finscraper

Release 0.2.3

Jesse Myrberg

Sep 10, 2023

DOCUMENTATION

The library provides an easy-to-use API for fetching data from various Finnish websites:

Website	Type	Spider API class
Ilta-Sanomat	News article	`ISArticle`
Iltalehti	News article	`ILArticle`
YLE Uutiset	News article	`YLEArticle`
Suomi24	Discussion thread	`Suomi24Page`
Vauva	Discussion thread	`VauvaPage`
Oikotie Asunnot	Apartment ad	`OikotieApartment`
Tori	Item deal	`ToriDeal`

Documentation is available at https://finscraper.readthedocs.io and simple online demo here.

CHAPTER

ONE

INSTALLATION

```
pip install finscraper
```

CHAPTER

TWO

QUICKSTART

Fetch 10 news articles as a pandas DataFrame from Ilta-Sanomat:

```
from finscraper.spiders import ISArticle

spider = ISArticle().scrape(10)

articles = spider.get()
```

The API is similar for all the spiders:

CHAPTER

THREE

CONTRIBUTING

Please see CONTRIBUTING.md for more information.

Jesse Myrberg (jesse.myrberg@gmail.com)

3.1 finscraper

The library provides an easy-to-use API for fetching data from various Finnish websites:

Website	Type	Spider API class
Ilta-Sanomat	News article	`ISArticle`
Iltalehti	News article	`ILArticle`
YLE Uutiset	News article	`YLEArticle`
Suomi24	Discussion thread	`Suomi24Page`
Vauva	Discussion thread	`VauvaPage`
Oikotie Asunnot	Apartment ad	`OikotieApartment`
Tori	Item deal	`ToriDeal`

Documentation is available at https://finscraper.readthedocs.io and simple online demo here.

3.1.1 Installation

```
pip install finscraper
```

3.1.2 Quickstart

Fetch 10 news articles as a pandas DataFrame from Ilta-Sanomat:

```
from finscraper.spiders import ISArticle

spider = ISArticle().scrape(10)

articles = spider.get()
```

The API is similar for all the spiders:

3.1.3 Contributing

Please see CONTRIBUTING.md for more information.

Jesse Myrberg (jesse.myrberg@gmail.com)

3.2 Installation

You can install finscraper and its dependencies from PyPi with:

```
pip install finscraper
```

To enjoy all the spiders of the library, install Google Chrome, which is used for rendering Javascript on some websites.

Note: Under the hood, finscraper uses Scrapy for web scraping, and Selenium together with Chrome and ChromeDriver for Javascript rendering. finscraper tries to find and download a suitable driver for your operating system automatically, but in case of issues, you may do this manually as well.

In case of issues, please check the installation details of the core dependencies:

- Scrapy
- Selenium
- ChromeDriver

3.3 Spiders

3.3.1 Classes

The main interface of finscraper is the *Spider API*, which aims to make web scraping as easy as possible. Each class represents specific content that can be scraped with it, and no more.

Note: If you need more flexibility in terms of the content to be scraped, you might want to implement your spiders with the lower level library Scrapy.

3.3.2 Parameters

Some spiders may have their own parameters which typically describe the content to be crawled. All spiders also have common parameters, such as `jobdir`, `log_level` and `progress_bar`. These are mainly for controlling the verbosity and working directories of the spiders.

3.3.3 Scraping

The `scrape` -method starts making HTTP requests and parsing desired items. In most cases, the spider keeps track of already visited pages to avoid fetching the same content more than once.

The workflow of a typical spider in pseudocode would look something like:

```
1. Start from a certain URL

2. While there are links to follow
    a) Find new links to follow
    b) Find links with desired content and parse them
    c) Stop, if a condition is met (e.g. number of scraped items)
```

Warning: Some websites try to prevent bots from crawling them, which could lead to a **ban of your IP address**. By default, finscraper obeys website-specific crawling rules (robots.txt) to avoid this.

3.3.4 Results

Scraped items and the state of the spider are saved into your hard disk as defined by `jobdir`. By default, this is your temporary directory.

The items are saved in *jsonline* -format into a single file at `spider.items_save_path`, and when calling the *get* -method, the data is read from the disk and returned as a pandas.DataFrame.

3.3.5 Save & load

You may *save* and *load* spiders to continue scraping later on. Moreover, you may use *clear* -method to reset the state and scraped items of an existing spider.

3.4 Usage

Fetch 10 articles from Iltalehti with *ILArticle* with the help of *scrape* and *get* -methods:

```
from finscraper.spiders import ILArticle

spider = ILArticle()
spider.scrape(10)
articles = spider.get()
```

Use *save* and *load* to continue scraping later on:

```
save_dir = spider.save()
spider = ILArticle.load(save_dir)
articles = spider.scrape(10).get()  # 20 articles in total
```

Items are fetched into `spider.jobdir` -directory which **is destroyed together with the `spider` -object unless *`spider.save()`* have been called**.

Because Scrapy is used under the hood, any of its settings can be passed into *`scrape`* -method. For example, to limit the number of concurrent requests per domain:

```
from finscraper.spiders import ILArticle

settings = {'CONCURRENT_REQUESTS_PER_DOMAIN': 1}

spider = ILArticle().scrape(10, settings=settings)
articles = spider.get()
```

3.5 Contributing

When websites change, spiders tend to break. I can't make a promise to keep this repository up-to-date all by myself, so pull requests are more than welcome!

3.5.1 Adding a new spider

1. Create a branch for the spider, e.g. `mtvarticle`
2. Add the spider name in pytest.ini and SPIDERS -file
3. Add the spider in tests/test_spiders.py similar to others
4. Add the spider API in finscraper/spiders.py
5. Write the Scrapy spider under finscraper/scrapy_spiders by naming the spider exactly with the spider name, e.g. `mtvarticle.py` - use of flake8 linting and Google style docstrings is recommended
6. Make sure the spider passes all non-benchmark tests within test_spiders.py
7. Push your branch into Github and make a pull request against master
8. Repository admin will merge to master and create a release after a review

3.5.2 Updating an existing spider

Steps 5. - 8. above.

3.6 finscraper package

3.6.1 Subpackages

finscraper.scrapy_spiders package

Submodules

finscraper.scrapy_spiders.ilarticle module

Module for ILArticle spider.

finscraper.scrapy_spiders.isarticle module

Module for ISArticle spider.

finscraper.scrapy_spiders.mixins module

Module for Scrapy spider mixins.

class finscraper.scrapy_spiders.mixins.**FollowAndParseItemMixin**(*follow_meta=None*, *items_meta=None*, *follow_selenium_callback=False*, *items_selenium_callback=False*)

Bases: `object`

Parse items and follow links based on defined link extractors.

The following needs to be defined when inheriting:

1) `item_link_extractor` -attribute: LinkExtractor that defines the links to parse items from.
2) `follow_link_extractor` -attribute: LinkExtractor that defines the links to follow and find item pages from.
3) `parse_item` -function: Parses the item from response.

Parameters

- **follow_meta** (*dict or None, optional*) – Dictionary to pass within link follow requests. Defaults to None.
- **follow_items** (*dict or None, optional*) – Dictionary to pass within item link requests. Defaults to None.
- **follow_selenium_callback** (*function, bool or None, optional*) – Selenium callback to use for follow requests. If function, takes in parameters (request, spider, driver) and returns response. If None, follows the default behavior of `SeleniumCallbackRequest`. If False, uses normal Scrapy `Request`. Defaults to None.
- **items_selenium_callback** (*function, bool or None, optional*) – Selenium callback to use for item requests. If function, takes in parameters (request, spider, driver) and returns response. If None, follows the default behavior of `SeleniumCallbackRequest`. If False, uses normal Scrapy `Request`. Defaults to None.

Raises
AttributeError, if required attributes not defined when inheriting. –

itemcount = 0

parse(*resp*, *to_parse=False*)

Parse items and follow links based on defined link extractors.

start_requests()

finscraper.scrapy_spiders.oikotieapartment module

Module for OikotieApartment spider.

finscraper.scrapy_spiders.suomi24page module

Module for Suomi24Page spider.

finscraper.scrapy_spiders.torideal module

Module for ToriDeal spider.

finscraper.scrapy_spiders.vauvapage module

Module for VauvaPage spider.

finscraper.scrapy_spiders.ylearticle module

Module for YLEArticle spider.

Module contents

3.6.2 Submodules

3.6.3 finscraper.extensions module

Module for Scrapy extensions.

class finscraper.extensions.**ProgressBar**(*crawler*)

Bases: object

Scrapy extension that displays progress bar.

Enabled via PROGRESS_BAR_ENABLED Scrapy setting.

classmethod from_crawler(*crawler*)

on_error(*failure*, *response*, *spider*)

on_item_scraped(*item*, *spider*)

on_response(*response*, *request*, *spider*)

spider_closed(*spider*)

spider_opened(*spider*)

3.6.4 finscraper.middlewares module

Module for Scrapy middlewares.

class finscraper.middlewares.**SeleniumCallbackMiddleware**(*settings*)

Bases: object

Middleware that processes request with given callback.

Headless mode can be disabled via DISABLE_HEADLESS Scrapy setting. In non-headless mode, window can be minimized via MINIMIZE_WINDOW Scrapy setting.

classmethod from_crawler(*crawler*)

process_request(*request*, *spider*)

spider_closed(*spider*)

spider_opened(*spider*)

3.6.5 finscraper.pipelines module

Module for Scrapy pipelines.

class finscraper.pipelines.**DefaultValueNonePipeline**

Bases: object

Pipeline that sets default values of all item fields to None.

process_item(*item*, *spider*)

3.6.6 finscraper.request module

Module for custom Scrapy request components.

class finscraper.request.**SeleniumCallbackRequest**(**args*, ***kwargs*)

Bases: Request

Process request with given callback using Selenium.

Parameters

selenium_callback (*func or None, optional*) – Function that will be called with the chrome webdriver. The function should take in parameters (request, spider, driver) and return request, response or None. If None, driver will be used for fetching the page, and return is response. Defaults to None.

attributes: Tuple[str, ...] = ('url', 'callback', 'method', 'headers', 'body', 'cookies', 'meta', 'encoding', 'priority', 'dont_filter', 'errback', 'flags', 'cb_kwargs')

A tuple of `str` objects containing the name of all public attributes of the class that are also keyword parameters of the `__init__` method.

Currently used by `Request.replace()`, `Request.to_dict()` and `request_from_dict()`.

property body: bytes

property cb_kwargs: dict

copy() → Request

property encoding: str

classmethod from_curl(*curl_command: str*, *ignore_unknown_options: bool = True*, ***kwargs*) → RequestTypeVar

Create a Request object from a string containing a cURL command. It populates the HTTP method, the URL, the headers, the cookies and the body. It accepts the same arguments as the `Request` class, taking preference and overriding the values of the same arguments contained in the cURL command.

Unrecognized options are ignored by default. To raise an error when finding unknown options call this method by passing `ignore_unknown_options=False`.

> **Caution:** Using *from_curl()* from `Request` subclasses, such as `JSONRequest`, or `XmlRpcRequest`, as well as having downloader middlewares and spider middlewares enabled, such as `DefaultHeadersMiddleware`, `UserAgentMiddleware`, or `HttpCompressionMiddleware`, may modify the `Request` object.

To translate a cURL command into a Scrapy request, you may use curl2scrapy.

property meta: dict

replace(**args*, ***kwargs*) → Request

Create a new Request with the same attributes except for those given new values

to_dict(*, *spider: Optional[Spider] = None*) → dict

Return a dictionary containing the Request's data.

Use `request_from_dict()` to convert back into a `Request` object.

If a spider is given, this method will try to find out the name of the spider methods used as callback and errback and include them in the output dict, raising an exception if they cannot be found.

property url: str

3.6.7 finscraper.settings module

Module for finscraper's default Scrapy settings.

3.6.8 finscraper.spiders module

Module for Spider API - the main interface of finscraper.

class finscraper.spiders.**ILArticle**(*jobdir=None*, *progress_bar=True*, *log_level=None*)

Bases: _SpiderWrapper

Fetch Iltalehti news articles.

Parameters

- **jobdir** (*str or None, optional*) – Working directory of the spider. Defaults to None, which creates a temp directory to be used. Note that this directory will only be deleted through the clear method!
- **progress_bar** (*bool, optional*) – Whether to enable progress bar or not. This parameter is ignored if log_level is not None. Defaults to True.
- **log_level** (*str or None, optional*) – Logging level to display. Should be in ['debug', 'info', 'warn', 'error', 'critical'] or None (disabled). Defaults to None.

Note: This parameter can be overridden through Scrapy settings (LOG_LEVEL, LOG_ENABLED) within the scrape -method.

Returned fields:

- url (str): URL of the scraped web page.
- time (int): UNIX timestamp of the scraping.
- title (str): Title of the article.
- ingress (str): Ingress of the article.
- content (str): Contents of the article.
- published (str): Publish time of the article.
- author (str): Author of the article.
- images (list of dict): Images of the article.

clear()

Clear contents of jobdir.

get(*fmt='df'*)

Return scraped data as DataFrame or list.

Parameters

fmt (*str, optional*) – Format to return parsed items as. Should be in ['df', 'list']. Defaults to 'df'.

Returns

If fmt = 'df', DataFrame of scraped items. If fmt = 'list', list of dict of scraped items.

Raises
ValueError – If `fmt` not in allowed values.

`property items_save_path`

Save of path of the scraped items.

Cannot be changed after initialization of a spider.

`property jobdir`

Working directory of the spider.

Can be changed after initialization of a spider.

`classmethod load(`*jobdir*`)`

Load existing spider from `jobdir`.

Parameters
jobdir (*str*) – Path to job directory.

Returns
Spider loaded from job directory.

`property log_level`

Logging level of the spider.

This attribute can be changed after initialization of a spider.

`property progress_bar`

Whether progress bar is enabled or not.

Can be changed after initialization of a spider.

`save()`

Save spider in `jobdir` for later use.

Returns
Path to job directory.

Return type
str

`scrape(`*n=10*, *timeout=60*, *settings=None*`)`

Scrape given number of items.

Parameters

- **n** (*int, optional*) – Number of items to attempt to scrape. Zero corresponds to no limit. Defaults to 10.
- **timeout** (*int, optional*) – Timeout in seconds to wait before stopping the spider. Zero corresponds to no limit. Defaults to 60.
- **settings** (*dict or None, optional*) – Scrapy spider settings to use. Defaults to None, which correspond to default settings. See list of available settings at: https://docs.scrapy.org/en/latest/topics/settings.html.

Returns
self

`property spider_save_path`

Save path of the spider.

Cannot be changed after initialization of a spider.

class finscraper.spiders.**ISArticle**(*jobdir=None*, *progress_bar=True*, *log_level=None*)

Bases: _SpiderWrapper

Fetch IltaSanomat news articles.

Parameters

- **jobdir** (*str or None, optional*) – Working directory of the spider. Defaults to None, which creates a temp directory to be used. Note that this directory will only be deleted through the clear method!
- **progress_bar** (*bool, optional*) – Whether to enable progress bar or not. This parameter is ignored if log_level is not None. Defaults to True.
- **log_level** (*str or None, optional*) – Logging level to display. Should be in ['debug', 'info', 'warn', 'error', 'critical'] or None (disabled). Defaults to None.

Note: This parameter can be overridden through Scrapy settings (LOG_LEVEL, LOG_ENABLED) within the scrape -method.

Returned fields:

- url (str): URL of the scraped web page.
- time (int): UNIX timestamp of the scraping.
- title (str): Title of the article.
- ingress (str): Ingress of the article.
- content (str): Contents of the article.
- published (str): Publish time of the article.
- author (str): Author of the article.
- images (list of dict): Images of the article.

clear()

Clear contents of jobdir.

get(*fmt='df'*)

Return scraped data as DataFrame or list.

Parameters
fmt (*str, optional*) – Format to return parsed items as. Should be in ['df', 'list']. Defaults to 'df'.

Returns
If fmt = 'df', DataFrame of scraped items. If fmt = 'list', list of dict of scraped items.

Raises
ValueError – If fmt not in allowed values.

property items_save_path

Save of path of the scraped items.

Cannot be changed after initialization of a spider.

property jobdir

Working directory of the spider.

Can be changed after initialization of a spider.

classmethod load(*jobdir*)

Load existing spider from jobdir.

Parameters
jobdir (*str*) – Path to job directory.

Returns
Spider loaded from job directory.

property log_level

Logging level of the spider.

This attribute can be changed after initialization of a spider.

property progress_bar

Whether progress bar is enabled or not.

Can be changed after initialization of a spider.

save()

Save spider in jobdir for later use.

Returns
Path to job directory.

Return type
str

scrape(*n=10*, *timeout=60*, *settings=None*)

Scrape given number of items.

Parameters

- **n** (*int, optional*) – Number of items to attempt to scrape. Zero corresponds to no limit. Defaults to 10.
- **timeout** (*int, optional*) – Timeout in seconds to wait before stopping the spider. Zero corresponds to no limit. Defaults to 60.
- **settings** (*dict or None, optional*) – Scrapy spider settings to use. Defaults to None, which correspond to default settings. See list of available settings at: https://docs.scrapy.org/en/latest/topics/settings.html.

Returns
self

property spider_save_path

Save path of the spider.

Cannot be changed after initialization of a spider.

class finscraper.spiders.**OikotieApartment**(*area=None*, *jobdir=None*, *progress_bar=True*, *log_level=None*)

Bases: _SpiderWrapper

Fetch oikotie.fi apartments.

Args:

area (str, optional): Scrape listings based on area, e.g.
"helsinki" or "hausjärvi". The final URL will be formed as: 'https://asunnot.oikotie.fi/myytavat-asunnot/{area}'. Defaults to None.

jobdir (str or None, optional): Working directory of the spider.
Defaults to None, which creates a temp directory to be used. Note that this directory will only be deleted through the `clear` method!

progress_bar (bool, optional): Whether to enable progress bar or not. This
parameter is ignored if `log_level` is not None. Defaults to True.

log_level (str or None, optional): Logging level to display. Should be in
['debug', 'info', 'warn', 'error', 'critical'] or None (disabled). Defaults to None.

Note: This parameter can be overridden through Scrapy `settings` (LOG_LEVEL, LOG_ENABLED) within the `scrape` -method.

Returned fields:

- url (str): URL of the scraped web page.
- time (int): UNIX timestamp of the scraping.
- title (str): Title of the web browser tab.
- overview (str): Overview text of the apartment.
- contact_person_name (str): Name of the contact person.
- contact_person_job_title (str): Job title of the contact person.
- contact_person_phone_number (str): Phone number of the contact person.
- contact_person_company (str): Company of the contact person.
- location (str): Sijainti
- city (str): Kaupunki
- district (str): Kaupunginosa
- oikotie_id (str): Kohdenumero
- floor (str): Kerros
- life_sq (str): Asuinpinta-ala
- property_sq (str): Tontin pinta-ala
- total_sq (str): Kokonaispinta-ala
- room_info (str): Huoneiston kokoonpano
- number_of_rooms (str): Huoneita
- condition (str): Kunto
- condition_details (str): Kunnon lisätiedot
- availability (str): Lisätietoa vapautumisesta
- kitchen_appliances (str): Keittiön varusteet
- bathroom_appliances (str): Kylpyhuoneen varusteet
- window_direction (str): Ikkunoiden suunta

- has_balcony (str): Parveke
- balcony_details (str): Parvekkeen lisätiedot
- storage_space (str): Säilytystilat
- view (str): Näkymät
- future_renovations (str): Tulevat remontit
- completed_renovations (str): Tehdyt remontit
- has_sauna (str): Asunnossa sauna
- sauna_details (str): Saunan lisätiedot
- housing_type (str): Asumistyyppi
- services (str): Palvelut
- additional_info (str): Lisätiedot
- property_id (str): Kiinteistötunnus
- apartment_is (str): Kohde on
- telecommunication_services (str): Tietoliikennepalvelut
- price_no_tax (str): Velaton hinta
- sales_price (str): Myyntihinta
- shared_loan_payment (str): Lainaosuuden maksu
- price_per_sq (str): Neliöhinta
- share_of_liabilities (str): Velkaosuus
- mortgages (str): Kiinnitykset
- financial_charge (str): Rahoitusvastike
- condominium_payment (str): Hoitovastike
- maintenance_charge (str): Yhtiövastike
- water_charge (str): Vesimaksu
- water_charge_details (str): Vesimaksun lisätiedot
- heating_charge (str): Lämmityskustannukset
- other_costs (str): Muut kustannukset
- is_brand_new (str): Uudiskohde
- housing_company_name (str): Taloyhtiön nimi
- building_type (str): Rakennuksen tyyppi
- build_year (str): Rakennusvuosi
- build_year_details (str): Rakennusvuoden lisätiedot
- number_of_apartments (str): Huoneistojen lukumäärä
- total_floors (str): Kerroksia
- building_has_elevator (str): Hissi
- building_has_sauna (str): Taloyhtiössä on sauna

- building_material (str): Rakennusmateriaali
- roof_type (str): Kattotyyppi
- energy_class (str): Energialuokka
- has_energy_certificate (str): Energiatoditus
- antenna_system (str): Kiinteistön antennijärjestelmä
- property_size (str): Tontin koko
- maintenance (str): Kiinteistönhoito
- real_estate_management (str): Isännöinti
- plan_info (str): Kaavatiedot
- plan (str): Kaavatilanne
- traffic_communication (str): Liikenneyhteydet
- heating (str): Lämmitys
- parking_space_description (str): Pysäköintitilan kuvaus
- common_spaces (str): Yhteiset tilat
- wallcovering (str): Pintamateriaalit

clear()

Clear contents of `jobdir`.

get(*fmt='df'*)

Return scraped data as DataFrame or list.

Parameters
fmt (*str, optional*) – Format to return parsed items as. Should be in ['df', 'list']. Defaults to 'df'.

Returns
If `fmt = 'df'`, DataFrame of scraped items. If `fmt = 'list'`, list of dict of scraped items.

Raises
ValueError – If `fmt` not in allowed values.

property items_save_path

Save of path of the scraped items.

Cannot be changed after initialization of a spider.

property jobdir

Working directory of the spider.

Can be changed after initialization of a spider.

classmethod load(*jobdir*)

Load existing spider from `jobdir`.

Parameters
jobdir (*str*) – Path to job directory.

Returns
Spider loaded from job directory.

property log_level

Logging level of the spider.

This attribute can be changed after initialization of a spider.

property progress_bar

Whether progress bar is enabled or not.

Can be changed after initialization of a spider.

save()

Save spider in jobdir for later use.

Returns
Path to job directory.

Return type
str

scrape(*n=10*, *timeout=60*, *settings=None*)

Scrape given number of items.

Parameters

- **n** (*int, optional*) – Number of items to attempt to scrape. Zero corresponds to no limit. Defaults to 10.
- **timeout** (*int, optional*) – Timeout in seconds to wait before stopping the spider. Zero corresponds to no limit. Defaults to 60.
- **settings** (*dict or None, optional*) – Scrapy spider settings to use. Defaults to None, which correspond to default settings. See list of available settings at: https://docs.scrapy.org/en/latest/topics/settings.html.

Returns
self

property spider_save_path

Save path of the spider.

Cannot be changed after initialization of a spider.

class finscraper.spiders.**Suomi24Page**(*jobdir=None*, *progress_bar=True*, *log_level=None*)

Bases: _SpiderWrapper

Fetch comments from suomi24.fi.

Parameters

- **jobdir** (*str or None, optional*) – Working directory of the spider. Defaults to None, which creates a temp directory to be used. Note that this directory will only be deleted through the clear method!
- **progress_bar** (*bool, optional*) – Whether to enable progress bar or not. This parameter is ignored if log_level is not None. Defaults to True.
- **log_level** (*str or None, optional*) – Logging level to display. Should be in ['debug', 'info', 'warn', 'error', 'critical'] or None (disabled). Defaults to None.

Note: This parameter can be overridden through Scrapy settings (LOG_LEVEL, LOG_ENABLED) within the scrape -method.

Returned page fields:

- url (str): URL of the scraped web page.
- time (int): UNIX timestamp of the scraping.
- title (str): Title of the thread.
- content (str): Content of the first message.
- comments (str): Comments of the thread page.
- published (str): Publish time of the thread.
- author (str): Author of the thread.
- n_comments (int): Number of comments in the thread.
- views (str): Number of views.

Returned comment fields:

- author (str): Author of the comment.
- date (str): Publish time of the comment.
- quotes (list of str): List of quotes in the comment.
- responses (list of dict): Response comments to this comment.
- content (str): Contents of the comment.

Returned comment response fields:

- author (str): Author of the comment response.
- date (str): Publish time of the comment response.
- quotes (list of str): List of quotes in the comment response.
- content (str): Contents of the comment response.

`clear()`

Clear contents of `jobdir`.

`get(`*fmt='df'*`)`

Return scraped data as DataFrame or list.

Parameters
fmt (*str, optional*) – Format to return parsed items as. Should be in [‘df’, ‘list’]. Defaults to ‘df’.

Returns
If `fmt = 'df'`, DataFrame of scraped items. If `fmt = 'list'`, list of dict of scraped items.

Raises
ValueError – If `fmt` not in allowed values.

`property items_save_path`

Save of path of the scraped items.

Cannot be changed after initialization of a spider.

property jobdir

Working directory of the spider.

Can be changed after initialization of a spider.

classmethod load(*jobdir*)

Load existing spider from `jobdir`.

Parameters
jobdir (*str*) – Path to job directory.

Returns
Spider loaded from job directory.

property log_level

Logging level of the spider.

This attribute can be changed after initialization of a spider.

property progress_bar

Whether progress bar is enabled or not.

Can be changed after initialization of a spider.

save()

Save spider in `jobdir` for later use.

Returns
Path to job directory.

Return type
str

scrape(*n=10*, *timeout=60*, *settings=None*)

Scrape given number of items.

Parameters

- **n** (*int, optional*) – Number of items to attempt to scrape. Zero corresponds to no limit. Defaults to 10.
- **timeout** (*int, optional*) – Timeout in seconds to wait before stopping the spider. Zero corresponds to no limit. Defaults to 60.
- **settings** (*dict or None, optional*) – Scrapy spider settings to use. Defaults to None, which correspond to default settings. See list of available settings at: https://docs.scrapy.org/en/latest/topics/settings.html.

Returns
self

property spider_save_path

Save path of the spider.

Cannot be changed after initialization of a spider.

class finscraper.spiders.**ToriDeal**(*jobdir=None*, *progress_bar=True*, *log_level=None*)

Bases: _SpiderWrapper

Fetch deals from tori.fi.

Parameters

- **jobdir** (*str or None, optional*) – Working directory of the spider. Defaults to None, which creates a temp directory to be used. Note that this directory will only be deleted through the `clear` method!
- **progress_bar** (*bool, optional*) – Whether to enable progress bar or not. This parameter is ignored if `log_level` is not None. Defaults to True.
- **log_level** (*str or None, optional*) – Logging level to display. Should be in ['debug', 'info', 'warn', 'error', 'critical'] or None (disabled). Defaults to None.

Note: This parameter can be overridden through Scrapy `settings` (LOG_LEVEL, LOG_ENABLED) within the `scrape` -method.

Returned fields:

- url (str): URL of the scraped web page.
- time (int): UNIX timestamp of the scraping.
- seller (str): Seller of the item.
- name (str): Name of the item.
- description (list of str): Description of the item.
- price (str): Price of the item.
- type (str): Type of the deal.
- published (str): Publish time of the deal.
- images (list of dict): Images of the item.

`clear()`

Clear contents of `jobdir`.

`get(`*fmt='df'*`)`

Return scraped data as DataFrame or list.

Parameters
fmt (*str, optional*) – Format to return parsed items as. Should be in ['df', 'list']. Defaults to 'df'.

Returns
If `fmt = 'df'`, DataFrame of scraped items. If `fmt = 'list'`, list of dict of scraped items.

Raises
ValueError – If `fmt` not in allowed values.

`property items_save_path`

Save of path of the scraped items.

Cannot be changed after initialization of a spider.

`property jobdir`

Working directory of the spider.

Can be changed after initialization of a spider.

classmethod load(*jobdir*)

Load existing spider from `jobdir`.

Parameters
jobdir (*str*) – Path to job directory.

Returns
Spider loaded from job directory.

property log_level

Logging level of the spider.

This attribute can be changed after initialization of a spider.

property progress_bar

Whether progress bar is enabled or not.

Can be changed after initialization of a spider.

save()

Save spider in `jobdir` for later use.

Returns
Path to job directory.

Return type
str

scrape(*n=10*, *timeout=60*, *settings=None*)

Scrape given number of items.

Parameters

- **n** (*int, optional*) – Number of items to attempt to scrape. Zero corresponds to no limit. Defaults to 10.
- **timeout** (*int, optional*) – Timeout in seconds to wait before stopping the spider. Zero corresponds to no limit. Defaults to 60.
- **settings** (*dict or None, optional*) – Scrapy spider settings to use. Defaults to None, which correspond to default settings. See list of available settings at: https://docs.scrapy.org/en/latest/topics/settings.html.

Returns
self

property spider_save_path

Save path of the spider.

Cannot be changed after initialization of a spider.

class finscraper.spiders.**VauvaPage**(*jobdir=None*, *progress_bar=True*, *log_level=None*)

Bases: `_SpiderWrapper`

Fetch comments from vauva.fi.

Parameters

- **jobdir** (*str or None, optional*) – Working directory of the spider. Defaults to None, which creates a temp directory to be used. Note that this directory will only be deleted through the `clear` method!

- **progress_bar** (*bool, optional*) – Whether to enable progress bar or not. This parameter is ignored if `log_level` is not None. Defaults to True.
- **log_level** (*str or None, optional*) – Logging level to display. Should be in ['debug', 'info', 'warn', 'error', 'critical'] or None (disabled). Defaults to None.

Note: This parameter can be overridden through Scrapy `settings` (LOG_LEVEL, LOG_ENABLED) within the `scrape` -method.

Returned page fields:

- url (str): URL of the scraped web page.
- time (int): UNIX timestamp of the scraping.
- title (str): Title of the thread.
- page (int): Page number of the thread.
- pages (int): Pages in the thread.
- comments (str): Comments of the thread page.
- published (str): Publish time of the article.
- author (str): Author of the article.

Returned comment fields:

- author (str): Author of the comment.
- date (str): Publish time of the comment.
- quotes (list of str): List of quotes in the comment.
- content (str): Contents of the comment.
- upvotes (int): Upvotes of the comment.
- downvotes (int): Downvotes of the comment.

`clear()`

Clear contents of `jobdir`.

`get(`*fmt='df'*`)`

Return scraped data as DataFrame or list.

Parameters
fmt (*str, optional*) – Format to return parsed items as. Should be in ['df', 'list']. Defaults to 'df'.

Returns
If `fmt = 'df'`, DataFrame of scraped items. If `fmt = 'list'`, list of dict of scraped items.

Raises
ValueError – If `fmt` not in allowed values.

`property items_save_path`

Save of path of the scraped items.

Cannot be changed after initialization of a spider.

property jobdir

Working directory of the spider.

Can be changed after initialization of a spider.

classmethod load(*jobdir*)

Load existing spider from `jobdir`.

Parameters
jobdir (*str*) – Path to job directory.

Returns
Spider loaded from job directory.

property log_level

Logging level of the spider.

This attribute can be changed after initialization of a spider.

property progress_bar

Whether progress bar is enabled or not.

Can be changed after initialization of a spider.

save()

Save spider in `jobdir` for later use.

Returns
Path to job directory.

Return type
str

scrape(*n=10*, *timeout=60*, *settings=None*)

Scrape given number of items.

Parameters

- **n** (*int, optional*) – Number of items to attempt to scrape. Zero corresponds to no limit. Defaults to 10.
- **timeout** (*int, optional*) – Timeout in seconds to wait before stopping the spider. Zero corresponds to no limit. Defaults to 60.
- **settings** (*dict or None, optional*) – Scrapy spider settings to use. Defaults to None, which correspond to default settings. See list of available settings at: https://docs.scrapy.org/en/latest/topics/settings.html.

Returns
self

property spider_save_path

Save path of the spider.

Cannot be changed after initialization of a spider.

class finscraper.spiders.**YLEArticle**(*jobdir=None*, *progress_bar=True*, *log_level=None*)

Bases: _SpiderWrapper

Fetch YLE news articles.

Parameters

- **jobdir** (*str or None, optional*) – Working directory of the spider. Defaults to None, which creates a temp directory to be used. Note that this directory will only be deleted through the `clear` method!
- **progress_bar** (*bool, optional*) – Whether to enable progress bar or not. This parameter is ignored if `log_level` is not None. Defaults to True.
- **log_level** (*str or None, optional*) – Logging level to display. Should be in ['debug', 'info', 'warn', 'error', 'critical'] or None (disabled). Defaults to None.

Note: This parameter can be overridden through Scrapy `settings` (LOG_LEVEL, LOG_ENABLED) within the `scrape` -method.

Returned fields:

- url (str): URL of the scraped web page.
- time (int): UNIX timestamp of the scraping.
- title (str): Title of the article.
- ingress (str): Ingress of the article.
- content (str): Contents of the article.
- published (str): Publish time of the article.
- author (str): Author of the article.
- images (list of dict): Images of the article.

`clear()`

Clear contents of `jobdir`.

`get(`*fmt='df'*`)`

Return scraped data as DataFrame or list.

Parameters
fmt (*str, optional*) – Format to return parsed items as. Should be in ['df', 'list']. Defaults to 'df'.

Returns
If `fmt = 'df'`, DataFrame of scraped items. If `fmt = 'list'`, list of dict of scraped items.

Raises
ValueError – If `fmt` not in allowed values.

`property items_save_path`

Save of path of the scraped items.

Cannot be changed after initialization of a spider.

`property jobdir`

Working directory of the spider.

Can be changed after initialization of a spider.

`classmethod load(`*jobdir*`)`

Load existing spider from `jobdir`.

Parameters
jobdir (*str*) – Path to job directory.

Returns
Spider loaded from job directory.

property log_level

Logging level of the spider.

This attribute can be changed after initialization of a spider.

property progress_bar

Whether progress bar is enabled or not.

Can be changed after initialization of a spider.

save()

Save spider in `jobdir` for later use.

Returns
Path to job directory.

Return type
str

scrape(*n=10*, *timeout=60*, *settings=None*)

Scrape given number of items.

Parameters

- **n** (*int, optional*) – Number of items to attempt to scrape. Zero corresponds to no limit. Defaults to 10.
- **timeout** (*int, optional*) – Timeout in seconds to wait before stopping the spider. Zero corresponds to no limit. Defaults to 60.
- **settings** (*dict or None, optional*) – Scrapy spider settings to use. Defaults to None, which correspond to default settings. See list of available settings at: https://docs.scrapy.org/en/latest/topics/settings.html.

Returns
self

property spider_save_path

Save path of the spider.

Cannot be changed after initialization of a spider.

3.6.9 finscraper.text_utils module

Module for text processing utility functions and classes.

finscraper.text_utils.**drop_empty_elements**(*text_list*)

finscraper.text_utils.**paragraph_join**(*text_list*)

finscraper.text_utils.**replace**(*text*, *source*, *target*)

finscraper.text_utils.**safe_cast_int**(*text*)

finscraper.text_utils.**strip_elements**(*text_list*)

finscraper.text_utils.**strip_join**(*text_list*, *join_with=' '*)

3.6.10 finscraper.utils module

Module for utility functions and classes.

class finscraper.utils.**QueueHandler**(*queue*)

Bases: Handler

Sends events to a queue, allowing multiprocessing.

This handler checks for picklability before saving items into queue. Modified from: https://gist.github.com/vsajip/591589

acquire()

Acquire the I/O thread lock.

addFilter(*filter*)

Add the specified filter to this handler.

close()

Tidy up any resources used by the handler.

This version removes the handler from an internal map of handlers, _handlers, which is used for handler lookup by name. Subclasses should ensure that this gets called from overridden close() methods.

createLock()

Acquire a thread lock for serializing access to the underlying I/O.

emit(*record*)

Do whatever it takes to actually log the specified logging record.

This version is intended to be implemented by subclasses and so raises a NotImplementedError.

enqueue(*record*)

filter(*record*)

Determine if a record is loggable by consulting all the filters.

The default is to allow the record to be logged; any filter can veto this and the record is then dropped. Returns a zero value if a record is to be dropped, else non-zero.

Changed in version 3.2: Allow filters to be just callables.

flush()

Ensure all logging output has been flushed.

This version does nothing and is intended to be implemented by subclasses.

format(*record*)

Format the specified record.

If a formatter is set, use it. Otherwise, use the default formatter for the module.

get_name()

handle(*record*)

Conditionally emit the specified logging record.

Emission depends on filters which may have been added to the handler. Wrap the actual emission of the record with acquisition/release of the I/O thread lock. Returns whether the filter passed the record for emission.

handleError(*record*)

Handle errors which occur during an emit() call.

This method should be called from handlers when an exception is encountered during an emit() call. If raiseExceptions is false, exceptions get silently ignored. This is what is mostly wanted for a logging system - most users will not care about errors in the logging system, they are more interested in application errors. You could, however, replace this with a custom handler if you wish. The record which was being processed is passed in to this method.

property name

prepare(*record*)

release()

Release the I/O thread lock.

removeFilter(*filter*)

Remove the specified filter from this handler.

setFormatter(*fmt*)

Set the formatter for this handler.

setLevel(*level*)

Set the logging level of this handler. level must be an int or a str.

set_name(*name*)

class finscraper.utils.**TqdmLogger**(*logger*)

Bases: StringIO

File-like object that redirects buffer to stdout.

close()

Close the IO object.

Attempting any further operation after the object is closed will raise a ValueError.

This method has no effect if the file is already closed.

closed

detach()

Separate the underlying buffer from the TextIOBase and return it.

After the underlying buffer has been detached, the TextIO is in an unusable state.

encoding

Encoding of the text stream.

Subclasses should override.

errors

The error setting of the decoder or encoder.

Subclasses should override.

fileno()

Returns underlying file descriptor if one exists.

OSError is raised if the IO object does not use a file descriptor.

flush()

Flush write buffers, if applicable.

This is not implemented for read-only and non-blocking streams.

getvalue()

Retrieve the entire contents of the object.

isatty()

Return whether this is an 'interactive' stream.

Return False if it can't be determined.

line_buffering

newlines

read(*size=- 1*, */*)

Read at most size characters, returned as a string.

If the argument is negative or omitted, read until EOF is reached. Return an empty string at EOF.

readable()

Returns True if the IO object can be read.

readline(*size=- 1*, */*)

Read until newline or EOF.

Returns an empty string if EOF is hit immediately.

readlines(*hint=- 1*, */*)

Return a list of lines from the stream.

hint can be specified to control the number of lines read: no more lines will be read if the total size (in bytes/characters) of all lines so far exceeds hint.

seek(*pos*, *whence=0*, */*)

Change stream position.

Seek to character offset pos relative to position indicated by whence:
0 Start of stream (the default). pos should be >= 0; 1 Current position - pos must be 0; 2 End of stream - pos must be 0.

Returns the new absolute position.

seekable()

Returns True if the IO object can be seeked.

tell()

Tell the current file position.

truncate(*pos=None*, */*)

Truncate size to pos.

The pos argument defaults to the current file position, as returned by tell(). The current file position is unchanged. Returns the new absolute position.

writable()

Returns True if the IO object can be written.

write(*buf*)

Write string to file.

Returns the number of characters written, which is always equal to the length of the string.

writelines(*lines*, /)

Write a list of lines to stream.

Line separators are not added, so it is usual for each of the lines provided to have a line separator at the end.

finscraper.utils.**get_chromedriver**(*options=None*, *settings=None*)

Get chromedriver automatically.

Parameters

- **options** (*selenium.webdriver.chrome.options.Options, optional*) – Options to start chromedriver with. If None, will use default settings. Defaults to None.
- **settings** (*scrapy.settings.Settings, optional*) – Scrapy settings to take into consideration when starting chromedriver. If None, will not be taken into consideration. Defaults to None.

Returns

Selenium webdriver for Chrome (selenium.webdriver.Chrome).

3.6.11 finscraper.wrappers module

Module for wrapping Scrapy spiders.

3.6.12 Module contents

PYTHON MODULE INDEX

f

INDEX

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

Y