Ultimate Sitemap Parser Documentation

Release 0.5

Linas Valiukas, Hal Roberts, Media Cloud project

Mar 17, 2020

Contents

CHAPTER 1

usp

1.1 usp package

1.1.1 Subpackages

usp.objects package

Submodules

usp.objects.page module

Objects that represent a page found in one of the sitemaps.

usp.objects.page.**SITEMAP_PAGE_DEFAULT_PRIORITY = Decimal('0.5')**
Default sitemap page priority, as per the spec.

class usp.objects.page.**SitemapNewsStory**(*title: str*, *publish_date: datetime.datetime*, *publication_name: Optional[str] = None*, *publication_language: Optional[str] = None*, *access: Optional[str] = None*, *genres: List[str] = None*, *keywords: List[str] = None*, *stock_tickers: List[str] = None*)

Bases: object

Single story derived from Google News XML sitemap.

access
Return accessibility of the article.

Returns Accessibility of the article.

genres
Return list of properties characterizing the content of the article.

Returns genres such as “PressRelease” or “UserGenerated”.

Returns List of properties characterizing the content of the article

keywords
Return list of keywords describing the topic of the article.

Returns List of keywords describing the topic of the article.

publication_language
Return primary language of the news publication in which the article appears in.

It should be an ISO 639 Language Code (either 2 or 3 letters).

Returns Primary language of the news publication in which the article appears in.

publication_name
Return name of the news publication in which the article appears in.

Returns Name of the news publication in which the article appears in.

publish_date
Return story publication date.

Returns Story publication date.

stock_tickers
Return list of up to 5 stock tickers that are the main subject of the article.

Each ticker must be prefixed by the name of its stock exchange, and must match its entry in Google Finance. For example, "NASDAQ:AMAT" (but not "NASD:AMAT"), or "BOM:500325" (but not "BOM:RIL").

Returns List of up to 5 stock tickers that are the main subject of the article.

title
Return story title.

Returns Story title.

class usp.objects.page.**SitemapPage**(*url: str*, *priority: decimal.Decimal = Decimal('0.5')*, *last_modified: Optional[datetime.datetime] = None*, *change_frequency: Optional[usp.objects.page.SitemapPageChangeFrequency] = None*, *news_story: Optional[usp.objects.page.SitemapNewsStory] = None*)

Bases: object

Single sitemap-derived page.

change_frequency
Return change frequency of a sitemap URL.

Returns Change frequency of a sitemap URL.

last_modified
Return date of last modification of the URL.

Returns Date of last modification of the URL.

news_story
Return Google News story attached to the URL.

Returns Google News story attached to the URL.

priority
Return priority of this URL relative to other URLs on your site.

Returns Priority of this URL relative to other URLs on your site.

url
Return page URL.

Returns Page URL.

class usp.objects.page.**SitemapPageChangeFrequency**
Bases: enum.Enum

Change frequency of a sitemap URL.

ALWAYS = 'always'

DAILY = 'daily'

HOURLY = 'hourly'

MONTHLY = 'monthly'

NEVER = 'never'

WEEKLY = 'weekly'

YEARLY = 'yearly'

has_value = <bound method SitemapPageChangeFrequency.has_value of <enum 'SitemapPageCha

usp.objects.sitemap module

Objects that represent one of the found sitemaps.

class usp.objects.sitemap.**AbstractIndexSitemap**(*url: str, sub_sitemaps: List[usp.objects.sitemap.AbstractSitemap]*)
Bases: *usp.objects.sitemap.AbstractSitemap*

Abstract sitemap with URLs to other sitemaps.

all_pages() → Iterator[usp.objects.page.SitemapPage]
Return iterator which yields all pages of this sitemap and linked sitemaps (if any).

Returns Iterator which yields all pages of this sitemap and linked sitemaps (if any).

sub_sitemaps
Return sub-sitemaps that are linked to from this sitemap.

Returns Sub-sitemaps that are linked to from this sitemap.

class usp.objects.sitemap.**AbstractPagesSitemap**(*url: str, pages: List[usp.objects.page.SitemapPage]*)
Bases: *usp.objects.sitemap.AbstractSitemap*

Abstract sitemap that contains URLs to pages.

all_pages() → Iterator[usp.objects.page.SitemapPage]
Return iterator which yields all pages of this sitemap and linked sitemaps (if any).

Returns Iterator which yields all pages of this sitemap and linked sitemaps (if any).

pages
Return list of pages found in a sitemap.

Returns List of pages found in a sitemap.

class usp.objects.sitemap.**AbstractSitemap**(*url: str*)
Bases: object

Abstract sitemap.

all_pages() → Iterator[usp.objects.page.SitemapPage]
Return iterator which yields all pages of this sitemap and linked sitemaps (if any).

Returns Iterator which yields all pages of this sitemap and linked sitemaps (if any).

url
Return sitemap URL.

Returns Sitemap URL.

class usp.objects.sitemap.**IndexRobotsTxtSitemap**(*url: str, sub_sitemaps: List[usp.objects.sitemap.AbstractSitemap]*)
Bases: *usp.objects.sitemap.AbstractIndexSitemap*

robots.txt sitemap with URLs to other sitemaps.

class usp.objects.sitemap.**IndexWebsiteSitemap**(*url: str, sub_sitemaps: List[usp.objects.sitemap.AbstractSitemap]*)
Bases: *usp.objects.sitemap.AbstractIndexSitemap*

Website's root sitemaps, including robots.txt and extra ones.

class usp.objects.sitemap.**IndexXMLSitemap**(*url: str, sub_sitemaps: List[usp.objects.sitemap.AbstractSitemap]*)
Bases: *usp.objects.sitemap.AbstractIndexSitemap*

XML sitemap with URLs to other sitemaps.

class usp.objects.sitemap.**InvalidSitemap**(*url: str*, *reason: str*)
Bases: *usp.objects.sitemap.AbstractSitemap*

Invalid sitemap, e.g. the one that can't be parsed.

all_pages() → Iterator[usp.objects.page.SitemapPage]
Return iterator which yields all pages of this sitemap and linked sitemaps (if any).

Returns Iterator which yields all pages of this sitemap and linked sitemaps (if any).

reason
Return reason why the sitemap is deemed invalid.

Returns Reason why the sitemap is deemed invalid.

class usp.objects.sitemap.**PagesAtomSitemap**(*url: str, pages: List[usp.objects.page.SitemapPage]*)
Bases: *usp.objects.sitemap.AbstractPagesSitemap*

RSS 0.3 / 1.0 sitemap that contains URLs to pages.

class usp.objects.sitemap.**PagesRSSSitemap**(*url: str, pages: List[usp.objects.page.SitemapPage]*)
Bases: *usp.objects.sitemap.AbstractPagesSitemap*

RSS 2.0 sitemap that contains URLs to pages.

class usp.objects.sitemap.**PagesTextSitemap**(*url: str, pages: List[usp.objects.page.SitemapPage]*)
Bases: *usp.objects.sitemap.AbstractPagesSitemap*

Plain text sitemap that contains URLs to pages.

class usp.objects.sitemap.**PagesXMLSitemap**(*url: str, pages: List[usp.objects.page.SitemapPage]*)

Bases: *usp.objects.sitemap.AbstractPagesSitemap*

XML sitemap that contains URLs to pages.

Module contents

usp.web_client package

Submodules

usp.web_client.abstract_client module

Abstract web client class.

class usp.web_client.abstract_client.**AbstractWebClient**

Bases: object

Abstract web client to be used by the sitemap fetcher.

get(*url: str*) → usp.web_client.abstract_client.AbstractWebClientResponse

Fetch an URL and return a response.

Method shouldn't throw exceptions on connection errors (including timeouts); instead, such errors should be reported via Response object.

Parameters **url** – URL to fetch.

Returns Response object.

set_max_response_data_length(*max_response_data_length: int*) → None

Set the maximum number of bytes that the web client will fetch.

Parameters **max_response_data_length** – Maximum number of bytes that the web client will fetch.

class usp.web_client.abstract_client.**AbstractWebClientResponse**

Bases: object

Abstract response.

class usp.web_client.abstract_client.**AbstractWebClientSuccessResponse**

Bases: *usp.web_client.abstract_client.AbstractWebClientResponse*

Successful response.

header(*case_insensitive_name: str*) → Optional[str]

Return HTTP header value for a given case-insensitive name, or None if such header wasn't set.

Parameters **case_insensitive_name** – HTTP header's name, e.g. "Content-Type".

Returns HTTP header's value, or None if it was unset.

raw_data() → bytes

Return encoded raw data of the response.

Returns Encoded raw data of the response.

status_code() → int

Return HTTP status code of the response.

Returns HTTP status code of the response, e.g. 200.

status_message() → str
Return HTTP status message of the response.

Returns HTTP status message of the response, e.g. "OK".

usp.web_client.abstract_client.**RETRYABLE_HTTP_STATUS_CODES = {400, 408, 429, 499, 500, 502,**
HTTP status codes on which a request should be retried.

class usp.web_client.abstract_client.**WebClientErrorResponse**(*message: str*, *retryable: bool*)
Bases: *usp.web_client.abstract_client.AbstractWebClientResponse*

Error response.

message() → str
Return message describing what went wrong.

Returns Message describing what went wrong.

retryable() → bool
Return True if request should be retried.

Returns True if request should be retried.

usp.web_client.requests_client module

requests-based implementation of web client class.

class usp.web_client.requests_client.**RequestsWebClient**
Bases: *usp.web_client.abstract_client.AbstractWebClient*

requests-based web client to be used by the sitemap fetcher.

get(*url: str*) → usp.web_client.abstract_client.AbstractWebClientResponse
Fetch an URL and return a response.

Method shouldn't throw exceptions on connection errors (including timeouts); instead, such errors should be reported via Response object.

Parameters **url** – URL to fetch.

Returns Response object.

set_max_response_data_length(*max_response_data_length: int*) → None
Set the maximum number of bytes that the web client will fetch.

Parameters **max_response_data_length** – Maximum number of bytes that the web client will fetch.

set_proxies(*proxies: Dict[str, str]*) → None
Set proxies from dictionnary where:

- keys are schemes, e.g. "http" or "https";
- values are "scheme://user:password@host:port/".

For example:

proxies = {'http': 'http://user:pass@10.10.1.10:3128/'}

set_timeout(*timeout: int*) → None
Set HTTP request timeout.

class usp.web_client.requests_client.**RequestsWebClientErrorResponse**(*message: str*, *retryable: bool*)

Bases: *usp.web_client.abstract_client.WebClientErrorResponse*

requests-based error response.

class usp.web_client.requests_client.**RequestsWebClientSuccessResponse**(*requests_response: requests.models.Response*, *max_response_data_length: Optional[int] = None*)

Bases: *usp.web_client.abstract_client.AbstractWebClientSuccessResponse*

requests-based successful response.

header(*case_insensitive_name: str*) → Optional[str]

Return HTTP header value for a given case-insensitive name, or None if such header wasn't set.

Parameters **case_insensitive_name** – HTTP header's name, e.g. "Content-Type".

Returns HTTP header's value, or None if it was unset.

raw_data() → bytes

Return encoded raw data of the response.

Returns Encoded raw data of the response.

status_code() → int

Return HTTP status code of the response.

Returns HTTP status code of the response, e.g. 200.

status_message() → str

Return HTTP status message of the response.

Returns HTTP status message of the response, e.g. "OK".

Module contents

1.1.2 Submodules

1.1.3 usp.exceptions module

Exceptions used by the sitemap parser.

exception usp.exceptions.**GunzipException**

Bases: Exception

gunzip() exception.

exception usp.exceptions.**SitemapException**

Bases: Exception

Problem due to which we can't run further, e.g. wrong input parameters.

exception usp.exceptions.**SitemapXMLParsingException**
Bases: Exception

XML parsing exception to be handled gracefully.

exception usp.exceptions.**StripURLToHomepageException**
Bases: Exception

strip_url_to_homepage() exception.

1.1.4 usp.tree module

Helpers to generate a sitemap tree.

usp.tree.**sitemap_tree_for_homepage**(*homepage_url: str*, *web_client: Optional[usp.web_client.abstract_client.AbstractWebClient] = None*) → usp.objects.sitemap.AbstractSitemap

Using a homepage URL, fetch the tree of sitemaps and pages listed in them.

Parameters

- **homepage_url** – Homepage URL of a website to fetch the sitemap tree for, e.g. "http://www.example.com/".
- **web_client** – Web client implementation to use for fetching sitemaps.

Returns Root sitemap object of the fetched sitemap tree.

1.1.5 Module contents

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

u

Index