
Search Engine Parser Documentation

Release 0.6.1

Diretnan Domnan, Mmadu Manasseh

Jul 19, 2020

Contents:

1	Search Engine Parser	1
1.1	Installation	2
1.2	Development	2
1.3	Code Documentation	2
1.4	Running the tests	2
1.5	Usage	3
1.6	Code of Conduct	5
1.7	Contribution	5
1.8	License (MIT)	5
1.9	Contributors	5
2	Indices and tables	7

CHAPTER 1

Search Engine Parser

`pypi package` **0.6.1**

Package to query popular search engines and scrape for result titles, links and descriptions. Aims to scrape the widest range of search engines. View all [supported engines](#)

- *Search Engine Parser*
 - *Popular Supported Engines*
 - *Installation*
 - *Development*
 - *Code Documentation*
 - *Running the tests*
 - *Usage*
 - * *Code*
 - * *Command line*
 - *Code of Conduct*
 - *Contribution*
 - *License (MIT)* ## Popular Supported Engines

Some of the popular search engines include:

- Google
- DuckDuckGo
- GitHub
- StackOverflow
- Baidu
- YouTube

View all [supported engines](#)

1.1 Installation

```
# install only package dependencies
pip install search-engine-parser
# Installs `pysearch` cli tool
pip install "search-engine-parser[cli]"
```

Install current version on master branch

```
pip install git+https://github.com/bisoncorps/search-engine-parser
```

1.2 Development

Clone the repository

```
git clone git@github.com:bisoncorps/search-engine-parser.git
```

Create virtual environment and install requirements

```
mkvirtualenv search_engine_parser
pip install -r requirements/dev.txt
```

1.3 Code Documentation

Found on [Read the Docs](#)

1.4 Running the tests

```
pytest
```

1.5 Usage

1.5.1 Code

Query Results can be scraped from popular search engines as shown in the example snippet below

```
import pprint

from search_engine_parser.core.engines.bing import Search as BingSearch
from search_engine_parser.core.engines.google import Search as GoogleSearch
from search_engine_parser.core.engines.yahoo import Search as YahooSearch

search_args = ('preaching to the choir', 1)
gsearch = GoogleSearch()
ysearch = YahooSearch()
bsearch = BingSearch()
gresults = gsearch.search(*search_args)
yresults = ysearch.search(*search_args)
bresults = bsearch.search(*search_args)
a = {
    "Google": gresults,
    "Yahoo": yresults,
    "Bing": bresults
}

# pretty print the result from each engine
for k, v in a.items():
    print(f"-----{k}-----")
    for result in v:
        pprint.pprint(result)

# print first title from google search
print(gresults["titles"][0])
# print 10th link from yahoo search
print(yresults["links"][9])
# print 6th description from bing search
print(bresults["descriptions"][5])

# print first result containing links, descriptions and title
print(gresults[0])
```

For localization, you can pass the url keyword and a localized url. This would use the url to query and parse using the same engine's parser

```
# Use google.de instead of google.com
results = gsearch.search(*search_args, url="google.de")
```

Cache

The results are automatically cached for engine searches, you can either bypass cache by adding `cache=False` to the search or `async_search` method or clear the engines cache

```
from search_engine_parser.core.engines.github import Search as GitHub
github = GitHub()
# bypass the cache
```

(continues on next page)

(continued from previous page)

```
github.search("search-engine-parser", cache=False)

#OR
# clear cache before search
github.clear_cache()
github.search("search-engine-parser")
```

Async

search-engine-parser supports async hence you could use codes like

```
results = await gsearch.async_search(*search_args)
```

Results

The SearchResults after the searching

```
>>> results = gsearch.search("preaching the choir", 1)
>>> results
<search_engine_parser.core.base.SearchResult object at 0x7f907426a280>
# The object supports retrieving individual results by iteration of just by type
↳ (links, descriptions, titles)
>>> results[0] # Returns the first <SearchItem>
>>> results[0]["description"] # Get the description of the first item
>>> results[0]["link"] # get the link of the first item
>>> results["descriptions"] # Returns a list of all descriptions from all results
```

It can be iterated like a normal list to return individual SearchItem

1.5.2 Command line

Search engine parser comes with a CLI tool known as pysearch e.g

```
pysearch --engine bing search --query "Preaching to the choir" --type descriptions
```

Result

```
'Preaching to the choir' originated in the USA in the 1970s. It is a variant of the
↳ earlier 'preaching to the converted', which dates from England in the late 1800s
↳ and has the same meaning. Origin - the full story 'Preaching to the choir' (also
↳ sometimes spelled quire) is of US origin.
```

There is a needed argument for the CLI i.e -e Engine followed by either of two subcommands in the CLI i.e search and summary

```
usage: pysearch [-h] [-u URL] [-e ENGINE] {search,summary} ...

SearchEngineParser

positional arguments:
```

(continues on next page)

(continued from previous page)

```

{search,summary}    help for subcommands
  search            search help
  summary           summary help

optional arguments:
  -h, --help            show this help message and exit
  -u URL, --url URL     A custom link to use as base url for search e.g
                        google.de
  -e ENGINE, --engine ENGINE
                        Engine to use for parsing the query e.g google, yahoo,
                        bing,duckduckgo (default: google)

```

summary just shows the summary of each search engine added with descriptions on the return

```
pysearch --engine google summary
```

Full arguments for the search subcommand shown below

```

usage: pysearch search [-h] -q QUERY [-p PAGE] [-t TYPE] [-r RANK]

optional arguments:
  -h, --help            show this help message and exit
  -q QUERY, --query QUERY
                        Query string to search engine for
  -p PAGE, --page PAGE  Page of the result to return details for (default: 1)
  -t TYPE, --type TYPE  Type of detail to return i.e full, links, descriptions
                        or titles (default: full)
  -r RANK, --rank RANK  ID of Detail to return e.g 5 (default: 0)
  -cc, --clear_cache    Clear cache of engine before searching

```

1.6 Code of Conduct

All actions performed should adhere to the [code of conduct](#)

1.7 Contribution

Before making any contribution, please follow the [contribution guide](#)

1.8 License (MIT)

This project is opened under the [MIT 2.0 License](#) which allows very broad use for both academic and commercial purposes.

1.9 Contributors

Thanks goes to these wonderful people ([emoji key](#)):

This project follows the [all-contributors](#) specification. Contributions of any kind welcome!

CHAPTER 2

Indices and tables

- `genindex`
- `modindex`
- `search`