...]
with Goose() as g:
    for tmp in urls:
        article = g.extract(url=tmp)
        print(article.cleaned_text)
```

2.3 Setting Config Options

One can also alter how goose3 performs the extraction and what items are extracted by passing a configuration to Goose. There are several ways to set the configuration options.

For more details on available configuration settings, see *Configuration*

2.3.1 Use Configuration object

```
from goose3 import Goose
from goose3.configuration import Configuration

config = Configuration()
config.strict = False  # turn of strict exception handling
config.browser_user_agent = 'Mozilla 5.0'  # set the browser agent string
config.http_timeout = 5.05  # set http timeout in seconds

with Goose(config) as g:
    ...
```

2.3.2 Use Dictionary

One can pass in a dictionary with keys that match the configuration properties one would like to change:

```
from goose3 import Goose

config = {}
config['strict'] = False  # turn of strict exception handling
config['browser_user_agent'] = 'Mozilla 5.0'  # set the browser agent string
config['http_timeout'] = 5.05  # set http timeout in seconds

with Goose(config) as g:
    pass
```

Or if there are only a few changes:

```
from goose3 import Goose

with Goose({'http_timeout': 5.0}) as g:
    pass
```

2.3.3 After Object Creation

One can also change configuration options after the Goose object has been created:

```
from goose3 import Goose

g = Goose()
g.config.browser_user_agent = 'Mozilla 5.0'
```

2.3.4 Configuration Helper Classes

For some, more complex configuration options, there are classes available to help ensure that the correct values are provided. One does not need to use the provided classes, but it does make things a bit simpler.

```
from goose3 import Goose
from goose3.configuration import Configuration, ArticleContextPattern,␣
↪PublishDatePattern, AuthorPattern

config = Configuration()

# we know of a particular article location in the site we are pulling from
config.known_context_patterns = ArticleContextPattern(attr="id", value="my-site-
↪article")

# publish date
config.known_publish_date_tags = PublishDatePattern(attr="id", value="pubdate",␣
↪content="content")

# author
config.known_author_patterns = AuthorPattern(attr="id", value="writer", content=
↪"content")
```

2.4 Reading Results

Results from the extraction are returned as an Article object. Reading the desired results is as simple as reading the desired property. The most commonly asked for property is *cleaned_text* which holds the non-html formatted text of the extracted article.

For more details and for all available properties, see *Article*

```
from goose3 import Goose

urls = [...]
with Goose() as g:
    for tmp in urls:
        article = g.extract(url=tmp)
        print(article.cleaned_text)
```

2.5 Using with PyInstaller

It should be possible to use `goose3` with tools such as PyInstaller to add goose to your executable program. To do so, you will need to add the required resources to the executable.

You will need to add the files to a folder in your executable called **goose3/resources/** to match the location that `goose3` checks for the required files.

```
pyinstaller \
    --add-data="goose3/resources/images/*;goose3/resources/images/" \
    --add-data="goose3/resources/text/*;goose3/resources/text/" \
    my_prog.py
```

CHAPTER 3

Indices and tables

- *Goose3 API*
- *Quickstart*
- genindex
- modindex
- search

Index