Heritrix3 Client

Release 0.4.1

Erik Körner

Jan 19, 2021

CONTENTS

CHAPTER

ONE

OVERVIEW

docs	
tests	
package	

A internetarchive/heritrix3 python REST API client.

- Free software: MIT license

1.1 Installation

```
pip install heritrix3
```

You can also install the in-development version with:

```
pip install https://github.com/Querela/python-heritrix3-client/archive/master.zip
```

1.2 Documentation

https://python-heritrix3-client.readthedocs.io/

1.3 Development

To run all the tests run:

```
tox
```

Note, to combine the coverage data from all the tox environments run:

Windows	`set PYTEST_ADDOPTS=--cov-append` `tox`
Other	`PYTEST_ADDOPTS=--cov-append tox`

CHAPTER

TWO

INSTALLATION

At the command line:

```
pip install heritrix3
```

CHAPTER

THREE

USAGE

To use Heritrix3 Client in a project:

```
import heritrix3
```

A basic workflow follows:

```
# basic imports
from pathlib import Path
from pprint import pprint
from heritrix3 import disable_ssl_warnings
from heritrix3 import HeritrixAPI

# disable insecure requests warning
disable_ssl_warnings()

# create the REST API client
api = HeritrixAPI(host="https://localhost:8443/engine", user="admin", passwd="admin",␣
↪verbose=True)

# dump info
pprint(api.info(raw=False))
# similar to info (output wise)
pprint(api.rescan(raw=False))
# alternative also
pprint(HeritrixAPI._xml2json(api.rescan(raw=True).text))
```

How to work with jobs:

```
# create job (if it exists, it will not do anything?)
jobname = "test"
pprint(api.create(jobname))
assert jobname in api.list_jobs()

# list all jobs
api.list_jobs()
# and their actions
api.get_job_actions(jobname)

# send a config file (that allows a separate seeds.txt file)
p = (Path.cwd() / "..").resolve() / "examples" / "crawler-beans.seed_file.cxml"
api.send_config(jobname, p)

# create + send seeds
p = (Path.cwd() / "..").resolve() / "examples" / "seeds.txt"
```

(continues on next page)

(continued from previous page)

```
p.write_text("https://www.google.com/\n")
api.send_file(jobname, p)  # or with "seeds.txt" as third param

# build job (required for some functions, like script execution)
pprint(api.build(jobname))
# can be used to wait until an action is available
# might block indefinitely if this actions does not exists or won't ever be available
api.wait_for_action(jobname, "launch")

# launch the job
pprint(api.launch(jobname))
# pause a job
pprint(api.pause(jobname))
# checkpoint
pprint(api.checkpoint(jobname))
# unpause a job
pprint(api.unpause(jobname))
# terminate the job
pprint(api.terminate(jobname))

# unbuild/teardown the job
pprint(api.teardown(jobname))

# NOTE: the following requires the job to be built! (so no teardown)
# clean up the job (all files are gone)
# NOTE: you should be careful that the job is not still running
api.delete_job_dir(jobname)
pprint(api.rescan())
assert jobname not in api.list_jobs()
```

See the official Heritrix REST API docs.

Show job information:

```
job_info_dict = api.info(jobname)
job_xml_txt = api.info(jobname, raw=True).text

config_xml_txt = api.get_config(jobname)

# crawl report (plain text)
launchid = None  # "latest"
report_txt = api.crawl_report(jobname, launchid)

# the following functions require the job to be built

# list the jobs files
pprint(api.list_files(jobname))

# show the warcs (after pause/terminate)
pprint(api.list_warcs(jobname))

# launch id
launchid = api.get_launchid(jobname)
```

If you require a basic heritrix setup, you may use the ekoerner/heritrix Docker image.

3.1 CLI

The Heritrix3 client library also provides a commandline utility, named **heritrix3**:

```
heritrix3 --help
# configure your heritrix REST endpoint:
heritrix3 --host https://localhost:8443/engine --username admin --password admin

# interactive python shell
heritrix3 shell

# list jobs, actions
heritrix3 list-jobs
heritrix3 list-jobs-actions

# show info
heritrix3 info
# show job info for "test"
heritrix3 info test
```

CHAPTER

FOUR

REFERENCE

4.1 heritrix3

4.2 heritrix3.api

exception heritrix3.api.**HeritrixAPIError**(*message: str*, **args*, ***kwargs*)
Error as response from Heritrix3 REST API.

Parameters

- **message** (*str*) – Error description / message.
- **response** (*Optional[requests.Response]*) – Optional api response object.

class heritrix3.api.**HeritrixAPI**(*host: str = 'https://localhost:8443/engine'*, *user: str = 'admin'*, *passwd: str = 'admin'*, *verbose: bool = False*, *insecure: bool = True*, *headers: Optional[Dict[str, str]] = None*, *timeout: Optional[Union[int, float]] = None*)

send_file(*job_name: str*, *filepath: os.PathLike*, *name: Optional[str] = None*) → bool

send_content(*job_name: str*, *filecontent: Union[bytes, BinaryIO]*, *name: str*) → bool

retrieve_file(*job_name: str*, *local_filepath: os.PathLike*, *job_filepath: Union[str, os.PathLike]*, *overwrite: bool = False*) → bool

info(*job_name: Optional[str] = None*, *raw: bool = False*) → Union[str, requests.models.Response]

list_jobs(*status: Optional[str] = None*, *unbuilt: bool = False*) → List[str]

get_job_state(*job_name: str*) → Optional[str]

get_crawl_exit_state(*job_name: str*) → Optional[str]

get_job_actions(*job_name: str*) → List[str]

wait_for_action(*job_name: str*, *action: str*, *timeout: Union[int, float] = 20*, *poll_delay: Union[int, float] = 1*) → bool

wait_for_jobstate(*job_name: str*, *state: str*, *timeout: Union[int, float] = 20*, *poll_delay: Union[int, float] = 1*) → bool

create(*job_name: str*, *raw: bool = False*) → Union[str, requests.models.Response]

add(*job_dir: str*, *raw: bool = False*) → Union[str, requests.models.Response]

rescan(*raw: bool = False*) → Union[str, requests.models.Response]

copy(*job_name: str*, *new_job_name: str*, *as_profile: bool = False*, *raw: bool = False*) → Union[str, requests.models.Response]

build (*job_name: str*, *raw: bool = False*) → Union[str, requests.models.Response]

launch (*job_name: str*, *checkpoint: Optional[str] = None*, *raw: bool = False*) → Union[str, requests.models.Response]

pause (*job_name: str*, *raw: bool = False*) → Union[str, requests.models.Response]

unpause (*job_name: str*, *raw: bool = False*) → Union[str, requests.models.Response]

terminate (*job_name: str*, *raw: bool = False*) → Union[str, requests.models.Response]

teardown (*job_name: str*, *raw: bool = False*) → Union[str, requests.models.Response]

checkpoint (*job_name: str*, *raw: bool = False*) → Union[str, requests.models.Response]

execute_script (*job_name: str*, *script: str*, *engine: str = 'beanshell'*, *raw: bool = False*) → Union[str, requests.models.Response]

get_config (*job_name: str*, *raw: bool = True*) → str

send_config (*job_name: str*, *cxml_filepath: os.PathLike*) → bool

get_config_url (*job_name: str*) → str

get_launchid (*job_name: str*) → Optional[str]

crawl_report (*job_name: str*, *launch_id: Optional[str] = None*) → Optional[str]

seeds_report (*job_name: str*, *launch_id: Optional[str] = None*) → Optional[str]

hosts_report (*job_name: str*, *launch_id: Optional[str] = None*) → Optional[str]

mimetypes_report (*job_name: str*, *launch_id: Optional[str] = None*) → Optional[str]

responsecodes_report (*job_name: str*, *launch_id: Optional[str] = None*) → Optional[str]

job_log (*job_name: str*) → Optional[str]

crawl_log (*job_name: str*, *launch_id: Optional[str] = None*) → Optional[str]

list_files (*job_name: str*, *gather_files: bool = True*, *gather_folders: bool = True*) → List[str]

list_warcs (*job_name: str*, *launchid: Optional[str] = None*) → Optional[List[str]]

retrieve_warcs (*job_name: str*, *local_folderpath: os.PathLike*, *launchid: Optional[str] = None*, *warcs_job_filepaths: Optional[List[Union[str, os.PathLike]]] = None*, *overwrite: bool = False*) → Optional[int]

delete_job_dir (*job_name: str*) → None

heritrix3.api.**disable_ssl_warnings**()

Quieten SSL insecure warnings.

See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-warnings

4.3 heritrix3.cli

4.3.1 heritrix3

CLI for the Heritrix API.

```
heritrix3 [OPTIONS] COMMAND [ARGS]...
```

Options

-h, --host <host>
Heritrix base URI

Default https://localhost:8443/engine

-u, --username <username>
HTTP Digest Username

Default admin

-p, --password <password>
HTTP Digest Password

Default admin

--version
Show the version and exit.

info

Show information about all jobs or a single job. If given a jobname as argument then only display information about this job. Tries to use `pygments` to colorize the output.

```
heritrix3 info [OPTIONS] [JOBNAME]
```

Options

--raw
Output plain XML response.

Arguments

JOBNAME
Optional argument

list-jobs

List jobs, allow filtering for unbuilt ones.

```
heritrix3 list-jobs [OPTIONS]
```

Options

--unbuilt

--sorted

list-jobs-actions

List jobs and available heritrix actions.

```
heritrix3 list-jobs-actions [OPTIONS]
```

Options

--sorted

shell

Open an interactive shell for testing.

```
heritrix3 shell [OPTIONS]
```

CHAPTER

FIVE

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

5.1 Bug reports

When reporting a bug please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.2 Documentation improvements

Heritrix3 Client could always use more documentation, whether as part of the official Heritrix3 Client docs, in docstrings, or even on the web in blog posts, articles, and such.

5.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/Querela/python-heritrix3-client/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome :)

5.4 Development

To set up *python-heritrix3-client* for local development:

1. Fork python-heritrix3-client (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:YOURGITHUBNAME/python-heritrix3-client.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

4. When you're done making changes run all the checks and docs builder with tox one command:

```
tox
```

5. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

5.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`).
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

5.4.2 Tips

To run a subset of tests:

```
tox -e envname -- pytest -k test_myfeature
```

To run all the test environments in *parallel*:

```
tox -p auto
```

CHAPTER

SIX

AUTHORS

- Erik Körner - koerner@informatik.uni-leipzig.de

CHAPTER

SEVEN

CHANGELOG

7.1 WIP

- Tests using real Heritrix? (Coverage?)
- Refactoring common code fragments.
- Documentation (docstrings).

7.2 0.4.0 (2021-01-11)

- Reorder *api* functions.
- Add log retrieval methods.
- Add job state check + `wait_for` methods.

7.3 0.3.0 (2021-01-11)

- Move into separate *api* module. Empty `__init__.py`.

7.4 0.2.0 (2021-01-09)

- Typings.
- Add file download (e.g. all WARCs).
- Add report retrieval.

7.5 0.1.0 (2021-01-09)

- First release on PyPI.
- Initial implementation and documentation.

7.6 0.0.0 (2021-01-09)

- Code skeleton using `cookiecutter gh:ionelmc/cookiecutter-pylibrary`

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

h

INDEX

Symbols

A

B

C

D

E

G

H