kwalitee Documentation

Release 1.0.0.dev20160426

CERN

Sep 28, 2016

Contents

Kwalitee is a tool that runs static analysis checks on Git repository.

- Free software: GPLv2 license
- Documentation: https://kwalitee.readthedocs.io/

CHAPTER 1

Introduction

Kwalitee is a tool that runs static analysis checks on invenio and invenio-related repositories. It can be used as a web service using the Github API or as a git hook from the command line.

It aims at slowly, but steadily enforce good practices regarding commit message formatting, code layout (PEP8), documentation (PYDOCSTYLE) and help the integrators doing their job without having to worry about recurrent mistakes.

It relies on and thanks the following softwares and libraries:

- pyflakes,
- PEP8,
- PEP257,

CHAPTER 2

Git Hooks

Install git hooks into your repository using:

```
cd /path/to/git-repo
kwalitee githooks install
```

and uninstall hooks using:

```
kwalitee githooks uninstall
```

Following hooks are installed:

- `pre-commit` - run PEP8, pyflakes and copyright year checks on files being committed. If errors are found, the commit is aborted.
- `prepare-commit-msg` - prepare standard form commit message.
- `post-commit` - check commit message form and signatures. If errors are found, they can be fixed with `git commit --amend`.

All checks can be disabled using:

```
git commit --no-verify
```

CHAPTER 3

Kwalitee checks

- **Static analysis of files:**
  - pyflakes
  - PEP8
  - PEP257
  - Copyright year in license
- **Commit message analysis:**
  - First line less than 50 chars and according to the pattern `<component>: <short description>` (using nouns).
  - Body with detailed description of what this patch does, formatted as a bulletted list. (using present tense).
  - Required signatures: `Signed-off-by` and `Reviewed-by`.

CHAPTER 4

User's Guide

This part of the documentation will show you how to get started in using Kwalitee.

4.1 Versions

We recommend you to use the stable version unless you want to contribute.

4.1.1 Stable version

kwalitee is on PyPI so all you need is:

```
$ pip install --user kwalitee
```

4.1.2 Development version

```
$ git clone https://github.com/inveniosoftware/kwalitee
$ cd kwalitee
$ pip install --user -r requirements.txt
```

4.2 Installation of the command-line interface

Kwalitee can be used as a *Command-line interface*, which has some handy features like the *githooks* and the *Messages* checks.

By default the messages checks will try to use `GitPython` but we are recommending you to install and use pygit2. As, it has not stable version yet, the installation requires some work.

4.2.1 Ubuntu

```
$ sudo apt-add-repository ppa:dennis/python
$ sudo apt-get update
$ sudo apt-get install python-dev libffi-dev libgit2
$ pip install cffi pygit2
```

If you don't find a suitable version using `ppa:dennis/python`, you can always install it manually via `cmake`.

4.2.2 OSX

The important detail here is to use the same version for `libgit2` **and** `pygit2`. Homebrew is a way to get it working.

```
$ brew update
$ brew install libgit2 # currently 0.21.0 (2014-07-28)
$ pip install pygit2
```

4.3 Command-line interface

kwalitee comes with a command-line tool that goes by the name of `kwalitee`. If you've installed it using the `--user` option, you'll have to add `~/.local/bin` to your path. Otherwise, you should be able to call it without any trouble.

```
$ export PATH+=:~/.local/bin
```

4.3.1 Help

The command line tool is able to give you direct little help.

```
$ kwalitee --help
```

4.3.2 `check`

Utility to run various *kwalitee* checks in your repository.

Messages

`message`

Runs the checks on the existing commits.

```
$ kwalitee check message master..
```

4.3.3 `githooks`

This tool can install or uninstall some hooks into the *current* git repository.

See also:

- *kwalitee.cli.githooks*
- *kwalitee.hooks*

Hooks

`pre-commit`

Runs the checks on the files about to be commited.

prepare-commit-msg

Based on the state of the commit create a commit message to be filled in.

See also:

kwalitee.config.COMMIT_MSG_TEMPLATE

post-commit

Verifies that the commit message passes the checks.

Installation

```
$ kwalitee install
```

Uninstallation

```
$ kwalitee uninstall
```

4.3.4 account

Utililty to manage to user account that have registred.

Listing

```
$ kwalitee account list
```

Creation and modification

Creation and modification are using the `add` command. You can alter the user's email and its GitHub API token. Any user with a token will have the comments posted on his repository made using the token's account instead of the default one.

```
$ kwalitee account add <ACCOUNT>

$ kwalitee account add <ACCOUNT> --email <EMAIL> --token <TOKEN>
```

See also:

kwalitee.config.ACCESS_TOKEN

Deletion

Deletion is permanent and it deletes everything belonging to the given account.

```
$ kwalitee account remove <ACCOUNT>
```

4.3.5 repository

Utility to manage to user's repositories.

Listing

```
$ kwalitee repository list
```

Creation

```
$ kwalitee repository add <ACCOUNT>/<REPOSITORY>
```

Deletion

Deletion is permanent and it deletes everything belonging to the given repository.

```
$ kwalitee repository remove <ACCOUNT>/<REPOSITORY>
```

4.4 Testing

Running the tests are as simple as:

```
$ python setup.py test
```

The code coverage can be output by passing some arguments to py.test.

```
$ python setup.py test -a "tests --cov kwalitee --cov-config .coveragerc"
# html report
$ python setup.py test -a "tests --cov kwalitee --cov-report html"
```

Ditto for running only one test or debugging with pdb.

```
$ python setup.py test -a tests/tests_ping.py
$ python setup.py test -a tests/tests_ping.py::test_ping
$ python setup.py test -a "tests --pdb"
```

4.4.1 Writing tests

The tests are using PyHamcrest for its richness and the nice default output provided. To be consistent, avoid using `unittest` or bare `assert`.

4.4.2 Fixtures

Fixtures are provided by the very powerful py.test. Take a look at the fixtures defined in the conftest.py files.

4.4.3 Other tools

HTTPretty

`HTTPretty` is a HTTP client mock library which lets you define your own custom answers and status code as well as testing which calls have been made.

mock

`mock` is used to mock file opening (`open()`) and inspect the content that was written in it without having to create temporary files with `tempfile`.

CHAPTER 5

API Reference

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

5.1 API

5.1.1 Kwalitee package

Command line interface

kwalitee.cli package

Submodules

kwalitee.cli.githooks module

Command-line tools for the git hooks.

Module contents

Command line interfaces entrypoints.

Configuration

kwalitee base configuration.

To change it, put a config.py into your instance.

`kwalitee.config.`**`CONTEXT`**
Context for Github.

See also:

Github API Statuses

`kwalitee.config.`**`COMPONENTS`**
List of supported components.

See also:

kwalitee.check_message()

kwalitee.config.**ACCESS_TOKEN**
Github access token. Used to post statuses and comments. It MUST be set.

kwalitee.config.**AUTO_CREATE**
Allow anyone to add its repository by setting up the webhook on Github.

Default: `False`

kwalitee.config.**CHECK_COMMIT_MESSAGES**
Enable the *commit message checks*.

Default: `True`

kwalitee.config.**CHECK_WIP**
Enable the *work-in-progress* pull requests checks. Disabled by default.

Default: `False`

kwalitee.config.**CHECK_LICENSE**
Enable the *license checks*.

Default: `True`

kwalitee.config.**CHECK_PEP8**
Enable the *PEP8 checks*.

Default: `True`

kwalitee.config.**CHECK_PYDOCSTYLE**
Enable the *PYDOCSTYLE checks*.

Default: `True`

kwalitee.config.**CHECK_PYFLAKES**
Enable the PyFlakes checks. PEP8 checks are required.

Default: `True`

kwalitee.config.**IGNORE**
Error codes to ignore.

Default: `['E123','E226','E24','E501','E265']`

kwalitee.config.**SELECT**
Error codes to specially enable.

Default: `[]`

kwalitee.config.**WORKER_TIMEOUT**
Background worker job time window.

Any job taking longer than that will be killed.

RQ default timeout is 180 seconds

kwalitee.config.**MIN_REVIEWERS**
Minimum number of reviewers for py:func:*message check* <.*kwalitee.check_message*>.

Default: 3

kwalitee.config.**LABEL_WIP**
Label to apply for a *work-in-progress* pull request.

Default: `"in_work"`

kwalitee.config.**LABEL_REVIEW**
Label to apply for a pull request that needs more reviewers.

Default: `"in_review"`

kwalitee.config.**LABEL_READY**
Label to apply for a pull request that passes all the checks.

Default: `"in_integration"`

kwalitee.config.**EXCLUDES**
List of regex of excluded files.

Default: `[]`

kwalitee.config.**AUTHORS**
List of filenames containing authors and contributors.

Default: `['AUTHORS.rst',]`

kwalitee.config.**EXCLUDE_AUTHOR_NAMES**
List of author names to be excluded from checking (e.g. known second email addresses).

Default: `[]`

kwalitee.config.**ALT_SIGNATURES = ('Reported-by',)**
Alternative signatures recognized but not counted as reviewers.

kwalitee.config.**COMMIT_MSG_TEMPLATE = '{component}: description (max 50 chars, using nouns)\n\n* Detailed descri**
Template used to generate the commit message from the git hook.

kwalitee.config.**GITHUB = 'https://github.com/'**
Github base URL.

kwalitee.config.**GITHUB_REPO = 'https://github.com/{account}/{repository}/'**
Github repository URL template.

kwalitee.config.**PYDOCSTYLE_MATCH = '(?!test_).*\\.py'**
Files checked for PYDOCSTYLE conformance.

kwalitee.config.**PYDOCSTYLE_MATCH_DIR = '[^\\.].*'**
Directories checkes for PYDOCSTYLE conformance.

kwalitee.config.**SIGNATURES = ('Signed-off-by', 'Co-authored-by', 'Tested-by', 'Reviewed-by', 'Acked-by')**
Authors and reviewers signatures.

kwalitee.config.**TRUSTED_DEVELOPERS = []**
Super developers who's code never fail.

Hooks

Git hooks.

kwalitee.hooks.**run**(*command*, *raw_output=False*)
Run a command using subprocess.

Parameters

- **command** (*str*) – command line to be run
- **raw_output** (*bool*) – does not attempt to convert the output as unicode

Returns error code, output (`stdout`) and error (`stderr`)

Return type tuple

Checks

Kwalitee checks for PEP8, PYDOCSTYLE, PyFlakes and License.

kwalitee.kwalitee.**SUPPORTED_FILES = ('.py', '.html', '.tpl', '.js', '.jsx', '.css', '.less')**
Supported file types.

kwalitee.kwalitee.**check_author**(*author*, ***kwargs*)
Check the presence of the author in the AUTHORS/THANKS files.

Rules:

- the author full name and email must appear in AUTHORS file

Parameters

- **authors** (*list*) – name of AUTHORS files
- **path** (*str*) – path to the repository home

Returns errors

Return type *list*

kwalitee.kwalitee.**check_file**(*filename*, ***kwargs*)
Perform static analysis on the given file.

See also:

- *SUPPORTED_FILES*
- *check_pep8()*
- *check_pydocstyle()*
- and *check_license()*

Parameters **filename** (*str*) – path of file to check.

Returns errors sorted by line number or None if file is excluded

Return type *list*

kwalitee.kwalitee.**check_license**(*filename*, ***kwargs*)
Perform a license check on the given file.

The license format should be commented using # and live at the top of the file. Also, the year should be the current one.

Parameters

- **filename** (*str*) – path of file to check.
- **year** (*int*) – default current year
- **ignore** (*list*) – codes to ignore, e.g. `('L100','L101')`
- **python_style** (*bool*) – False for JavaScript or CSS files

Returns errors

Return type *list*

kwalitee.kwalitee.**check_message**(*message*, ***kwargs*)
Check the message format.

Rules:

•the first line must start by a component name

•and a short description (52 chars),

•then bullet points are expected

•and finally signatures.

Parameters

- **components** (*list*) – compontents, e.g. `('auth','utils','misc')`
- **signatures** (*list*) – signatures, e.g. `('Signed-off-by','Reviewed-by')`
- **alt_signatures** (*list*) – alternative signatures, e.g. `('Tested-by',)`
- **trusted** (*list*) – optional list of reviewers, e.g. `('john.doe@foo.org',)`
- **max_length** (*int*) – optional maximum line length (by default: 72)
- **max_first_line** (*int*) – optional maximum first line length (by default: 50)
- **allow_empty** (*bool*) – optional way to allow empty message (by default: False)

Returns errors sorted by line number

Return type *list*

kwalitee.kwalitee.**check_pep8**(*filename*, ***kwargs*)
Perform static analysis on the given file.

Parameters

- **filename** (*str*) – path of file to check.
- **ignore** (*list*) – codes to ignore, e.g. `('E111','E123')`
- **select** (*list*) – codes to explicitly select.
- **pyflakes** (*bool*) – run the pyflakes checks too (default `True`)

Returns errors

Return type *list*

See also:

`pycodestyle.Checker`

kwalitee.kwalitee.**check_pydocstyle**(*filename*, ***kwargs*)
Perform static analysis on the given file docstrings.

Parameters

- **filename** (*str*) – path of file to check.
- **ignore** (*list*) – codes to ignore, e.g. ('D400',)
- **match** (*str*) – regex the filename has to match to be checked
- **match_dir** (*str*) – regex everydir in path should match to be checked

Returns errors

Return type *list*

See also:

PyCQA/pydocstyle

`kwalitee.kwalitee.`**`get_options`**(*config=None*)
Build the options from the config object.

`kwalitee.kwalitee.`**`is_file_excluded`**(*filename*, *excludes*)
Check if the file should be excluded.

Parameters

- **`filename`** – file name
- **`excludes`** – list of regex to match

Returns True if the file should be excluded

Module contents

Tool for checking commit kwalitee.

CHAPTER 6

Additional Notes

Notes on how to contribute, legal information and changes are here for the interested.

6.1 Contributing

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

6.1.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/inveniosoftware/kwalitee/issues.

If you are 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.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/inveniosoftware/kwalitee/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 contributions are welcome :)

6.1.2 Get Started!

Ready to contribute? Here's how to set up *invenio* for local development.

1. Fork the *invenio* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/kwalitee.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv kwalitee
$ cd kwalitee/
$ pip install -e .[all]
```

4. Create a branch for local development:

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

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass tests:

```
$ ./run-tests.sh
```

   The tests will provide you with test coverage and also check PEP8 (code style), PEP257 (documentation), flake8 as well as build the Sphinx documentation and run doctests.

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

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

7. Submit a pull request through the GitHub website.

6.1.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests and must not decrease test coverage.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring.

3. The pull request should work for Python 2.7, 3.3, 3.4 and 3.5. Check https://travis-ci.com/inveniosoftware/kwalitee/pull_requests and make sure that the tests pass for all supported Python versions.

6.2 Changelog

This file describes the set of features of each version as well as the incompatibilities.

6.2.1 Release 0.2.0: *The next big thing®*

This version uses a database (SQLite, PostgreSQL) for persistence.

- Adds a new CLI option *-s, –skip-merge-commits* to both *kwalitee check* commands. (#60)
- Support for Docker.
- Support for `.kwalitee.yml` configuration per repository.
- Cli for preparing release notes `kwalitee prepare release`.
- Cli for checking changed files `kwalitee check files`.
- Cli for checking commit messages `kwalitee check message`.
- Support of `push` events.
- Support for multiple repositories.
- Support for multiple users.
- Alembic setup for upcoming migrations.
- New Sphinx documentation.
- Fixes double commenting bug.

Incompatibilities

- The commit statuses are still accessible but are not migrated to the database.
- Previously created git hooks will have to be uninstalled and re-installed as the Flask application is not always created.

6.2.2 Release 0.1.0: *The playground*

Initial version. It supports `pull request` events on one repository and will perform checks on the commit message and files.

- Commit message checks.
- Git hooks.
- PEP8 checks.
- PYDOCSTYLE checks.
- PyFlakes checks.
- License checks.
- Asynchronous checks using RQ.

- New unit tests.
- Auto labelling of the pull requests.
- Skip work in progress (wip) pull requests.

6.3 License

6.4 Authors

kwalitee is developed to support developers of Invenio digital library software.

Contact us at info@inveniosoftware.org.

6.4.1 Contributors

- Jiri Kuncar <jiri.kuncar@cern.ch>
- Yoan Blanc <yoan@dosimple.ch>
- Lars Holm Nielsen <lars.holm.nielsen@cern.ch>
- Mateusz Susik <mateuszsusik@gmail.com>
- Harris Tzovanakis <me@drjova.com>
- Leonardo Rossi <leonardo.r@cern.ch>
- Tibor Simko <tibor.simko@cern.ch>
- David Caro <david@dcaro.es>
- Ivan Masár <helix84@centrum.sk>

k

A

C

E

G

I

K

L

M

P

R

S

T

W