darc

Release 0.5.0rc2

Jarry Shaw

Nov 21, 2020

CONTENTS

1 Darkweb Crawler Project **1**
- 1.1 URL Utilities . . . 1
- 1.2 Source Parsing . . . 4
- 1.3 Link Database . . . 7
- 1.4 Proxy Utilities . . . 10
- 1.5 Sites Customisation . . . 18
- 1.6 Module Constants . . . 19
- 1.7 Custom Exceptions . . . 25

2 Web Backend Demo **27**

3 Docker Integration **33**

4 Auxiliary Scripts **43**
- 4.1 Health Check . . . 43
- 4.2 Upload API Submission Files . . . 43
- 4.3 Remove Repeated Lines . . . 44

5 Installation **47**

6 Usage **49**

7 Configuration **51**
- 7.1 General Configurations . . . 51
- 7.2 Data Storage . . . 53
- 7.3 Web Crawlers . . . 53
- 7.4 White / Black Lists . . . 54
- 7.5 Data Submission . . . 56
- 7.6 Tor Proxy Configuration . . . 56
- 7.7 I2P Proxy Configuration . . . 57
- 7.8 ZeroNet Proxy Configuration . . . 58
- 7.9 Freenet Proxy Configuration . . . 59

8 Indices and tables **61**

Python Module Index **63**

Index **65**

CHAPTER

ONE

DARKWEB CRAWLER PROJECT

darc is designed as a swiss army knife for darkweb crawling. It integrates `requests` to collect HTTP request and response information, such as cookies, header fields, etc. It also bundles `selenium` to provide a fully rendered web page and screenshot of such view.

1.1 URL Utilities

The *Link* class is the key data structure of the *darc* project, it contains all information required to identify a URL's proxy type, hostname, path prefix when saving, etc.

The *link* module also provides several wrapper function to the `urllib.parse`.

class darc.link.**Link**(*url*, *proxy*, *url_parse*, *host*, *base*, *name*)

Bases: `object`

Parsed link.

Parameters

- **url** – original link
- **proxy** – proxy type
- **host** – URL's hostname
- **base** – base folder for saving files
- **name** – hashed link for saving files
- **url_parse** – parsed URL from `urllib.parse.urlparse()`

Returns Parsed link object.

Return type *Link*

Note: *Link* is a dataclass object. It is safely *hashable*, through `hash(url)`.

__hash__()

Provide hash support to the *Link* object.

base: **str**

base folder for saving files

host: **str**

URL's hostname

name: *str*
hashed link for saving files

proxy: *str*
proxy type

url: *str*
original link

url_parse: *urllib.parse.ParseResult*
parsed URL from `urllib.parse.urlparse()`

`darc.link.`**parse_link**(*link*, *host=None*)
Parse link.

Parameters

- **link** (*str*) – link to be parsed
- **host** (*Optional[str]*) – hostname of the link

Returns The parsed link object.

Return type *darc.link.Link*

Note: If `host` is provided, it will override the hostname of the original `link`.

The parsing process of proxy type is as follows:

0. If `host` is `None` and the parse result from `urllib.parse.urlparse()` has no `netloc` (or hostname) specified, then set `hostname` as `(null)`; else set it as is.
1. If the scheme is `data`, then the `link` is a data URI, set `hostname` as `data` and `proxy` as `data`.
2. If the scheme is `javascript`, then the link is some JavaScript codes, set `proxy` as `script`.
3. If the scheme is `bitcoin`, then the link is a Bitcoin address, set `proxy` as `bitcoin`.
4. If the scheme is `ed2k`, then the link is an ED2K magnet link, set `proxy` as `ed2k`.
5. If the scheme is `magnet`, then the link is a magnet link, set `proxy` as `magnet`.
6. If the scheme is `mailto`, then the link is an email address, set `proxy` as `mail`.
7. If the scheme is `irc`, then the link is an IRC link, set `proxy` as `irc`.
8. If the scheme is **NOT** any of `http` or `https`, then set `proxy` to the scheme.
9. If the host is `None`, set `hostname` to `(null)`, set `proxy` to `null`.
10. If the host is an onion (`.onion`) address, set `proxy` to `tor`.
11. If the host is an I2P (`.i2p`) address, or any of `localhost:7657` and `localhost:7658`, set `proxy` to `i2p`.
12. If the host is *localhost* on *ZERONET_PORT*, and the path is not `/`, i.e. **NOT** root path, set `proxy` to `zeronet`; and set the first part of its path as `hostname`.

    Example:

    For a ZeroNet address, e.g. http://127.0.0.1:43110/1HeLLo4uzjaLetFx6NH3PMwFP3qbRbTf3D, *parse_link()* will parse the `hostname` as `1HeLLo4uzjaLetFx6NH3PMwFP3qbRbTf3D`.

13. If the host is *localhost* on *FREENET_PORT*, and the path is not `/`, i.e. **NOT** root path, set `proxy` to `freenet`; and set the first part of its path as `hostname`.

    Example:

    For a Freenet address, e.g. http://127.0.0.1:8888/USK@nwa8lHa271k2QvJ8aa0Ov7IHAV-DFOCFgmDt3X6BpCI,DuQSUZiI~agF8c-6tjsFFGuZ8eICrzWCILB60nT8KKo,AQACAAE/sone/77/, *parse_link()* will parse the `hostname` as `USK@nwa8lHa271k2QvJ8aa0Ov7IHAV-DFOCFgmDt3X6BpCI,DuQSUZiI~agF8c-6tjsFFGuZ8eICrzWCILB60nT8KKo,AQACAAE`.

14. If none of the cases above satisfied, the `proxy` will be set as `null`, marking it a plain normal link.

The `base` for parsed link *Link* object is defined as

```
<root>/<proxy>/<scheme>/<hostname>/
```

where `root` is *PATH_DB*.

The `name` for parsed link *Link* object is the sha256 hash (c.f. `hashlib.sha256()`) of the original `link`.

darc.link.**quote**(*string*, *safe='/'*, *encoding=None*, *errors=None*)
Wrapper function for `urllib.parse.quote()`.

Parameters

- **string** (*AnyStr*) – string to be quoted
- **safe** (*AnyStr*) – charaters not to escape
- **encoding** (*Optional[str]*) – string encoding
- **errors** (*Optional[str]*) – encoding error handler

Returns The quoted string.

Return type str

Note: The function suppressed possible errors when calling `urllib.parse.quote()`. If any, it will return the original string.

---

darc.link.**unquote**(*string*, *encoding='utf-8'*, *errors='replace'*)
Wrapper function for `urllib.parse.unquote()`.

Parameters

- **string** (*AnyStr*) – string to be unquoted
- **encoding** (*str*) – string encoding
- **errors** (*str*) – encoding error handler

Returns The quoted string.

Return type str

Note: The function suppressed possible errors when calling `urllib.parse.unquote()`. If any, it will return the original string.

---

darc.link.**urljoin**(*base*, *url*, *allow_fragments=True*)
Wrapper function for `urllib.parse.urljoin()`.

Parameters

- **base** (*AnyStr*) – base URL
- **url** (*AnyStr*) – URL to be joined
- **allow_fragments** (*bool*) – if allow fragments

Returns The joined URL.

Return type str

Note: The function suppressed possible errors when calling `urllib.parse.urljoin()`. If any, it will return `base/url` directly.

`darc.link.`**`urlparse`**(*url*, *scheme=''*, *allow_fragments=True*)

Wrapper function for `urllib.parse.urlparse()`.

Parameters

- **url** (*str*) – URL to be parsed
- **scheme** (*str*) – URL scheme
- **allow_fragments** (*bool*) – if allow fragments

Returns The parse result.

Return type urllib.parse.ParseResult

Note: The function suppressed possible errors when calling `urllib.parse.urlparse()`. If any, it will return `urllib.parse.ParseResult(scheme=scheme, netloc='', path=url, params='', query='', fragment='')` directly.

1.2 Source Parsing

The *darc.parse* module provides auxiliary functions to read `robots.txt`, sitemaps and HTML documents. It also contains utility functions to check if the proxy type, hostname and content type if in any of the black and white lists.

`darc.parse.`**`_check`**(*temp_list*)

Check hostname and proxy type of links.

Parameters **temp_list** (*List[darc.link.Link]*) – List of links to be checked.

Returns List of links matches the requirements.

Return type List[*darc.link.Link*]

Note: If *CHECK_NG* is `True`, the function will directly call *_check_ng()* instead.

See also:

- *darc.parse.match_host()*
- *darc.parse.match_proxy()*

darc.parse.**_check_ng**(*temp_list*)
Check content type of links through HEAD requests.

Parameters **temp_list** (*List[darc.link.Link]*) – List of links to be checked.

Returns List of links matches the requirements.

Return type List[*darc.link.Link*]

See also:

- *darc.parse.match_host()*
- *darc.parse.match_proxy()*
- *darc.parse.match_mime()*

darc.parse.**check_robots**(*link*)
Check if link is allowed in robots.txt.

Parameters **link** (darc.link.Link) – The link object to be checked.

Returns If link is allowed in robots.txt.

Return type bool

Note: The root path of a URL will always return True.

darc.parse.**extract_links**(*link*, *html*, *check=False*)
Extract links from HTML document.

Parameters

- **link** (darc.link.Link) – Original link of the HTML document.
- **html** (*Union[str, bytes]*) – Content of the HTML document.
- **check** (*bool*) – If perform checks on extracted links, default to *CHECK*.

Returns List of extracted links.

Return type List[*darc.link.Link*]

See also:

- *darc.parse._check()*
- *darc.parse._check_ng()*

darc.parse.**get_content_type**(*response*)
Get content type from response.

Parameters **response** (requests.Response.) – Response object.

Returns The content type from response.

Return type str

Note: If the Content-Type header is not defined in response, the function will utilise magic to detect its content type.

darc.parse.**match_host**(*host*)
Check if hostname in black list.

Parameters **host** (*str*) – Hostname to be checked.

Returns If `host` in black list.

Return type bool

Note: If `host` is `None`, then it will always return `True`.

See also:

- *darc.const.LINK_WHITE_LIST*
- *darc.const.LINK_BLACK_LIST*
- *darc.const.LINK_FALLBACK*

darc.parse.**match_mime**(*mime*)
Check if content type in black list.

Parameters **mime** (*str*) – Content type to be checked.

Returns If `mime` in black list.

Return type bool

See also:

- *darc.const.MIME_WHITE_LIST*
- *darc.const.MIME_BLACK_LIST*
- *darc.const.MIME_FALLBACK*

darc.parse.**match_proxy**(*proxy*)
Check if proxy type in black list.

Parameters **proxy** (*str*) – Proxy type to be checked.

Returns If `proxy` in black list.

Return type bool

Note: If `proxy` is `script`, then it will always return `True`.

See also:

- *darc.const.PROXY_WHITE_LIST*
- *darc.const.PROXY_BLACK_LIST*
- *darc.const.PROXY_FALLBACK*

darc.save.**_SAVE_LOCK: multiprocessing.Lock**
I/O lock for saving link hash database `link.csv`.

See also:

- darc.save.save_link()

1.3 Link Database

The *darc* project utilises file system based database to provide tele-process communication.

Note: In its first implementation, the *darc* project used multiprocessing.Queue to support such communication. However, as noticed when runtime, the multiprocessing.Queue object will be much affected by the lack of memory.

There will be two databases, both locate at root of the data storage path *PATH_DB*:

- the requests database – queue_requests.txt
- the selenium database – queue_selenium.txt

At runtime, after reading such database, *darc* will keep a backup of the database with .tmp suffix to its file extension.

darc.db.**drop_hostname**(*link*)
Remove link from the hostname database.

Parameters **link** (darc.link.Link) – Link to be removed.

darc.db.**drop_requests**(*link*)
Remove link from the requests database.

Parameters **link** (darc.link.Link) – Link to be removed.

darc.db.**drop_selenium**(*link*)
Remove link from the selenium database.

Parameters **link** (darc.link.Link) – Link to be removed.

darc.db.**get_lock**(*name*, *timeout=None*, *sleep=0.1*, *blocking_timeout=None*, *lock_class=None*, *thread_local=True*)
Get a lock for Redis operations.

Parameters

- **name** (*str*) – Lock name.
- **timeout** (*Optional[float]*) – Maximum life for the lock.
- **sleep** (*float*) – Amount of time to sleep per loop iteration when the lock is in blocking mode and another client is currently holding the lock.
- **blocking_timeout** (*Optional[float]*) – Maximum amount of time in seconds to spend trying to acquire the lock.
- **lock_class** (*Optional[redis.lock.Lock]*) – Lock implementation.
- **thread_local** (*bool*) – Whether the lock token is placed in thread-local storage.

Returns Return a new redis.lock.Lock object using key name that mimics the behavior of threading.Lock.

Return type Union[redis.lock.Lock, contextlib.nullcontext]

Notes

If REDIS_LOCK is False, returns a contextlib.nullcontext instead.

darc.db.**have_hostname**(*link*)
Check if current link is a new host.

Parameters **link** (darc.link.Link) – Link to check against.

Returns If such link is a new host.

Return type bool

darc.db.**load_requests**(*check=False*)
Load link from the requests database.

Parameters **check** (*bool*) – If perform checks on loaded links, default to *CHECK*.

Returns List of loaded links from the requests database.

Return type List[*darc.link.Link*]

Note: At runtime, the function will load links with maximum number at *MAX_POOL* to limit the memory usage.

darc.db.**load_selenium**(*check=False*)
Load link from the selenium database.

Parameters **check** (*bool*) – If perform checks on loaded links, default to *CHECK*.

Returns List of loaded links from the selenium database.

Return type List[*darc.link.Link*]

Note: At runtime, the function will load links with maximum number at *MAX_POOL* to limit the memory usage.

darc.db.**save_requests**(*entries*, *single=False*, *score=None*, *nx=False*, *xx=False*)
Save link to the requests database.

Parameters

- **entries** (*List[darc.link.Link]*) – Links to be added to the requests database. It can be either a list of links, or a single link string (if single set as True).
- **single** (*bool*) – Indicate if entries is a list of links or a single link string.
- **score** – Score to for the Redis sorted set.
- **nx** – Forces ZADD to only create new elements and not to update scores for elements that already exist.
- **xx** – Forces ZADD to only update scores of elements that already exist. New elements will not be added.

darc.db.**save_selenium**(*entries*, *single=False*, *score=None*, *nx=False*, *xx=False*)
Save link to the selenium database.

Parameters

- **entries** (*List[darc.link.Link]*) – Links to be added to the `selenium` database. It can be either an *iterable* of links, or a single link string (if `single` set as `True`).
- **single** (*bool*) – Indicate if `entries` is an *iterable* of links or a single link string.
- **score** – Score to for the Redis sorted set.
- **nx** – Forces `ZADD` to only create new elements and not to update scores for elements that already exist.
- **xx** – Forces `ZADD` to only update scores of elements that already exist. New elements will not be added.

darc.db.**QR_LOCK: multiprocessing.Lock**
I/O lock for the `requests` database `_queue_requests.txt`.

See also:

- *darc.db.save_requests()*

darc.db.**QS_LOCK: Union[multiprocessing.Lock, threading.Lock, contextlib.nullcontext]**
I/O lock for the `selenium` database `_queue_selenium.txt`.

If *FLAG_MP* is `True`, it will be an instance of `multiprocessing.Lock`. If *FLAG_TH* is `True`, it will be an instance of `threading.Lock`. If none above, it will be an instance of `contextlib.nullcontext`.

See also:

- *darc.db.save_selenium()*
- *darc.const.FLAG_MP*
- *darc.const.FLAG_TH*

darc.db.**MAX_POOL: int**

Default `1_000`

Environ *DARC_MAX_POOL*

Maximum number of links loading from the database.

Note: If is an infinit `inf`, no limit will be applied.

---

darc.submit.**PATH_API = '{PATH_DB}/api/'**
Path to the API submittsion records, i.e. `api` folder under the root of data storage.

See also:

- *darc.const.PATH_DB*

darc.submit.**API_RETRY: int**
Retry times for API submission when failure.

Default `3`

Environ *API_RETRY*

darc.submit.**API_NEW_HOST: str**
API URL for `submit_new_host()`.

Default `None`

Environ *API_NEW_HOST*

darc.submit.**API_REQUESTS: str**
API URL for `submit_requests()`.

Default `None`

Environ *API_REQUESTS*

darc.submit.**API_SELENIUM: str**
API URL for `submit_selenium()`.

Default `None`

Environ *API_SELENIUM*

Note: If *API_NEW_HOST*, *API_REQUESTS* and *API_SELENIUM* is `None`, the corresponding submit function will save the JSON data in the path specified by *PATH_API*.

See also:

The *darc* provides a demo on how to implement a *darc*-compliant web backend for the data submission module. See the *demo* page for more information.

1.4 Proxy Utilities

The `darc.proxy` module provides various proxy support to the *darc* project.

darc.proxy.bitcoin.**PATH = '{PATH_MISC}/bitcoin.txt'**
Path to the data storage of bitcoin addresses.

See also:

- *darc.const.PATH_MISC*

darc.proxy.bitcoin.**LOCK: multiprocessing.Lock**
I/O lock for saving bitcoin addresses *PATH*.

darc.proxy.data.**PATH = '{PATH_MISC}/data/'**
Path to the data storage of data URI schemes.

See also:

- *darc.const.PATH_MISC*

darc.proxy.ed2k.**PATH = '{PATH_MISC}/ed2k.txt'**
Path to the data storage of bED2K magnet links.

See also:

- *darc.const.PATH_MISC*

darc.proxy.ed2k.**LOCK: multiprocessing.Lock**
I/O lock for saving ED2K magnet links *PATH*.

The following constants are configuration through environment variables:

darc.proxy.freenet.**FREENET_PORT: int**
Port for Freenet proxy connection.

Default 8888

Environ *FREENET_PORT*

darc.proxy.freenet.**FREENET_RETRY: int**
Retry times for Freenet bootstrap when failure.

Default 3

Environ *FREENET_RETRY*

darc.proxy.freenet.**BS_WAIT: float**
Time after which the attempt to start Freenet is aborted.

Default 90

Environ FREENET_WAIT

Note: If not provided, there will be **NO** timeouts.

---

darc.proxy.freenet.**FREENET_PATH: str**
Path to the Freenet project.

Default /usr/local/src/freenet

Environ *FREENET_PATH*

darc.proxy.freenet.**FREENET_ARGS: List[str]**
Freenet bootstrap arguments for `run.sh start`.

If provided, it should be parsed as command line arguments (c.f. `shlex.split`).

Default ''

Environ *FREENET_ARGS*

Note: The command will be run as *DARC_USER*, if current user (c.f. `getpass.getuser()`) is *root*.

The following constants are defined for internal usage:

darc.proxy.freenet.**_FREENET_BS_FLAG: bool**
If the Freenet proxy is bootstrapped.

darc.proxy.freenet.**_FREENET_PROC: subprocess.Popen**
Freenet proxy process running in the background.

darc.proxy.freenet.**_FREENET_ARGS: List[str]**
Freenet proxy bootstrap arguments.

1.4.1 I2P Proxy

The *darc.proxy.i2p* module contains the auxiliary functions around managing and processing the I2P proxy.

darc.proxy.i2p.**_i2p_bootstrap**()
I2P bootstrap.

The bootstrap arguments are defined as *_I2P_ARGS*.

Raises **subprocess.CalledProcessError** – If the return code of *_I2P_PROC* is non-zero.

See also:

- *darc.proxy.i2p.i2p_bootstrap()*
- *darc.proxy.i2p.BS_WAIT*
- *darc.proxy.i2p._I2P_BS_FLAG*
- *darc.proxy.i2p._I2P_PROC*

darc.proxy.i2p.**fetch_hosts**(*link*)
Fetch `hosts.txt`.

Parameters **link** (darc.link.Link) – Link object to fetch for its `hosts.txt`.

Returns Content of the `hosts.txt` file.

darc.proxy.i2p.**get_hosts**(*link*)
Read `hosts.txt`.

Parameters **link** (darc.link.Link) – Link object to read `hosts.txt`.

Returns

- If `hosts.txt` exists, return the data from `hosts.txt`.
  - `path` – relative path from `hosts.txt` to root of data storage *PATH_DB*, `<proxy>/<scheme>/<hostname>/hosts.txt`
  - `data` – *base64* encoded content of `hosts.txt`
- If not, return `None`.

Return type Optional[Dict[str, Union[str, ByteString]]]

See also:

- `darc.submit.submit_new_host()`
- *darc.proxy.i2p.save_hosts()*

darc.proxy.i2p.**have_hosts**(*link*)
Check if `hosts.txt` already exists.

Parameters **link** (darc.link.Link) – Link object to check if `hosts.txt` already exists.

Returns

- If `hosts.txt` exists, return the path to `hosts.txt`, i.e. `<root>/<proxy>/<scheme>/<hostname>/hosts.txt`.
- If not, return `None`.

Return type Optional[str]

darc.proxy.i2p.**i2p_bootstrap**()
Bootstrap wrapper for I2P.

The function will bootstrap the I2P proxy. It will retry for *I2P_RETRY* times in case of failure.

Also, it will **NOT** re-bootstrap the proxy as is guaranteed by *_I2P_BS_FLAG*.

Warns **I2PBootstrapFailed** – If failed to bootstrap I2P proxy.

Raises ***UnsupportedPlatform*** – If the system is not supported, i.e. not macOS or Linux.

See also:

- *darc.proxy.i2p._i2p_bootstrap()*
- *darc.proxy.i2p.I2P_RETRY*
- *darc.proxy.i2p._I2P_BS_FLAG*

darc.proxy.i2p.**read_hosts**(*text*, *check=False*)
Read hosts.txt.

Parameters

- **text** (*str*) – Content of hosts.txt.
- **check** (*bool*) – If perform checks on extracted links, default to *CHECK*.

Returns List of links extracted.

Return type List[*darc.link.Link*]

darc.proxy.i2p.**save_hosts**(*link*, *text*)
Save hosts.txt.

Parameters

- **link** (darc.link.Link) – Link object of hosts.txt.
- **text** (*str*) – Content of hosts.txt.

Returns Saved path to hosts.txt, i.e. <root>/<proxy>/<scheme>/<hostname>/hosts.txt.

Return type str

See also:

- darc.save.sanitise()

darc.proxy.i2p.**I2P_REQUESTS_PROXY: Dict[str, Any]**
Proxy for I2P sessions.

See also:

- darc.requests.i2p_session()

darc.proxy.i2p.**I2P_SELENIUM_PROXY: selenium.webdriver.Proxy**
Proxy (selenium.webdriver.Proxy) for I2P web drivers.

See also:

- darc.selenium.i2p_driver()

The following constants are configuration through environment variables:

darc.proxy.i2p.**I2P_PORT: int**
Port for I2P proxy connection.

Default 4444

Environ *I2P_PORT*

darc.proxy.i2p.**I2P_RETRY: int**
Retry times for I2P bootstrap when failure.

Default 3

Environ *I2P_RETRY*

darc.proxy.i2p.**BS_WAIT: float**
Time after which the attempt to start I2P is aborted.

Default 90

Environ I2P_WAIT

Note: If not provided, there will be **NO** timeouts.

---

darc.proxy.i2p.**I2P_ARGS: List[str]**
I2P bootstrap arguments for `i2prouter start`.

If provided, it should be parsed as command line arguments (c.f. `shlex.split`).

Default ''

Environ *I2P_ARGS*

Note: The command will be run as *DARC_USER*, if current user (c.f. `getpass.getuser()`) is *root*.

The following constants are defined for internal usage:

darc.proxy.i2p.**_I2P_BS_FLAG: bool**
If the I2P proxy is bootstrapped.

darc.proxy.i2p.**_I2P_PROC: subprocess.Popen**
I2P proxy process running in the background.

darc.proxy.i2p.**_I2P_ARGS: List[str]**
I2P proxy bootstrap arguments.

darc.proxy.irc.**PATH = '{PATH_MISC}/irc.txt'**
Path to the data storage of IRC addresses.

See also:

- *darc.const.PATH_MISC*

darc.proxy.irc.**LOCK: multiprocessing.Lock**
I/O lock for saving IRC addresses *PATH*.

darc.proxy.magnet.**PATH = '{PATH_MISC}/magnet.txt'**
Path to the data storage of magnet links.

See also:

- *darc.const.PATH_MISC*

darc.proxy.magnet.**LOCK: multiprocessing.Lock**
I/O lock for saving magnet links *PATH*.

darc.proxy.mail.**PATH = '{PATH_MISC}/mail.txt'**
Path to the data storage of email addresses.

See also:

- *darc.const.PATH_MISC*

darc.proxy.mail.**LOCK: multiprocessing.Lock**
I/O lock for saving email addresses *PATH*.

darc.proxy.null.**PATH = '{PATH_MISC}/invalid.txt'**
Path to the data storage of links with invalid scheme.

See also:

- *darc.const.PATH_MISC*

darc.proxy.null.**LOCK: multiprocessing.Lock**
I/O lock for saving links with invalid scheme *PATH*.

darc.proxy.tor.**TOR_REQUESTS_PROXY: Dict[str, Any]**
Proxy for Tor sessions.

See also:

- darc.requests.tor_session()

darc.proxy.tor.**TOR_SELENIUM_PROXY: selenium.webdriver.Proxy**
Proxy (selenium.webdriver.Proxy) for Tor web drivers.

See also:

- darc.selenium.tor_driver()

The following constants are configuration through environment variables:

darc.proxy.tor.**TOR_PORT: int**
Port for Tor proxy connection.

Default 9050

Environ *TOR_PORT*

darc.proxy.tor.**TOR_CTRL: int**
Port for Tor controller connection.

Default 9051

Environ *TOR_CTRL*

darc.proxy.tor.**TOR_STEM: bool**
If manage the Tor proxy through stem.

Default True

Environ *TOR_STEM*

darc.proxy.tor.**TOR_PASS: str**
Tor controller authentication token.

Default None

Environ *TOR_PASS*

Note: If not provided, it will be requested at runtime.

darc.proxy.tor.**TOR_RETRY: int**
Retry times for Tor bootstrap when failure.

Default 3

Environ *TOR_RETRY*

darc.proxy.tor.**BS_WAIT: float**
Time after which the attempt to start Tor is aborted.

Default 90

Environ TOR_WAIT

Note: If not provided, there will be **NO** timeouts.

darc.proxy.tor.**TOR_CFG: Dict[str, Any]**
Tor bootstrap configuration for stem.process.launch_tor_with_config().

Default {}

Environ *TOR_CFG*

Note: If provided, it will be parsed from a JSON encoded string.

The following constants are defined for internal usage:

darc.proxy.tor.**_TOR_BS_FLAG: bool**
If the Tor proxy is bootstrapped.

darc.proxy.tor.**_TOR_PROC: subprocess.Popen**
Tor proxy process running in the background.

darc.proxy.tor.**_TOR_CTRL: stem.control.Controller**
Tor controller process (stem.control.Controller) running in the background.

darc.proxy.tor.**_TOR_CONFIG: List[str]**
Tor bootstrap configuration for stem.process.launch_tor_with_config().

The following constants are configuration through environment variables:

darc.proxy.zeronet.**ZERONET_PORT: int**
Port for ZeroNet proxy connection.

Default 43110

Environ *ZERONET_PORT*

darc.proxy.zeronet.**ZERONET_RETRY: int**
Retry times for ZeroNet bootstrap when failure.

Default 3

Environ *ZERONET_RETRY*

darc.proxy.zeronet.**BS_WAIT: float**
Time after which the attempt to start ZeroNet is aborted.

Default 90

Environ ZERONET_WAIT

Note: If not provided, there will be **NO** timeouts.

---

darc.proxy.zeronet.**ZERONET_PATH: str**
Path to the ZeroNet project.

Default /usr/local/src/zeronet

Environ *ZERONET_PATH*

darc.proxy.zeronet.**ZERONET_ARGS: List[str]**
ZeroNet bootstrap arguments for `run.sh start`.

If provided, it should be parsed as command line arguments (c.f. `shlex.split`).

Default ''

Environ *ZERONET_ARGS*

Note: The command will be run as *DARC_USER*, if current user (c.f. `getpass.getuser()`) is *root*.

The following constants are defined for internal usage:

darc.proxy.zeronet.**_ZERONET_BS_FLAG: bool**
If the ZeroNet proxy is bootstrapped.

darc.proxy.zeronet.**_ZERONET_PROC: subprocess.Popen**
ZeroNet proxy process running in the background.

darc.proxy.zeronet.**_ZERONET_ARGS: List[str]**
ZeroNet proxy bootstrap arguments.

To tell the *darc* project which proxy settings to be used for the `requests.Session` objects and `selenium.webdriver.Chrome` objects, you can specify such information in the *darc.proxy.LINK_MAP* mapping dictionarty.

darc.proxy.**LINK_MAP: DefaultDict[str, Tuple[types.FunctionType, types.FunctionType]]**

```
LINK_MAP = collections.defaultdict(
    lambda: (darc.requests.null_session, darc.selenium.null_driver),
    dict(
        tor=(darc.requests.tor_session, darc.selenium.tor_driver),
        i2p=(darc.requests.i2p_session, darc.selenium.i2p_driver),
    )
)
```

The mapping dictionary for proxy type to its corresponding `requests.Session` factory function and `selenium.webdriver.Chrome` factory function.

The fallback value is the no proxy `requests.Session` object (`null_session()`) and `selenium.webdriver.Chrome` object (`null_driver()`).

See also:

- `darc.requests` – `requests.Session` factory functions
- `darc.selenium` – `selenium.webdriver.Chrome` factory functions

1.5 Sites Customisation

As websites may have authentication requirements, etc., over its content, the `darc.sites` module provides sites customisation hooks to both |requests|_ and |selenium|_ crawling processes.

```python
# -*- coding: utf-8 -*-
"""Default Hooks
==================

The :mod:`darc.sites.default` module is the fallback for sites
customisation.

"""

import time

import darc.typing as typing
from darc.const import SE_WAIT
from darc.link import Link


def crawler(session: typing.Session, link: Link) -> typing.Response:
    """Default crawler hook.

    Args:
        session (|Session|_): Session object with proxy settings.
        link: Link object to be crawled.

    Returns:
        |Response|_: The final response object with crawled data.

    See Also:
        * :func:`darc.crawl.crawler`

    """
    response = session.get(link.url, allow_redirects=True)
    return response


def loader(driver: typing.Driver, link: Link) -> typing.Driver:
    """Default loader hook.

    When loading, if :data:`~darc.const.SE_WAIT` is a valid time lapse,
    the function will sleep for such time to wait for the page to finish
    loading contents.

    Args:
```

(continues on next page)

(continued from previous page)

```
        driver (|Chrome|_): Web driver object with proxy settings.
        link: Link object to be loaded.

    Returns:
        |Chrome|_: The web driver object with loaded data.

    Note:
        Internally, |selenium|_ will wait for the browser to finish
        loading the pages before return (i.e. the web API event
        |event|_). However, some extra scripts may take more time
        running after the event.

        .. |event| replace:: ``DOMContentLoaded``
        .. _event: https://developer.mozilla.org/en-US/docs/Web/API/Window/
↪DOMContentLoaded_event

    See Also:
        * :func:`darc.crawl.loader`
        * :data:`darc.const.SE_WAIT`

    """
    driver.get(link.url)

    # wait for page to finish loading
    if SE_WAIT is not None:
        time.sleep(SE_WAIT)

    return driver
```

To customise behaviours over **|requests|_**, you sites customisation module should have a `crawler()` function, e.g. `crawler()`.

The function takes the **|Session|_** object with proxy settings and a *Link* object representing the link to be crawled, then returns a **|Response|_** object containing the final data of the crawling process.

To customise behaviours over **|selenium|_**, you sites customisation module should have a `loader()` function, e.g. `loader()`.

The function takes the **|Chrome|_** object with proxy settings and a *Link* object representing the link to be loaded, then returns the **|Chrome|_** object containing the final data of the loading process.

1.6 Module Constants

1.6.1 Auxiliary Function

1.6.2 General Configurations

`darc.const.`**`REBOOT`**`: bool`

If exit the program after first round, i.e. crawled all links from the `requests` link database and loaded all links from the `selenium` link database.

This can be useful especially when the capacity is limited and you wish to save some space before continuing next round. See *Docker integration* for more information.

Default `False`

Environ *DARC_REBOOT*

darc.const.**DEBUG: bool**
If run the program in debugging mode.

Default `False`

Environ *DARC_DEBUG*

darc.const.**VERBOSE: bool**
If run the program in verbose mode. If *DEBUG* is `True`, then the verbose mode will be always enabled.

Default `False`

Environ *DARC_VERBOSE*

darc.const.**FORCE: bool**
If ignore `robots.txt` rules when crawling (c.f. `crawler()`).

Default `False`

Environ *DARC_FORCE*

darc.const.**CHECK: bool**
If check proxy and hostname before crawling (when calling *extract_links()*, `read_sitemap()` and *read_hosts()*).

If *CHECK_NG* is `True`, then this environment variable will be always set as `True`.

Default `False`

Environ *DARC_CHECK*

darc.const.**CHECK_NG: bool**
If check content type through `HEAD` requests before crawling (when calling *extract_links()*, `read_sitemap()` and *read_hosts()*).

Default `False`

Environ *DARC_CHECK_CONTENT_TYPE*

darc.const.**ROOT: str**
The root folder of the project.

darc.const.**CWD = '.'**
The current working direcory.

darc.const.**DARC_CPU: int**
Number of concurrent processes. If not provided, then the number of system CPUs will be used.

Default `None`

Environ *DARC_CPU*

darc.const.**FLAG_MP: bool**
If enable *multiprocessing* support.

Default `True`

Environ *DARC_MULTIPROCESSING*

darc.const.**FLAG_TH: bool**
If enable *multithreading* support.

Default `False`

Environ *DARC_MULTITHREADING*

Note: *FLAG_MP* and *FLAG_TH* can **NOT** be toggled at the same time.

darc.const.**DARC_USER: str**
Non-root user for proxies.

Default current login user (c.f. `getpass.getuser()`)

Environ *DARC_USER*

1.6.3 Data Storage

darc.const.**REDIS: redis.Redis**
URL to the Redis database.

Default `redis://127.0.0.1`

Environ *REDIS_URL*

darc.const.**PATH_DB: str**
Path to data storage.

Default `data`

Environ *PATH_DATA*

See also:

See `darc.save` for more information about source saving.

darc.const.**PATH_MISC = '{PATH_DB}/misc/'**
Path to miscellaneous data storage, i.e. `misc` folder under the root of data storage.

See also:

- *darc.const.PATH_DB*

darc.const.**PATH_LN = '{PATH_DB}/link.csv'**
Path to the link CSV file, `link.csv`.

See also:

- *darc.const.PATH_DB*
- `darc.save.save_link`

darc.const.**PATH_QR = '{PATH_DB}/_queue_requests.txt'**
Path to the `requests` database, `_queue_requests.txt`.

See also:

- *darc.const.PATH_DB*
- *darc.db.load_requests()*
- *darc.db.save_requests()*

darc.const.**PATH_QS = '{PATH_DB}/_queue_selenium.txt'**
Path to the `selenium` database, `_queue_selenium.txt`.

See also:

- *darc.const.PATH_DB*
- *darc.db.load_selenium()*
- *darc.db.save_selenium()*

darc.const.**PATH_ID = '{PATH_DB}/darc.pid'**
Path to the process ID file, `darc.pid`.

See also:

- *darc.const.PATH_DB*
- `darc.const.getpid()`

1.6.4 Web Crawlers

darc.const.**DARC_WAIT: Optional[float]**
Time interval between each round when the `requests` and/or `selenium` database are empty.

Default `60`

Environ *DARC_WAIT*

darc.const.**SAVE: bool**
If save processed link back to database.

Note: If *SAVE* is `True`, then *SAVE_REQUESTS* and *SAVE_SELENIUM* will be forced to be `True`.

Default `False`

Environ *DARC_SAVE*

See also:

See *darc.db* for more information about link database.

darc.const.**SAVE_REQUESTS: bool**
If save `crawler()` crawled link back to `requests` database.

Default `False`

Environ *DARC_SAVE_REQUESTS*

See also:

See *darc.db* for more information about link database.

darc.const.**SAVE_SELENIUM: bool**
If save `loader()` crawled link back to `selenium` database.

Default `False`

Environ *DARC_SAVE_SELENIUM*

See also:

See *darc.db* for more information about link database.

darc.const.**TIME_CACHE: float**
Time delta for caches in seconds.

The *darc* project supports *caching* for fetched files. *TIME_CACHE* will specify for how log the fetched files will be cached and **NOT** fetched again.

Note: If *TIME_CACHE* is `None` then caching will be marked as *forever*.

Default `60`

Environ *TIME_CACHE*

darc.const.**SE_WAIT: float**
Time to wait for `selenium` to finish loading pages.

Note: Internally, `selenium` will wait for the browser to finish loading the pages before return (i.e. the web API event `DOMContentLoaded`). However, some extra scripts may take more time running after the event.

Default `60`

Environ *SE_WAIT*

darc.const.**SE_EMPTY = '<html><head></head><body></body></html>'**
The empty page from `selenium`.

See also:

- `darc.crawl.loader()`

1.6.5 White / Black Lists

darc.const.**LINK_WHITE_LIST: List[re.Pattern]**
White list of hostnames should be crawled.

Default `[]`

Environ *LINK_WHITE_LIST*

Note: Regular expressions are supported.

darc.const.**LINK_BLACK_LIST: List[re.Pattern]**
Black list of hostnames should be crawled.

Default `[]`

Environ *LINK_BLACK_LIST*

Note: Regular expressions are supported.

darc.const.**LINK_FALLBACK: bool**
Fallback value for *match_host()*.

Default `False`

Environ *LINK_FALLBACK*

darc.const.**MIME_WHITE_LIST: List[re.Pattern]**
White list of content types should be crawled.

Default `[]`

Environ *MIME_WHITE_LIST*

Note: Regular expressions are supported.

---

darc.const.**MIME_BLACK_LIST: List[re.Pattern]**
Black list of content types should be crawled.

Default `[]`

Environ *MIME_BLACK_LIST*

Note: Regular expressions are supported.

---

darc.const.**MIME_FALLBACK: bool**
Fallback value for *match_mime()*.

Default `False`

Environ *MIME_FALLBACK*

darc.const.**PROXY_WHITE_LIST: List[str]**
White list of proxy types should be crawled.

Default `[]`

Environ *PROXY_WHITE_LIST*

Note: The proxy types are **case insensitive**.

---

darc.const.**PROXY_BLACK_LIST: List[str]**
Black list of proxy types should be crawled.

Default `[]`

Environ *PROXY_BLACK_LIST*

Note: The proxy types are **case insensitive**.

---

darc.const.**PROXY_FALLBACK: bool**
Fallback value for *match_proxy()*.

Default `False`

Environ *PROXY_FALLBACK*

1.7 Custom Exceptions

The *render_error()* function can be used to render multi-line error messages with stem.util.term colours.

The *darc* project provides following custom exceptions:

- *LinkNoReturn*
- *UnsupportedLink*
- *UnsupportedPlatform*
- *UnsupportedProxy*

The *darc* project provides following custom exceptions:

- *TorBootstrapFailed*
- *I2PBootstrapFailed*
- *ZeroNetBootstrapFailed*
- *FreenetBootstrapFailed*
- *APIRequestFailed*
- *SiteNotFoundWarning*
- *LockWarning*
- *TorRenewFailed*
- *RedisConnectionFailed*

exception darc.error.**APIRequestFailed**
Bases: Warning

API submit failed.

exception darc.error.**FreenetBootstrapFailed**
Bases: Warning

Freenet bootstrap process failed.

exception darc.error.**I2PBootstrapFailed**
Bases: Warning

I2P bootstrap process failed.

exception darc.error.**LinkNoReturn**
Bases: Exception

The link has no return value from the hooks.

exception darc.error.**LockWarning**
Bases: Warning

Failed to acquire Redis lock.

exception darc.error.**RedisConnectionFailed**
Bases: Warning

Redis connection failed.

exception darc.error.**SiteNotFoundWarning**
Bases: ImportWarning

Site customisation not found.

exception darc.error.**TorBootstrapFailed**
Bases: Warning

Tor bootstrap process failed.

exception darc.error.**TorRenewFailed**
Bases: Warning

Tor renew request failed.

exception darc.error.**UnsupportedLink**
Bases: Exception

The link is not supported.

exception darc.error.**UnsupportedPlatform**
Bases: Exception

The platform is not supported.

exception darc.error.**UnsupportedProxy**
Bases: Exception

The proxy is not supported.

exception darc.error.**ZeroNetBootstrapFailed**
Bases: Warning

ZeroNet bootstrap process failed.

darc.error.**render_error**(*message*, *colour*)
Render error message.

The function wraps the stem.util.term.format() function to provide multi-line formatting support.

Parameters

- **message** (*str*) – Multi-line message to be rendered with colour.
- **colour** (*stem.util.term.Color*) – Front colour of text, c.f. stem.util.term.Color.

Returns The rendered error message.

Return type str

As the websites can be sometimes irritating for their anti-robots verification, login requirements, etc., the *darc* project also privides hooks to customise crawling behaviours around both requests and selenium.

See also:

Such customisation, as called in the *darc* project, site hooks, is site specific, user can set up your own hooks unto a certain site, c.f. darc.sites for more information.

Still, since the network is a world full of mysteries and miracles, the speed of crawling will much depend on the response speed of the target website. To boost up, as well as meet the system capacity, the *darc* project introduced multiprocessing, multithreading and the fallback slowest single-threaded solutions when crawling.

Note: When rendering the target website using selenium powered by the renown Google Chrome, it will require much memory to run. Thus, the three solutions mentioned above would only toggle the behaviour around the use of selenium.

To keep the *darc* project as it is a swiss army knife, only the main entrypoint function darc.process.process() is exported in global namespace (and renamed to darc.darc()), see below:

CHAPTER

TWO

WEB BACKEND DEMO

This is a demo of API for communication between the *darc* crawlers (`darc.submit`) and web UI.

Assuming the web UI is developed using the `Flask` microframework.

```python
# -*- coding: utf-8 -*-

import flask  # pylint: disable=import-error

# Flask application
app = flask.Flask(__file__)


@app.route('/api/new_host', methods=['POST'])
def new_host():
    """When a new host is discovered, the :mod:`darc` crawler will submit the
    host information. Such includes ``robots.txt`` (if exists) and
    ``sitemap.xml`` (if any).

    Data format::

        {
            // metadata of URL
            "[metadata]": {
                // original URL - <scheme>://<netloc>/<path>;<params>?<query>#
↪<fragment>
                "url": ...,
                // proxy type - null / tor / i2p / zeronet / freenet
                "proxy": ...,
                // hostname / netloc, c.f. ``urllib.parse.urlparse``
                "host": ...,
                // base folder, relative path (to data root path ``PATH_DATA``) in␣
↪containter - <proxy>/<scheme>/<host>
                "base": ...,
                // sha256 of URL as name for saved files (timestamp is in ISO format)
                //   JSON log as this one - <base>/<name>_<timestamp>.json
                //   HTML from requests - <base>/<name>_<timestamp>_raw.html
                //   HTML from selenium - <base>/<name>_<timestamp>.html
                //   generic data files - <base>/<name>_<timestamp>.dat
                "name": ...
            },
            // requested timestamp in ISO format as in name of saved file
            "Timestamp": ...,
            // original URL
            "URL": ...,
```

(continues on next page)

(continued from previous page)

```
            // robots.txt from the host (if not exists, then ``null``)
            "Robots": {
                // path of the file, relative path (to data root path ``PATH_DATA``)␣
↪in container
                //   - <proxy>/<scheme>/<host>/robots.txt
                "path": ...,
                // content of the file (**base64** encoded)
                "data": ...,
            },
            // sitemaps from the host (if none, then ``null``)
            "Sitemaps": [
                {
                    // path of the file, relative path (to data root path ``PATH_
↪DATA``) in container
                    //   - <proxy>/<scheme>/<host>/sitemap_<name>.txt
                    "path": ...,
                    // content of the file (**base64** encoded)
                    "data": ...,
                },
                ...
            ],
            // hosts.txt from the host (if proxy type is ``i2p``; if not exists, then␣
↪``null``)
            "Hosts": {
                // path of the file, relative path (to data root path ``PATH_DATA``)␣
↪in container
                //   - <proxy>/<scheme>/<host>/hosts.txt
                "path": ...,
                // content of the file (**base64** encoded)
                "data": ...,
            }
        }

    """
    # JSON data from the request
    data = flask.request.json  # pylint: disable=unused-variable

    # do whatever processing needed
    ...


@app.route('/api/requests', methods=['POST'])
def from_requests():
    """When crawling, we'll first fetch the URl using ``requests``, to check
    its availability and to save its HTTP headers information. Such information
    will be submitted to the web UI.

    Data format::

        {
            // metadata of URL
            "[metadata]": {
                // original URL - <scheme>://<netloc>/<path>;<params>?<query>#
↪<fragment>
                "url": ...,
                // proxy type - null / tor / i2p / zeronet / freenet
                "proxy": ...,
```

(continues on next page)

(continued from previous page)

```
                // hostname / netloc, c.f. ``urllib.parse.urlparse``
                "host": ...,
                // base folder, relative path (to data root path ``PATH_DATA``) in␣
↪containter - <proxy>/<scheme>/<host>
                "base": ...,
                // sha256 of URL as name for saved files (timestamp is in ISO format)
                //   JSON log as this one - <base>/<name>_<timestamp>.json
                //   HTML from requests - <base>/<name>_<timestamp>_raw.html
                //   HTML from selenium - <base>/<name>_<timestamp>.html
                //   generic data files - <base>/<name>_<timestamp>.dat
                "name": ...
            },
            // requested timestamp in ISO format as in name of saved file
            "Timestamp": ...,
            // original URL
            "URL": ...,
            // request method
            "Method": "GET",
            // response status code
            "Status-Code": ...,
            // response reason
            "Reason": ...,
            // response cookies (if any)
            "Cookies": {
                ...
            },
            // session cookies (if any)
            "Session": {
                ...
            },
            // request headers (if any)
            "Request": {
                ...
            },
            // response headers (if any)
            "Response": {
                ...
            },
            // requested file (if not exists, then ``null``)
            "Document": {
                // path of the file, relative path (to data root path ``PATH_DATA``)␣
↪in container
                //   - <proxy>/<scheme>/<host>/<name>_<timestamp>_raw.html
                // or if the document is of generic content type, i.e. not HTML
                //   - <proxy>/<scheme>/<host>/<name>_<timestamp>.dat
                "path": ...,
                // content of the file (**base64** encoded)
                "data": ...,
            },
            // redirection history (if any)
            "History": [
                // same records as the original response
                {"...": "..."}
            ]
        }

    """
```

(continues on next page)

(continued from previous page)

```
    # JSON data from the request
    data = flask.request.json  # pylint: disable=unused-variable

    # do whatever processing needed
    ...


@app.route('/api/selenium', methods=['POST'])
def from_selenium():
    """After crawling with ``requests``, we'll then render the URl using
    ``selenium`` with Google Chrome and its driver, to provide a fully rendered
    web page. Such information will be submitted to the web UI.

    Note:
        This information is optional, only provided if the content type from
        ``requests`` is HTML, status code < 400, and HTML data not empty.

    Data format::

        {
            // metadata of URL
            "[metadata]": {
                // original URL - <scheme>://<netloc>/<path>;<params>?<query>#
↪<fragment>
                "url": ...,
                // proxy type - null / tor / i2p / zeronet / freenet
                "proxy": ...,
                // hostname / netloc, c.f. ``urllib.parse.urlparse``
                "host": ...,
                // base folder, relative path (to data root path ``PATH_DATA``) in␣
↪containter - <proxy>/<scheme>/<host>
                "base": ...,
                // sha256 of URL as name for saved files (timestamp is in ISO format)
                //   JSON log as this one - <base>/<name>_<timestamp>.json
                //   HTML from requests - <base>/<name>_<timestamp>_raw.html
                //   HTML from selenium - <base>/<name>_<timestamp>.html
                //   generic data files - <base>/<name>_<timestamp>.dat
                "name": ...
            },
            // requested timestamp in ISO format as in name of saved file
            "Timestamp": ...,
            // original URL
            "URL": ...,
            // rendered HTML document (if not exists, then ``null``)
            "Document": {
                // path of the file, relative path (to data root path ``PATH_DATA``)␣
↪in container
                //   - <proxy>/<scheme>/<host>/<name>_<timestamp>.html
                "path": ...,
                // content of the file (**base64** encoded)
                "data": ...,
            },
            // web page screenshot (if not exists, then ``null``)
            "Screenshot": {
                // path of the file, relative path (to data root path ``PATH_DATA``)␣
↪in container
                //   - <proxy>/<scheme>/<host>/<name>_<timestamp>.png
```

(continues on next page)

(continued from previous page)

```
                "path": ...,
                // content of the file (**base64** encoded)
                "data": ...,
            }
        }

    """
    # JSON data from the request
    data = flask.request.json  # pylint: disable=unused-variable

    # do whatever processing needed
    ...


if __name__ == "__main__":
    flask.run()
```

CHAPTER

THREE

DOCKER INTEGRATION

The *darc* project is integrated with Docker and Compose. Though published to Docker Hub, you can still build by yourself.

Important: The `debug` image contains miscellaneous documents, i.e. whole repository in it; and pre-installed some useful tools for debugging, such as IPython, etc.

The Docker image is based on Ubuntu Bionic (18.04 LTS), setting up all Python dependencies for the *darc* project, installing Google Chrome (version 79.0.3945.36) and corresponding ChromeDriver, as well as installing and configuring Tor, I2P, ZeroNet, FreeNet, NoIP proxies.

Note: NoIP is currently not fully integrated in the *darc* due to misunderstanding in the configuration process. Contributions are welcome.

When building the image, there is an *optional* argument for setting up a *non-root* user, c.f. environment variable `DARC_USER` and module constant *DARC_USER*. By default, the username is `darc`.

```
FROM ubuntu:bionic

LABEL Name=darc \
      Version=0.5.0rc2

STOPSIGNAL SIGINT
HEALTHCHECK --interval=1h --timeout=1m \
    CMD wget https://httpbin.org/get -O /dev/null || exit 1

ARG DARC_USER="darc"
ENV LANG="C.UTF-8" \
    LC_ALL="C.UTF-8" \
    PYTHONIOENCODING="UTF-8" \
    DEBIAN_FRONTEND="teletype" \
    DARC_USER="${DARC_USER}"
    # DEBIAN_FRONTEND="noninteractive"

COPY extra/retry.sh /usr/local/bin/retry
COPY extra/install.py /usr/local/bin/pty-install
COPY vendor/jdk-11.0.7_linux-x64_bin.tar.gz /var/cache/oracle-jdk11-installer-local/

RUN set -x \
 && retry apt-get update \
 && retry apt-get install --yes --no-install-recommends \
```

(continues on next page)

(continued from previous page)

```
        apt-utils \
 && retry apt-get install --yes --no-install-recommends \
        gcc \
        g++ \
        libmagic1 \
        make \
        software-properties-common \
        tar \
        unzip \
        zlib1g-dev \
 && retry add-apt-repository ppa:deadsnakes/ppa --yes \
 && retry add-apt-repository ppa:linuxuprising/java --yes \
 && retry add-apt-repository ppa:i2p-maintainers/i2p --yes
RUN retry apt-get update \
 && retry apt-get install --yes --no-install-recommends \
        python3.8 \
        python3-pip \
        python3-setuptools \
        python3-wheel \
 && ln -sf /usr/bin/python3.8 /usr/local/bin/python3
RUN retry pty-install --stdin '6\n70' apt-get install --yes --no-install-recommends \
        tzdata \
 && retry pty-install --stdin 'yes' apt-get install --yes \
        oracle-java11-installer-local
RUN retry apt-get install --yes --no-install-recommends \
        sudo \
 && adduser --disabled-password --gecos '' ${DARC_USER} \
 && adduser ${DARC_USER} sudo \
 && echo '%sudo ALL=(ALL) NOPASSWD:ALL' >> /etc/sudoers

## Tor
RUN retry apt-get install --yes --no-install-recommends tor
COPY extra/torrc.bionic /etc/tor/torrc

## I2P
RUN retry apt-get install --yes --no-install-recommends i2p
COPY extra/i2p.bionic /etc/defaults/i2p

## ZeroNet
COPY vendor/ZeroNet-py3-linux64.tar.gz /tmp
RUN set -x \
 && cd /tmp \
 && tar xvpfz ZeroNet-py3-linux64.tar.gz \
 && mv ZeroNet-linux-dist-linux64 /usr/local/src/zeronet
COPY extra/zeronet.bionic.conf /usr/local/src/zeronet/zeronet.conf

## FreeNet
USER darc
COPY vendor/new_installer_offline.jar /tmp
RUN set -x \
 && cd /tmp \
 && ( pty-install --stdin '/home/darc/freenet\n1' java -jar new_installer_offline.jar␣
↪|| true ) \
 && sudo mv /home/darc/freenet /usr/local/src/freenet
USER root

## NoIP
```

(continues on next page)

(continued from previous page)

```
COPY vendor/noip-duc-linux.tar.gz /tmp
RUN set -x \
 && cd /tmp \
 && tar xvpfz noip-duc-linux.tar.gz \
 && mv noip-2.1.9-1 /usr/local/src/noip \
 && cd /usr/local/src/noip \
 && make
 # && make install

# # set up timezone
# RUN echo 'Asia/Shanghai' > /etc/timezone \
#  && rm -f /etc/localtime \
#  && ln -snf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime \
#  && dpkg-reconfigure -f noninteractive tzdata

COPY vendor/chromedriver_linux64-79.0.3945.36.zip \
     vendor/google-chrome-stable_current_amd64.deb /tmp/
RUN set -x \
 ## ChromeDriver
 && unzip -d /usr/bin /tmp/chromedriver_linux64-79.0.3945.36.zip \
 && which chromedriver \
 ## Google Chrome
 && ( dpkg --install /tmp/google-chrome-stable_current_amd64.deb || true ) \
 && retry apt-get install --fix-broken --yes --no-install-recommends \
 && dpkg --install /tmp/google-chrome-stable_current_amd64.deb \
 && which google-chrome

# Using pip:
COPY requirements.txt /tmp
RUN python3 -m pip install -r /tmp/requirements.txt --no-cache-dir

RUN set -x \
 && rm -rf \
        ## APT repository lists
        /var/lib/apt/lists/* \
        ## Python dependencies
        /tmp/requirements.txt \
        /tmp/pip \
        ## ChromeDriver
        /tmp/chromedriver_linux64-79.0.3945.36.zip \
        ## Google Chrome
        /tmp/google-chrome-stable_current_amd64.deb \
        ## Vendors
        /tmp/new_installer_offline.jar \
        /tmp/noip-duc-linux.tar.gz \
        /tmp/ZeroNet-py3-linux64.tar.gz \
 #&& apt-get remove --auto-remove --yes \
 #       software-properties-common \
 #       unzip \
 && apt-get autoremove -y \
 && apt-get autoclean \
 && apt-get clean

ENTRYPOINT [ "python3", "-m", "darc" ]
#ENTRYPOINT [ "bash", "/app/run.sh" ]
CMD [ "--help" ]
```

(continues on next page)

(continued from previous page)

```
WORKDIR /app
COPY darc/ /app/darc/
COPY LICENSE \
     MANIFEST.in \
     README.rst \
     extra/run.sh \
     setup.cfg \
     setup.py \
     test_darc.py /app/
RUN python3 -m pip install -e .
```

Note:

- `retry` is a shell script for retrying the commands until success

```
#!/usr/bin/env bash

while true; do
    >&2 echo "+ $@"
    $@ && break
    >&2 echo "exit: $?"
done
>&2 echo "exit: 0"
```

- `pty-install` is a Python script simulating user input for APT package installation with `DEBIAN_FRONTEND` set as `Teletype`.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""Install packages requiring interactions."""

import argparse
import os
import subprocess
import sys
import tempfile


def get_parser():
    """Argument parser."""
    parser = argparse.ArgumentParser('install',
                                     description='pseudo-interactive package installer
↪')

    parser.add_argument('-i', '--stdin', help='content for input')
    parser.add_argument('command', nargs=argparse.REMAINDER, help='command to execute
↪')

    return parser


def main():
    """Entrypoint."""
    parser = get_parser()
    args = parser.parse_args()
    text = args.stdin.encode().decode('unicode_escape')
```

(continues on next page)

(continued from previous page)

```
    path = tempfile.mktemp(prefix='install-')
    with open(path, 'w') as file:
        file.write(text)

    with open(path, 'r') as file:
        proc = subprocess.run(args.command, stdin=file)  # pylint: disable=subprocess-
↪run-check

    os.remove(path)
    return proc.returncode


if __name__ == "__main__":
    sys.exit(main())
```

As always, you can also use Docker Compose to manage the *darc* image. Environment variables can be set as described in the configuration section.

```yaml
version: '3'

services:
  crawler:
    image: jsnbzh/darc:latest
    build: &build
      context: .
      args:
        # non-root user
        DARC_USER: "darc"
    container_name: crawler
    #entrypoint: [ "bash", "/app/run.sh" ]
    command: [ "--type", "crawler",
               "--file", "/app/text/market.txt",
               "--file", "/app/text/top-20k.txt",
               "--file", "/app/text/i2p.txt",
               "--file", "/app/text/zeronet.txt",
               "--file", "/app/text/freenet.txt" ]
    depends_on: &depends_on
      - redis
    environment:
      ## [PYTHON] force the stdout and stderr streams to be unbuffered
      PYTHONUNBUFFERED: 1
      # reboot mode
      DARC_REBOOT: 1
      # debug mode
      DARC_DEBUG: 0
      # verbose mode
      DARC_VERBOSE: 1
      # force mode (ignore robots.txt)
      DARC_FORCE: 1
      # check mode (check proxy and hostname before crawling)
      DARC_CHECK: 1
      # check mode (check content type before crawling)
      DARC_CHECK_CONTENT_TYPE: 0
      # save mode
```

(continues on next page)

(continued from previous page)

```
      DARC_SAVE: 0
      # save mode (for requests)
      DAVE_SAVE_REQUESTS: 0
      # save mode (for selenium)
      DAVE_SAVE_SELENIUM: 0
      # processes
      DARC_CPU: 16
      # multiprocessing
      DARC_MULTIPROCESSING: 1
      # multithreading
      DARC_MULTITHREADING: 0
      # time lapse
      DARC_WAIT: 60
      # data storage
      PATH_DATA: "data"
      # Redis URL
      REDIS_URL: 'redis://
↪:UCf7y123aHgaYeGnvLRasALjFfDVHGCz6KiR5Z0WC0DL4ExvSGw5SkcOxBywc0qtZBHVrSVx2QMGewXNP6qVow@redis
↪'
      # max pool
      DARC_MAX_POOL: 1_000
      # Tor proxy & control port
      TOR_PORT: 9050
      TOR_CTRL: 9051
      # Tor management method
      TOR_STEM: 1
      # Tor authentication
      TOR_PASS: "16:B9D36206B5374B3F609045F9609EE670F17047D88FF713EFB9157EA39F"
      # Tor bootstrap retry
      TOR_RETRY: 10
      # Tor bootstrap wait
      TOR_WAIT: 90
      # Tor bootstrap config
      TOR_CFG: "{}"
      # I2P port
      I2P_PORT: 4444
      # I2P bootstrap retry
      I2P_RETRY: 10
      # I2P bootstrap wait
      I2P_WAIT: 90
      # I2P bootstrap config
      I2P_ARGS: ""
      # ZeroNet port
      ZERONET_PORT: 43110
      # ZeroNet bootstrap retry
      ZERONET_RETRY: 10
      # ZeroNet project path
      ZERONET_PATH: "/usr/local/src/zeronet"
      # ZeroNet bootstrap wait
      ZERONET_WAIT: 90
      # ZeroNet bootstrap config
      ZERONET_ARGS: ""
      # Freenet port
      FREENET_PORT: 8888
      # Freenet bootstrap retry
      FREENET_RETRY: 0
      # Freenet project path
```

(continues on next page)

(continued from previous page)

```
      FREENET_PATH: "/usr/local/src/freenet"
      # Freenet bootstrap wait
      FREENET_WAIT: 90
      # Freenet bootstrap config
      FREENET_ARGS: ""
      # time delta for caches in seconds
      TIME_CACHE: 2_592_000  # 30 days
      # time to wait for selenium
      SE_WAIT: 5
      # extract link pattern
      LINK_WHITE_LIST: '[
        ".*?\\.onion",
        ".*?\\.i2p", "127\\.0\\.0\\.1:7657", "localhost:7657", "127\\.0\\.0\\.1:7658",
↪ "localhost:7658",
        "127\\.0\\.0\\.1:43110", "localhost:43110",
        "127\\.0\\.0\\.1:8888", "localhost:8888"
      ]'
      # link black list
      LINK_BLACK_LIST: '[ "(.*\\.)?facebookcorewwwi\\.onion", "(.*\\.)?
↪nytimes3xbfgragh\\.onion" ]'
      # link fallback flag
      LINK_FALLBACK: 1
      # content type white list
      MIME_WHITE_LIST: '[ "text/html", "application/xhtml+xml" ]'
      # content type black list
      MIME_BLACK_LIST: '[ "text/css", "application/javascript", "text/json" ]'
      # content type fallback flag
      MIME_FALLBACK: 0
      # proxy type white list
      PROXY_WHITE_LIST: '[ "tor", "i2p", "freenet", "zeronet" ]'
      # proxy type black list
      PROXY_BLACK_LIST: '[ "null", "data" ]'
      # proxy type fallback flag
      PROXY_FALLBACK: 0
      # API retry times
      API_RETRY: 10
      # API URLs
      #API_NEW_HOST: 'https://example.com/api/new_host'
      #API_REQUESTS: 'https://example.com/api/requests'
      #API_SELENIUM: 'https://example.com/api/selenium'
    restart: "always"
    networks: &networks
      - darc
    volumes: &volumes
      - ./text:/app/text
      - ./extra:/app/extra
      - /data/darc:/app/data

  loader:
    image: jsnbzh/darc:latest
    build: *build
    container_name: loader
    #entrypoint: [ "bash", "/app/run.sh" ]
    command: [ "--type", "crawler" ]
    depends_on: *depends_on
    environment:
      ## [PYTHON] force the stdout and stderr streams to be unbuffered
```

(continues on next page)

(continued from previous page)

```
      PYTHONUNBUFFERED: 1
      # reboot mode
      DARC_REBOOT: 1
      # debug mode
      DARC_DEBUG: 0
      # verbose mode
      DARC_VERBOSE: 1
      # force mode (ignore robots.txt)
      DARC_FORCE: 1
      # check mode (check proxy and hostname before crawling)
      DARC_CHECK: 1
      # check mode (check content type before crawling)
      DARC_CHECK_CONTENT_TYPE: 0
      # save mode
      DARC_SAVE: 0
      # save mode (for requests)
      DAVE_SAVE_REQUESTS: 0
      # save mode (for selenium)
      DAVE_SAVE_SELENIUM: 0
      # processes
      DARC_CPU: 1
      # multiprocessing
      DARC_MULTIPROCESSING: 0
      # multithreading
      DARC_MULTITHREADING: 0
      # time lapse
      DARC_WAIT: 60
      # data storage
      PATH_DATA: "data"
      # Redis URL
      REDIS_URL: 'redis://
↪:UCf7y123aHgaYeGnvLRasALjFfDVHGCz6KiR5Z0WC0DL4ExvSGw5SkcOxBywc0qtZBHVrSVx2QMGewXNP6qVow@redis
↪'
      # max pool
      DARC_MAX_POOL: 1_000
      # Tor proxy & control port
      TOR_PORT: 9050
      TOR_CTRL: 9051
      # Tor management method
      TOR_STEM: 1
      # Tor authentication
      TOR_PASS: "16:B9D36206B5374B3F609045F9609EE670F17047D88FF713EFB9157EA39F"
      # Tor bootstrap retry
      TOR_RETRY: 10
      # Tor bootstrap wait
      TOR_WAIT: 90
      # Tor bootstrap config
      TOR_CFG: "{}"
      # I2P port
      I2P_PORT: 4444
      # I2P bootstrap retry
      I2P_RETRY: 10
      # I2P bootstrap wait
      I2P_WAIT: 90
      # I2P bootstrap config
      I2P_ARGS: ""
      # ZeroNet port
```

(continues on next page)

(continued from previous page)

```
      ZERONET_PORT: 43110
      # ZeroNet bootstrap retry
      ZERONET_RETRY: 10
      # ZeroNet project path
      ZERONET_PATH: "/usr/local/src/zeronet"
      # ZeroNet bootstrap wait
      ZERONET_WAIT: 90
      # ZeroNet bootstrap config
      ZERONET_ARGS: ""
      # Freenet port
      FREENET_PORT: 8888
      # Freenet bootstrap retry
      FREENET_RETRY: 0
      # Freenet project path
      FREENET_PATH: "/usr/local/src/freenet"
      # Freenet bootstrap wait
      FREENET_WAIT: 90
      # Freenet bootstrap config
      FREENET_ARGS: ""
      # time delta for caches in seconds
      TIME_CACHE: 2_592_000  # 30 days
      # time to wait for selenium
      SE_WAIT: 5
      # extract link pattern
      LINK_WHITE_LIST: '[
        ".*?\\.onion",
        ".*?\\.i2p", "127\\.0\\.0\\.1:7657", "localhost:7657", "127\\.0\\.0\\.1:7658",
↪ "localhost:7658",
        "127\\.0\\.0\\.1:43110", "localhost:43110",
        "127\\.0\\.0\\.1:8888", "localhost:8888"
      ]'
      # link black list
      LINK_BLACK_LIST: '[ "(.*\\.)?facebookcorewwwi\\.onion", "(.*\\.)?
↪nytimes3xbfgragh\\.onion" ]'
      # link fallback flag
      LINK_FALLBACK: 1
      # content type white list
      MIME_WHITE_LIST: '[ "text/html", "application/xhtml+xml" ]'
      # content type black list
      MIME_BLACK_LIST: '[ "text/css", "application/javascript", "text/json" ]'
      # content type fallback flag
      MIME_FALLBACK: 0
      # proxy type white list
      PROXY_WHITE_LIST: '[ "tor", "i2p", "freenet", "zeronet" ]'
      # proxy type black list
      PROXY_BLACK_LIST: '[ "null", "data" ]'
      # proxy type fallback flag
      PROXY_FALLBACK: 0
      # API retry times
      API_RETRY: 10
      # API URLs
      #API_NEW_HOST: 'https://example.com/api/new_host'
      #API_REQUESTS: 'https://example.com/api/requests'
      #API_SELENIUM: 'https://example.com/api/selenium'
    restart: "always"
    networks: *networks
    volumes: *volumes
```

(continues on next page)

(continued from previous page)

```
  redis:
    image: redis:darc
    build:
      context: .
      dockerfile: extra/redis.dockerfile
    container_name: redis
    expose:
      - 6379
    restart: "always"
    networks: *networks
    volumes:
      - /data/darc/redis:/data

# network settings
networks:
  darc:
    driver: bridge
```

Note: Should you wish to run *darc* in reboot mode, i.e. set *DARC_REBOOT* and/or *REBOOT* as `True`, you may wish to change the entrypoint to

```
bash /app/run.sh
```

where `run.sh` is a shell script wraps around *darc* especially for reboot mode.

```
#!/usr/bin/env bash

set -e

# time lapse
WAIT=${DARC_WAIT=10}

# signal handlers
trap '[ -f ${PATH_DATA}/darc.pid ] && kill -2 $(cat ${PATH_DATA}/darc.pid)' SIGINT␣
↪SIGTERM SIGKILL

# initialise
echo "+ Starting application..."
python3 -m darc $@
sleep ${WAIT}

# mainloop
while true; do
    echo "+ Restarting application..."
    python3 -m darc
    sleep ${WAIT}
done
```

In such scenario, you can customise your `run.sh` to, for instance, archive then upload current data crawled by *darc* to somewhere else and save up some disk space.

Or you may wish to look into the `_queue_requests.txt` and `_queue_selenium.txt` databases (c.f. *darc.db*), and make some minor adjustments to, perhaps, narrow down the crawling targets.

CHAPTER

FOUR

AUXILIARY SCRIPTS

Since the `darc` project can be deployed through *Docker Integration*, we provided some auxiliary scripts to help with the deployment.

4.1 Health Check

File location `extra/healthcheck.py`

```
usage: healthcheck [-h] [-f FILE] [-i INTERVAL]

health check running container

optional arguments:
  -h, --help            show this help message and exit
  -f FILE, --file FILE  path to compose file
  -i INTERVAL, --interval INTERVAL
                        interval (in seconds) of health check
```

This script will watch the running status of containers managed by Docker Compose. If the containers are stopped or of *unhealthy* status, it will bring the containers back alive.

Also, as the internal program may halt unexpectedly whilst the container remains *healthy*, the script will watch if the program is still active through its output messages. If inactive, the script will restart the containers.

4.2 Upload API Submission Files

File location

- Entry point: `extra/upload.py`
- Helper script: `extra/upload.sh`

```
usage: upload [-h] [-f FILE] [-p PATH] [-i INTERVAL] -H HOST [-U USER]

upload API submission files

optional arguments:
  -h, --help            show this help message and exit
  -f FILE, --file FILE  path to compose file
  -p PATH, --path PATH  path to data storage
  -i INTERVAL, --interval INTERVAL
```

(continues on next page)

(continued from previous page)

```
                        interval (in seconds) to upload
  -H HOST, --host HOST  upstream hostname
  -U USER, --user USER  upstream user credential
```

This script will automatically upload API submission files, c.f. `darc.submit`, using *curl(1)*. The `--user` option is supplied for the same option of *curl(1)*.

When uploading, the script will *pause* the running containers and it will *unpause* them upon completion.

4.3 Remove Repeated Lines

File location `extra/uniq.py`

This script works the same as *uniq(1)*, except it filters one input line at a time without putting pressure onto memory utilisation.

darc is designed as a swiss army knife for darkweb crawling. It integrates `requests` to collect HTTP request and response information, such as cookies, header fields, etc. It also bundles `selenium` to provide a fully rendered web page and screenshot of such view.

The general process of *darc* can be described as following:

0. `process()`: obtain URLs from the `requests` link database (c.f. *load_requests()*), and feed such URLs to `crawler()` with *multiprocessing* support.

1. `crawler()`: parse the URL using *parse_link()*, and check if need to crawl the URL (c.f. *PROXY_WHITE_LIST*, *PROXY_BLACK_LIST* , *LINK_WHITE_LIST* and *LINK_BLACK_LIST*); if true, then crawl the URL with `requests`.

   If the URL is from a brand new host, *darc* will first try to fetch and save `robots.txt` and sitemaps of the host (c.f. `save_robots()` and `save_sitemap()`), and extract then save the links from sitemaps (c.f. `read_sitemap()`) into link database for future crawling (c.f. *save_requests()*). Also, if the submission API is provided, `submit_new_host()` will be called and submit the documents just fetched.

   If `robots.txt` presented, and *FORCE* is `False`, *darc* will check if allowed to crawl the URL.

   Note: The root path (e.g. `/` in https://www.example.com/) will always be crawled ignoring `robots.txt`.

   At this point, *darc* will call the customised hook function from `darc.sites` to crawl and get the final response object. *darc* will save the session cookies and header information, using `save_headers()`.

   Note: If `requests.exceptions.InvalidSchema` is raised, the link will be saved by `save_invalid()`. Further processing is dropped.

   If the content type of response document is not ignored (c.f. *MIME_WHITE_LIST* and *MIME_BLACK_LIST*), *darc* will save the document using `save_html()` or `save_file()` accordingly. And if the submission API is provided, `submit_requests()` will be called and submit the document just fetched.

   If the response document is HTML (`text/html` and `application/xhtml+xml`), *extract_links()* will be called then to extract all possible links from the HTML document and save such links into the database (c.f. *save_requests()*).

And if the response status code is between `400` and `600`, the URL will be saved back to the link database (c.f. *save_requests()*). If **NOT**, the URL will be saved into `selenium` link database to proceed next steps (c.f. *save_selenium()*).

2. `process()`: in the meanwhile, *darc* will obtain URLs from the `selenium` link database (c.f. *load_selenium()*), and feed such URLs to `loader()`.

   Note: If *FLAG_MP* is `True`, the function will be called with *multiprocessing* support; if *FLAG_TH* if `True`, the function will be called with *multithreading* support; if none, the function will be called in single-threading.

3. `loader()`: parse the URL using *parse_link()* and start loading the URL using `selenium` with Google Chrome.

   At this point, *darc* will call the customised hook function from `darc.sites` to load and return the original `selenium.webdriver.Chrome` object.

   If successful, the rendered source HTML document will be saved using `save_html()`, and a full-page screenshot will be taken and saved.

   If the submission API is provided, `submit_selenium()` will be called and submit the document just loaded.

   Later, *extract_links()* will be called then to extract all possible links from the HTML document and save such links into the `requests` database (c.f. *save_requests()*).

CHAPTER

FIVE

INSTALLATION

Note: *darc* supports Python all versions above and includes **3.6**. Currently, it only supports and is tested on Linux (*Ubuntu 18.04*) and macOS (*Catalina*).

When installing in Python versions below **3.8**, *darc* will use `walrus` to compile itself for backport compatibility.

```
pip install darc
```

Please make sure you have Google Chrome and corresponding version of Chrome Driver installed on your system.

Important: Starting from version **0.3.0**, we introduced Redis for the task queue database backend. Please make sure you have it installed, configured, and running when using the `darc` project.

However, the *darc* project is shipped with Docker and Compose support. Please see *Docker Integration* for more information.

Or, you may refer to and/or install from the Docker Hub repository:

```
docker pull jsnbzh/darc[:TAGNAME]
```

CHAPTER

SIX

USAGE

The *darc* project provides a simple CLI:

```
usage: darc [-h] [-f FILE] ...

the darkweb crawling swiss army knife

positional arguments:
  link                  links to craw

optional arguments:
  -h, --help            show this help message and exit
  -f FILE, --file FILE  read links from file
```

It can also be called through module entrypoint:

```
python -m python-darc ...
```

Note: The link files can contain **comment** lines, which should start with `#`. Empty lines and comment lines will be ignored when loading.

CHAPTER

SEVEN

CONFIGURATION

Though simple CLI, the *darc* project is more configurable by environment variables.

7.1 General Configurations

DARC_REBOOT

Type bool (int)

Default 0

If exit the program after first round, i.e. crawled all links from the requests link database and loaded all links from the selenium link database.

This can be useful especially when the capacity is limited and you wish to save some space before continuing next round. See *Docker integration* for more information.

DARC_DEBUG

Type bool (int)

Default 0

If run the program in debugging mode.

DARC_VERBOSE

Type bool (int)

Default 0

If run the program in verbose mode. If DARC_DEBUG is True, then the verbose mode will be always enabled.

DARC_FORCE

Type bool (int)

Default 0

If ignore robots.txt rules when crawling (c.f. crawler()).

DARC_CHECK

Type bool (int)

Default 0

If check proxy and hostname before crawling (when calling *extract_links()*, read_sitemap() and *read_hosts()*).

If DARC_CHECK_CONTENT_TYPE is True, then this environment variable will be always set as True.

DARC_CHECK_CONTENT_TYPE

Type bool (int)

Default 0

If check content type through HEAD requests before crawling (when calling *extract_links()*, read_sitemap() and *read_hosts()*).

DARC_CPU

Type int

Default None

Number of concurrent processes. If not provided, then the number of system CPUs will be used.

DARC_MULTIPROCESSING

Type bool (int)

Default 1

If enable *multiprocessing* support.

DARC_MULTITHREADING

Type bool (int)

Default 0

If enable *multithreading* support.

Note: DARC_MULTIPROCESSING and DARC_MULTITHREADING can **NOT** be toggled at the same time.

DARC_USER

Type str

Default current login user (c.f. getpass.getuser())

Non-root user for proxies.

DARC_MAX_POOL

Type int

Default 1_000

Maximum number of links loaded from the database.

Note: If is an infinit inf, no limit will be applied.

See also:

- *darc.db.load_requests()*
- *darc.db.load_selenium()*

7.2 Data Storage

REDIS_URL

Type `str` (url)

Default `redis://127.0.0.1`

URL to the Redis database.

PATH_DATA

Type `str` (path)

Default `data`

Path to data storage.

See also:

See `darc.save` for more information about source saving.

7.3 Web Crawlers

DARC_WAIT

Type `float`

Default `60`

Time interval between each round when the `requests` and/or `selenium` database are empty.

DARC_SAVE

Type `bool` (`int`)

Default `0`

If save processed link back to database.

Note: If *DARC_SAVE* is `True`, then *DARC_SAVE_REQUESTS* and *DARC_SAVE_SELENIUM* will be forced to be `True`.

See also:

See *darc.db* for more information about link database.

DARC_SAVE_REQUESTS

Type `bool` (`int`)

Default `0`

If save `crawler()` crawled link back to `requests` database.

See also:

See *darc.db* for more information about link database.

DARC_SAVE_SELENIUM

Type `bool` (`int`)

Default `0`

If save `loader()` crawled link back to `selenium` database.

See also:

See *darc.db* for more information about link database.

TIME_CACHE

Type `float`

Default `60`

Time delta for caches in seconds.

The *darc* project supports *caching* for fetched files. `TIME_CACHE` will specify for how log the fetched files will be cached and **NOT** fetched again.

Note: If `TIME_CACHE` is `None` then caching will be marked as *forever*.

SE_WAIT

Type `float`

Default `60`

Time to wait for `selenium` to finish loading pages.

Note: Internally, `selenium` will wait for the browser to finish loading the pages before return (i.e. the web API event `DOMContentLoaded`). However, some extra scripts may take more time running after the event.

7.4 White / Black Lists

LINK_WHITE_LIST

Type `List[str]` (JSON)

Default `[]`

White list of hostnames should be crawled.

Note: Regular expressions are supported.

LINK_BLACK_LIST

Type `List[str]` (JSON)

Default `[]`

Black list of hostnames should be crawled.

Note: Regular expressions are supported.

LINK_FALLBACK

Type `bool` (`int`)

Default `0`

Fallback value for *match_host()*.

MIME_WHITE_LIST

Type List[str] (JSON)

Default []

White list of content types should be crawled.

Note: Regular expressions are supported.

MIME_BLACK_LIST

Type List[str] (JSON)

Default []

Black list of content types should be crawled.

Note: Regular expressions are supported.

MIME_FALLBACK

Type bool (int)

Default 0

Fallback value for *match_mime()*.

PROXY_WHITE_LIST

Type List[str] (JSON)

Default []

White list of proxy types should be crawled.

Note: The proxy types are **case insensitive**.

PROXY_BLACK_LIST

Type List[str] (JSON)

Default []

Black list of proxy types should be crawled.

Note: The proxy types are **case insensitive**.

PROXY_FALLBACK

Type bool (int)

Default 0

Fallback value for *match_proxy()*.

Note: If provided, `LINK_WHITE_LIST`, `LINK_BLACK_LIST`, `MIME_WHITE_LIST`, `MIME_BLACK_LIST`, `PROXY_WHITE_LIST` and `PROXY_BLACK_LIST` should all be JSON encoded strings.

7.5 Data Submission

API_RETRY

Type `int`

Default `3`

Retry times for API submission when failure.

API_NEW_HOST

Type `str`

Default `None`

API URL for `submit_new_host()`.

API_REQUESTS

Type `str`

Default `None`

API URL for `submit_requests()`.

API_SELENIUM

Type `str`

Default `None`

API URL for `submit_selenium()`.

Note: If `API_NEW_HOST`, `API_REQUESTS` and `API_SELENIUM` is `None`, the corresponding submit function will save the JSON data in the path specified by `PATH_DATA`.

7.6 Tor Proxy Configuration

TOR_PORT

Type `int`

Default `9050`

Port for Tor proxy connection.

TOR_CTRL

Type `int`

Default `9051`

Port for Tor controller connection.

TOR_STEM

Type bool (int)

Default 1

If manage the Tor proxy through stem.

TOR_PASS

Type str

Default None

Tor controller authentication token.

Note: If not provided, it will be requested at runtime.

TOR_RETRY

Type int

Default 3

Retry times for Tor bootstrap when failure.

TOR_WAIT

Type float

Default 90

Time after which the attempt to start Tor is aborted.

Note: If not provided, there will be **NO** timeouts.

TOR_CFG

Type Dict[str, Any] (JSON)

Default {}

Tor bootstrap configuration for stem.process.launch_tor_with_config().

Note: If provided, it should be a JSON encoded string.

7.7 I2P Proxy Configuration

I2P_PORT

Type int

Default 4444

Port for I2P proxy connection.

I2P_RETRY

Type int

Default 3

Retry times for I2P bootstrap when failure.

I2P_WAIT

Type `float`

Default `90`

Time after which the attempt to start I2P is aborted.

Note: If not provided, there will be **NO** timeouts.

I2P_ARGS

Type `str` (Shell)

Default `''`

I2P bootstrap arguments for `i2prouter start`.

If provided, it should be parsed as command line arguments (c.f. `shlex.split`).

Note: The command will be run as `DARC_USER`, if current user (c.f. `getpass.getuser()`) is *root*.

7.8 ZeroNet Proxy Configuration

ZERONET_PORT

Type `int`

Default `4444`

Port for ZeroNet proxy connection.

ZERONET_RETRY

Type `int`

Default `3`

Retry times for ZeroNet bootstrap when failure.

ZERONET_WAIT

Type `float`

Default `90`

Time after which the attempt to start ZeroNet is aborted.

Note: If not provided, there will be **NO** timeouts.

ZERONET_PATH

Type `str` (path)

Default `/usr/local/src/zeronet`

Path to the ZeroNet project.

ZERONET_ARGS

Type `str` (Shell)

Default `''`

ZeroNet bootstrap arguments for `ZeroNet.sh main`.

Note: If provided, it should be parsed as command line arguments (c.f. `shlex.split`).

7.9 Freenet Proxy Configuration

FREENET_PORT

Type `int`

Default `8888`

Port for Freenet proxy connection.

FREENET_RETRY

Type `int`

Default `3`

Retry times for Freenet bootstrap when failure.

FREENET_WAIT

Type `float`

Default `90`

Time after which the attempt to start Freenet is aborted.

Note: If not provided, there will be **NO** timeouts.

FREENET_PATH

Type `str` (path)

Default `/usr/local/src/freenet`

Path to the Freenet project.

FREENET_ARGS

Type `str` (Shell)

Default `''`

Freenet bootstrap arguments for `run.sh start`.

If provided, it should be parsed as command line arguments (c.f. `shlex.split`).

Note: The command will be run as `DARC_USER`, if current user (c.f. `getpass.getuser()`) is *root*.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

d

darc, 1
darc.db, 7
darc.error, 24
darc.link, 1
darc.parse, 4
darc.proxy.i2p, 11

INDEX

Symbols

__hash__() (*darc.link.Link method*), 1
_check() (*in module darc.parse*), 4
_check_ng() (*in module darc.parse*), 4
_i2p_bootstrap() (*in module darc.proxy.i2p*), 12

A

APIRequestFailed, 25

B

base (*darc.link.Link attribute*), 1

C

check_robots() (*in module darc.parse*), 5

D

darc
    module, 1
darc.const.CHECK (*built-in variable*), 20
darc.const.CHECK_NG (*built-in variable*), 20
darc.const.CWD (*built-in variable*), 20
darc.const.DARC_CPU (*built-in variable*), 20
darc.const.DARC_USER (*built-in variable*), 21
darc.const.DARC_WAIT (*built-in variable*), 22
darc.const.DEBUG (*built-in variable*), 20
darc.const.FLAG_MP (*built-in variable*), 20
darc.const.FLAG_TH (*built-in variable*), 20
darc.const.FORCE (*built-in variable*), 20
darc.const.LINK_BLACK_LIST (*built-in variable*), 23
darc.const.LINK_FALLBACK (*built-in variable*), 23
darc.const.LINK_WHITE_LIST (*built-in variable*), 23
darc.const.MIME_BLACK_LIST (*built-in variable*), 24
darc.const.MIME_FALLBACK (*built-in variable*), 24
darc.const.MIME_WHITE_LIST (*built-in variable*), 24
darc.const.PATH_DB (*built-in variable*), 21
darc.const.PATH_ID (*built-in variable*), 22
darc.const.PATH_LN (*built-in variable*), 21
darc.const.PATH_MISC (*built-in variable*), 21
darc.const.PATH_QR (*built-in variable*), 21
darc.const.PATH_QS (*built-in variable*), 21
darc.const.PROXY_BLACK_LIST (*built-in variable*), 24
darc.const.PROXY_FALLBACK (*built-in variable*), 24
darc.const.PROXY_WHITE_LIST (*built-in variable*), 24
darc.const.REBOOT (*built-in variable*), 19
darc.const.REDIS (*built-in variable*), 21
darc.const.ROOT (*built-in variable*), 20
darc.const.SAVE (*built-in variable*), 22
darc.const.SAVE_REQUESTS (*built-in variable*), 22
darc.const.SAVE_SELENIUM (*built-in variable*), 22
darc.const.SE_EMPTY (*built-in variable*), 23
darc.const.SE_WAIT (*built-in variable*), 23
darc.const.TIME_CACHE (*built-in variable*), 22
darc.const.VERBOSE (*built-in variable*), 20
darc.db
    module, 7
darc.db.MAX_POOL (*in module darc.db*), 9
darc.db.QR_LOCK (*in module darc.db*), 9
darc.db.QS_LOCK (*in module darc.db*), 9
darc.error
    module, 24
darc.link
    module, 1
darc.parse
    module, 4
darc.proxy.bitcoin.LOCK (*built-in variable*), 10
darc.proxy.bitcoin.PATH (*built-in variable*), 10
darc.proxy.data.PATH (*built-in variable*), 10
darc.proxy.ed2k.LOCK (*built-in variable*), 10
darc.proxy.ed2k.PATH (*built-in variable*), 10
darc.proxy.freenet._FREENET_ARGS (*built-in variable*), 11
darc.proxy.freenet._FREENET_BS_FLAG (*built-in variable*), 11

darc.proxy.freenet._FREENET_PROC (*built-in variable*), 11
darc.proxy.freenet.BS_WAIT (*built-in variable*), 11
darc.proxy.freenet.FREENET_ARGS (*built-in variable*), 11
darc.proxy.freenet.FREENET_PATH (*built-in variable*), 11
darc.proxy.freenet.FREENET_PORT (*built-in variable*), 10
darc.proxy.freenet.FREENET_RETRY (*built-in variable*), 11
darc.proxy.i2p
    module, 11
darc.proxy.i2p._I2P_ARGS (*in module darc.proxy.i2p*), 14
darc.proxy.i2p._I2P_BS_FLAG (*in module darc.proxy.i2p*), 14
darc.proxy.i2p._I2P_PROC (*in module darc.proxy.i2p*), 14
darc.proxy.i2p.BS_WAIT (*in module darc.proxy.i2p*), 14
darc.proxy.i2p.I2P_ARGS (*in module darc.proxy.i2p*), 14
darc.proxy.i2p.I2P_PORT (*in module darc.proxy.i2p*), 14
darc.proxy.i2p.I2P_REQUESTS_PROXY (*in module darc.proxy.i2p*), 13
darc.proxy.i2p.I2P_RETRY (*in module darc.proxy.i2p*), 14
darc.proxy.i2p.I2P_SELENIUM_PROXY (*in module darc.proxy.i2p*), 13
darc.proxy.irc.LOCK (*built-in variable*), 14
darc.proxy.irc.PATH (*built-in variable*), 14
darc.proxy.LINK_MAP (*built-in variable*), 17
darc.proxy.magnet.LOCK (*built-in variable*), 15
darc.proxy.magnet.PATH (*built-in variable*), 14
darc.proxy.mail.LOCK (*built-in variable*), 15
darc.proxy.mail.PATH (*built-in variable*), 15
darc.proxy.null.LOCK (*built-in variable*), 15
darc.proxy.null.PATH (*built-in variable*), 15
darc.proxy.tor._TOR_BS_FLAG (*built-in variable*), 16
darc.proxy.tor._TOR_CONFIG (*built-in variable*), 16
darc.proxy.tor._TOR_CTRL (*built-in variable*), 16
darc.proxy.tor._TOR_PROC (*built-in variable*), 16
darc.proxy.tor.BS_WAIT (*built-in variable*), 16
darc.proxy.tor.TOR_CFG (*built-in variable*), 16
darc.proxy.tor.TOR_CTRL (*built-in variable*), 15
darc.proxy.tor.TOR_PASS (*built-in variable*), 15
darc.proxy.tor.TOR_PORT (*built-in variable*), 15
darc.proxy.tor.TOR_REQUESTS_PROXY (*built-in variable*), 15
darc.proxy.tor.TOR_RETRY (*built-in variable*), 16
darc.proxy.tor.TOR_SELENIUM_PROXY (*built-in variable*), 15
darc.proxy.tor.TOR_STEM (*built-in variable*), 15
darc.proxy.zeronet._ZERONET_ARGS (*built-in variable*), 17
darc.proxy.zeronet._ZERONET_BS_FLAG (*built-in variable*), 17
darc.proxy.zeronet._ZERONET_PROC (*built-in variable*), 17
darc.proxy.zeronet.BS_WAIT (*built-in variable*), 17
darc.proxy.zeronet.ZERONET_ARGS (*built-in variable*), 17
darc.proxy.zeronet.ZERONET_PATH (*built-in variable*), 17
darc.proxy.zeronet.ZERONET_PORT (*built-in variable*), 16
darc.proxy.zeronet.ZERONET_RETRY (*built-in variable*), 16
darc.save._SAVE_LOCK (*built-in variable*), 6
darc.submit.API_NEW_HOST (*built-in variable*), 9
darc.submit.API_REQUESTS (*built-in variable*), 10
darc.submit.API_RETRY (*built-in variable*), 9
darc.submit.API_SELENIUM (*built-in variable*), 10
darc.submit.PATH_API (*built-in variable*), 9
DARC_CHECK, 20
DARC_CHECK_CONTENT_TYPE, 20
DARC_CPU, 20
DARC_DEBUG, 20
DARC_FORCE, 20
DARC_MAX_POOL, 9
DARC_MULTIPROCESSING, 20
DARC_MULTITHREADING, 20
DARC_REBOOT, 20, 42
DARC_SAVE, 22, 53
DARC_SAVE_REQUESTS, 22, 53
DARC_SAVE_SELENIUM, 22, 53
DARC_USER, 21
DARC_VERBOSE, 20
DARC_WAIT, 22
drop_hostname() (*in module darc.db*), 7
drop_requests() (*in module darc.db*), 7
drop_selenium() (*in module darc.db*), 7

E

environment variable
    API_NEW_HOST, 56
    API_REQUESTS, 56

API_RETRY, 56
API_SELENIUM, 56
DARC_CHECK, 20, 51
DARC_CHECK_CONTENT_TYPE, 20, 51
DARC_CPU, 20, 52
DARC_DEBUG, 20, 51
DARC_FORCE, 20, 51
DARC_MAX_POOL, 9, 52
DARC_MULTIPROCESSING, 20, 52
DARC_MULTITHREADING, 20, 52
DARC_REBOOT, 20, 42, 51
DARC_SAVE, 22, 53
DARC_SAVE_REQUESTS, 22, 53
DARC_SAVE_SELENIUM, 22, 53
DARC_USER, 21, 52
DARC_VERBOSE, 20, 51
DARC_WAIT, 22, 53
FREENET_ARGS, 59
FREENET_PATH, 59
FREENET_PORT, 59
FREENET_RETRY, 59
FREENET_WAIT, 59
I2P_ARGS, 58
I2P_PORT, 57
I2P_RETRY, 57
I2P_WAIT, 58
LINK_BLACK_LIST, 23, 54
LINK_FALLBACK, 24, 54
LINK_WHITE_LIST, 23, 54
MIME_BLACK_LIST, 24, 55
MIME_FALLBACK, 24, 55
MIME_WHITE_LIST, 24, 55
PATH_DATA, 21, 53
PROXY_BLACK_LIST, 24, 55
PROXY_FALLBACK, 24, 55
PROXY_WHITE_LIST, 24, 55
REDIS_URL, 21, 53
SE_WAIT, 23, 54
TIME_CACHE, 23, 54
TOR_CFG, 57
TOR_CTRL, 56
TOR_PASS, 57
TOR_PORT, 56
TOR_RETRY, 57
TOR_STEM, 56
TOR_WAIT, 57
ZERONET_ARGS, 58
ZERONET_PATH, 58
ZERONET_PORT, 58
ZERONET_RETRY, 58
ZERONET_WAIT, 58
extract_links() (*in module darc.parse*), 5

F

fetch_hosts() (*in module darc.proxy.i2p*), 12
FreenetBootstrapFailed, 25

G

get_content_type() (*in module darc.parse*), 5
get_hosts() (*in module darc.proxy.i2p*), 12
get_lock() (*in module darc.db*), 7

H

have_hostname() (*in module darc.db*), 8
have_hosts() (*in module darc.proxy.i2p*), 12
host (*darc.link.Link attribute*), 1

I

i2p_bootstrap() (*in module darc.proxy.i2p*), 12
I2PBootstrapFailed, 25

L

Link (*class in darc.link*), 1
LINK_BLACK_LIST, 23
LINK_FALLBACK, 24
LINK_WHITE_LIST, 23
LinkNoReturn, 25
load_requests() (*in module darc.db*), 8
load_selenium() (*in module darc.db*), 8
LockWarning, 25

M

match_host() (*in module darc.parse*), 5
match_mime() (*in module darc.parse*), 6
match_proxy() (*in module darc.parse*), 6
MIME_BLACK_LIST, 24
MIME_FALLBACK, 24
MIME_WHITE_LIST, 24
module
    darc, 1
    darc.db, 7
    darc.error, 24
    darc.link, 1
    darc.parse, 4
    darc.proxy.i2p, 11

N

name (*darc.link.Link attribute*), 1

P

parse_link() (*in module darc.link*), 2
PATH_DATA, 21
proxy (*darc.link.Link attribute*), 2
PROXY_BLACK_LIST, 24
PROXY_FALLBACK, 24
PROXY_WHITE_LIST, 24

Q

quote() (*in module darc.link*), 3

R

read_hosts() (*in module darc.proxy.i2p*), 13
REDIS_URL, 21
RedisConnectionFailed, 25
render_error() (*in module darc.error*), 26

S

save_hosts() (*in module darc.proxy.i2p*), 13
save_requests() (*in module darc.db*), 8
save_selenium() (*in module darc.db*), 8
SE_WAIT, 23
SiteNotFoundWarning, 25

T

TIME_CACHE, 23
TorBootstrapFailed, 25
TorRenewFailed, 26

U

unquote() (*in module darc.link*), 3
UnsupportedLink, 26
UnsupportedPlatform, 26
UnsupportedProxy, 26
url (*darc.link.Link attribute*), 2
url_parse (*darc.link.Link attribute*), 2
urljoin() (*in module darc.link*), 3
urlparse() (*in module darc.link*), 4

Z

ZeroNetBootstrapFailed, 26