Contributing to Scrapy

> **Important**
>
> Double check that you are reading the most recent version of this document at https://docs.scrapy.org/en/master/contributing.html

There are many ways to contribute to Scrapy. Here are some of them:

- Blog about Scrapy. Tell the world how you're using Scrapy. This will help newcomers with more examples and will help the Scrapy project to increase its visibility.
- Report bugs and request features in the issue tracker, trying to follow the guidelines detailed in Reporting bugs below.
- Submit patches for new functionalities and/or bug fixes. Please read :ref:`writing-patches` and Submitting patches below for details on how to write and submit a patch.

  > **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\(scrapy-master)(docs)contributing.rst, line 21); *backlink*
  >
  > Unknown interpreted text role "ref".

- Join the Scrapy subreddit and share your ideas on how to improve Scrapy. We're always open to suggestions.
- Answer Scrapy questions at Stack Overflow.

Reporting bugs

> **Note**
>
> Please report security issues **only** to scrapy-security@googlegroups.com. This is a private list only open to trusted Scrapy developers, and its archives are not public.

Well-written bug reports are very helpful, so keep in mind the following guidelines when you're going to report a new bug.

- check the :ref:`FAQ <faq>` first to see if your issue is addressed in a well-known question

  > **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\(scrapy-master)(docs)contributing.rst, line 44); *backlink*
  >
  > Unknown interpreted text role "ref".

- if you have a general question about Scrapy usage, please ask it at Stack Overflow (use "scrapy" tag).
- check the open issues to see if the issue has already been reported. If it has, don't dismiss the report, but check the ticket history and comments. If you have additional useful information, please leave a comment, or consider :ref:`sending a pull request <writing-patches>` with a fix.

  > **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\(scrapy-master)(docs)contributing.rst, line 51); *backlink*
  >
  > Unknown interpreted text role "ref".

- search the scrapy-users list and Scrapy subreddit to see if it has been discussed there, or if you're not sure if what you're seeing is a bug. You can also ask in the `#scrapy` IRC channel.
- write **complete, reproducible, specific bug reports**. The smaller the test case, the better. Remember that other developers won't have your project to reproduce the bug, so please include all relevant files required to reproduce it. See for example StackOverflow's guide on creating a Minimal, Complete, and Verifiable example exhibiting the issue.
- the most awesome way to provide a complete reproducible example is to send a pull request which adds a failing test case to the Scrapy testing suite (see :ref:`submitting-patches`). This is helpful even if you don't have an intention to fix the issue yourselves.

  > **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\(scrapy-master)(docs)contributing.rst, line 66); *backlink*
  >
  > Unknown interpreted text role "ref".

- include the output of `scrapy version -v` so developers working on your bug know exactly which version and platform it occurred on, which is often very helpful for reproducing it, or knowing if it was already fixed.

Writing patches

The better a patch is written, the higher the chances that it'll get accepted and the sooner it will be merged.

Well-written patches should:

- contain the minimum amount of code required for the specific change. Small patches are easier to review and merge. So, if you're doing more than one change (or bug fix), please consider submitting one patch per change. Do not collapse multiple changes into a single patch. For big changes consider using a patch queue.
- pass all unit-tests. See Running tests below.
- include one (or more) test cases that check the bug fixed or the new functionality added. See Writing tests below.
- if you're adding or changing a public (documented) API, please include the documentation changes in the same patch. See Documentation policies below.
- if you're adding a private API, please add a regular expression to the `coverage_ignore_pyobjects` variable of `docs/conf.py` to exclude the new private API from documentation coverage checks.

  To see if your private API is skipped properly, generate a documentation coverage report as follows:

  ```
  tox -e docs-coverage
  ```

- if you are removing deprecated code, first make sure that at least 1 year (12 months) has passed since the release that introduced the deprecation. See :ref:`deprecation-policy`.

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\(scrapy-master)(docs)contributing.rst`, **line 111**); ***backlink***

  Unknown interpreted text role "ref".

Submitting patches

The best way to submit a patch is to issue a pull request on GitHub, optionally creating a new issue first.

Remember to explain what was fixed or the new functionality (what it is, why it's needed, etc). The more info you include, the easier will be for core developers to understand and accept your patch.

You can also discuss the new functionality (or bug fix) before creating the patch, but it's always good to have a patch ready to illustrate your arguments and show that you have put some additional thought into the subject. A good starting point is to send a pull request on GitHub. It can be simple enough to illustrate your idea, and leave documentation/tests for later, after the idea has been validated and proven useful. Alternatively, you can start a conversation in the Scrapy subreddit to discuss your idea first.

Sometimes there is an existing pull request for the problem you'd like to solve, which is stalled for some reason. Often the pull request is in a right direction, but changes are requested by Scrapy maintainers, and the original pull request author hasn't had time to address them. In this case consider picking up this pull request: open a new pull request with all commits from the original pull request, as well as additional changes to address the raised issues. Doing so helps a lot; it is not considered rude as long as the original author is acknowledged by keeping his/her commits.

You can pull an existing pull request to a local branch by running `git fetch upstream pull/$PR_NUMBER/head:$BRANCH_NAME_TO_CREATE` (replace 'upstream' with a remote name for scrapy repository, `$PR_NUMBER` with an ID of the pull request, and `$BRANCH_NAME_TO_CREATE` with a name of the branch you want to create locally). See also: https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/checking-out-pull-requests-locally#modifying-an-inactive-pull-request-locally.

When writing GitHub pull requests, try to keep titles short but descriptive. E.g. For bug #411: "Scrapy hangs if an exception raises in start_requests" prefer "Fix hanging when exception occurs in start_requests (#411)" instead of "Fix for #411". Complete titles make it easy to skim through the issue tracker.

Finally, try to keep aesthetic changes (PEP 8 compliance, unused imports removal, etc) in separate commits from functional changes. This will make pull requests easier to review and more likely to get merged.

Coding style

Please follow these coding conventions when writing code for inclusion in Scrapy:

- Unless otherwise specified, follow PEP 8.
- It's OK to use lines longer than 79 chars if it improves the code readability.
- Don't put your name in the code you contribute; git provides enough metadata to identify author of the code. See

https://help.github.com/en/github/using-git/setting-your-username-in-git for setup instructions.

Documentation policies

For reference documentation of API members (classes, methods, etc.) use docstrings and make sure that the Sphinx documentation uses the :mod:`~sphinx.ext.autodoc` extension to pull the docstrings. API reference documentation should follow docstring conventions (PEP 257) and be IDE-friendly: short, to the point, and it may provide short examples.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\(scrapy-master)(docs)contributing.rst, line 187); *backlink***
>
> Unknown interpreted text role "mod".

Other types of documentation, such as tutorials or topics, should be covered in files within the `docs/` directory. This includes documentation that is specific to an API member, but goes beyond API reference documentation.

In any case, if something is covered in a docstring, use the :mod:`~sphinx.ext.autodoc` extension to pull the docstring into the documentation instead of duplicating the docstring in files within the `docs/` directory.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\(scrapy-master)(docs)contributing.rst, line 197); *backlink***
>
> Unknown interpreted text role "mod".

Documentation updates that cover new or modified features must use Sphinxâ€™s :rst:dir:`versionadded` and :rst:dir:`versionchanged` directives. Use `VERSION` as version, we will replace it with the actual version right before the corresponding release. When we release a new major or minor version of Scrapy, we remove these directives if they are older than 3 years.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\(scrapy-master)(docs)contributing.rst, line 202); *backlink***
>
> Unknown interpreted text role "rst:dir".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\(scrapy-master)(docs)contributing.rst, line 202); *backlink***
>
> Unknown interpreted text role "rst:dir".

Documentation about deprecated features must be removed as those features are deprecated, so that new readers do not run into it. New deprecations and deprecation removals are documented in the :ref:`release notes <news>`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\(scrapy-master)(docs)contributing.rst, line 208); *backlink***
>
> Unknown interpreted text role "ref".

Tests

Tests are implemented using the :doc:`Twisted unit-testing framework <twisted:core/development/policy/test-standard>`. Running tests requires :doc:`tox <tox:index>`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\(scrapy-master)(docs)contributing.rst, line 216); *backlink***
>
> Unknown interpreted text role "doc".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\(scrapy-master)(docs)contributing.rst, line 216); *backlink***
>
> Unknown interpreted text role "doc".

Running tests

To run all tests:

```
tox
```

To run a specific test (say `tests/test_loader.py`) use:

```
tox -- tests/test_loader.py
```

To run the tests on a specific :doc:`tox <tox:index>` environment, use `-e <name>` with an environment name from `tox.ini`. For example, to run the tests with Python 3.6 use:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\(scrapy-master)(docs)contributing.rst, line 233); *backlink***
>
> Unknown interpreted text role "doc".

```
tox -e py36
```

You can also specify a comma-separated list of environments, and use :ref:`toxâ€™s parallel mode <tox:parallel_mode>` to run the tests on multiple environments in parallel:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\(scrapy-master)(docs)contributing.rst, line 239); *backlink***
>
> Unknown interpreted text role "ref".

```
tox -e py36,py38 -p auto
```

To pass command-line options to :doc:`pytest <pytest:index>`, add them after `--` in your call to :doc:`tox <tox:index>`. Using `--` overrides the default positional arguments defined in `tox.ini`, so you must include those default positional arguments (`scrapy tests`) after `--` as well:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\(scrapy-master)(docs)contributing.rst, line 245); *backlink***
>
> Unknown interpreted text role "doc".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\(scrapy-master)(docs)contributing.rst, line 245); *backlink***
>
> Unknown interpreted text role "doc".

```
tox -- scrapy tests -x  # stop after first failure
```

You can also use the pytest-xdist plugin. For example, to run all tests on the Python 3.6 :doc:`tox <tox:index>` environment using all your CPU cores:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\(scrapy-master)(docs)contributing.rst, line 252); *backlink***
>
> Unknown interpreted text role "doc".

```
tox -e py36 -- scrapy tests -n auto
```

To see coverage report install :doc:`coverage <coverage:index>` (`pip install coverage`) and run:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\(scrapy-master)(docs)contributing.rst, line 257); *backlink***
>
> Unknown interpreted text role "doc".

```
coverage report
```

see output of `coverage --help` for more options like html or xml report.

Writing tests

All functionality (including new features and bug fixes) must include a test case to check that it works as expected, so please include tests for your patches if you want them to get accepted sooner.

Scrapy uses unit-tests, which are located in the tests/ directory. Their module name typically resembles the full path of the module they're testing. For example, the item loaders code is in:

```
scrapy.loader
```

And their unit-tests are in:

```
tests/test_loader.py
```