## Contributor's Guide

Thanks for your interest in the project! We welcome pull requests from

developers of all skill levels. To get started, simply fork the master branch

on GitHub to your personal account and then clone the fork into your

development environment.

If you would like to contribute but don't already have something in mind,

we invite you to take a look at the issues listed under our [next milestone][ms].

If you see one you'd like to work on, please leave a quick comment so that we don't

end up with duplicated effort. Thanks in advance!

The core Falcon project maintainers are:

\* Kurt Griffiths, Project Lead (\*\*kgriffs\*\* on GH, Gitter, and Twitter)

\* John Vrbanac (\*\*jmvrbanac\*\* on GH, Gitter, and Twitter)

\* Vytautas Liuolia (\*\*vytas7\*\* on GH and Gitter, and \*\*vliuolia\*\* on Twitter)

\* Nick Zaccardi (\*\*nZac\*\* on GH and Gitter)

Please don't hesitate to reach out if you have any questions, or just need a

little help getting started. You can find us in

[falconry/dev][gitter] on Gitter.

Please note that all contributors and maintainers of this project are subject to our [Code of Conduct][coc].

### Pull Requests

Before submitting a pull request, please ensure you have added or updated tests as appropriate, and that all existing tests still pass with your changes. Please also ensure that your coding style follows PEP 8.

You can check all this by running the following from within the Falcon project directory (requires Python 3.8 and 3.5 to be installed on your system):

```bash

$ tools/mintest.sh

```

You may also use Python 3.6 or 3.7 if you don't have 3.8 installed on your system. Substitute "py36" or "py37" as appropriate. For example:

```bash

$ pip install -U tox coverage

$ rm -f .coverage.\*

$ tox -e pep8 && tox -e py35,py37 && tools/testing/combine\_coverage.sh

```

If you are using pyenv, you will need to make sure both 3.8 and 3.5 are available in the current shell, e.g.:

```bash

$ pyenv shell 3.8.0 3.5.8

```

#### Reviews

Falcon is used in a number of mission-critical applications and is known for its stability and reliability. Therefore, we invest a lot of time in carefully reviewing PRs and working with contributors to ensure that every patch merged into the master branch is correct, complete, performant, well-documented, and appropriate.

Project maintainers review each PR for the following:

- [ ] \*\*Design.\*\* Does it do the right thing? Is the end goal well understood and correct?

- [ ] \*\*Correctness.\*\* Is the logic correct? Does it behave correctly according to the goal of the feature or bug fix?

- [ ] \*\*Fit.\*\* Is this feature or fix in keeping with the spirit of the project? Would this idea be better implemented as an add-on?

- [ ] \*\*Standards.\*\* Does this change align with approved or standards-track RFCs, de-facto standards, and currently accepted best practices?

- [ ] \*\*Tests.\*\* Does the PR implement sufficient test coverage in terms of value inputs, Python versions, and lines tested?

- [ ] \*\*Compatibility.\*\* Does it work across all of Falcon's supported Python versions and operating systems?

- [ ] \*\*Performance.\*\* Will this degrade performance for request or response handling? Are there opportunities to optimize the implementation?

- [ ] \*\*Docs.\*\* Does this impact any existing documentation or require new documentation? If so, does this PR include the aforementioned docs, and is the language friendly, clear, helpful, and grammatically correct with no misspellings? Do all docstrings conform to Google style ala [sphinx.ext.napoleon](https://sphinxcontrib-napoleon.readthedocs.io/en/latest/index.html)?

- [ ] \*\*Dependencies.\*\* Does this PR bring in any unnecessary dependencies that would prevent us from keeping the framework lean and mean, jeopardize the reliability of the project, or significantly increase Falcon's attack service?

- [ ] \*\*Changelog.\*\* Does the PR have a changelog entry in newsfragments? Is the

type correct? Try running `towncrier --draft` to ensure it renders correctly.

### Test coverage

Pull requests must maintain 100% test coverage of all code branches. This helps ensure the quality of the Falcon framework. To check coverage before submitting a pull request:

```bash

$ tools/mintest.sh

```

It is necessary to combine test coverage from multiple environments in order to account for branches in the code that are only taken for a given Python version.

The script generates an HTML coverage report that can be viewed by simply opening `.coverage\_html/index.html` in a browser. This can be helpful in tracking down specific lines or branches that are missing coverage.

### Debugging

We use pytest to run all of our tests. Pytest supports pdb and will break as expected on any

`pdb.set\_trace()` calls. If you would like to use pdb++ instead of the standard Python

debugger, simply run the following tox environment. This environment also disables

coverage checking to speed up the test run, making it ideal for quick sanity checks.

```bash

$ tox -e py3\_debug

```

If you wish, you can customize Falcon's `tox.ini` to install alternative debuggers, such as ipdb or pudb.

### Benchmarking

A few simple benchmarks are included with the source under ``falcon/bench``. These can be taken as a rough measure of the performance impact (if any) that your changes have on the framework. You can run these tests by invoking one of the tox environments included for this purpose (see also the ``tox.ini`` file). For example:

```bash

$ tox -e py38\_bench

```

Note that you may pass additional arguments via tox to the falcon-bench command:

```bash

$ tox -e py38\_bench -- -h

$ tox -e py38\_bench -- -b falcon -i 20000

```

Alternatively, you may run falcon-bench directly by creating a new virtual environment and installing falcon directly in development mode. In this example we use pyenv with pyenv-virtualenv from within a falcon source directory:

```bash

$ pyenv virtualenv 3.8.0 falcon-sandbox-38

$ pyenv shell falcon-sandbox-38

$ pip install -r requirements/bench

$ pip install -e .

$ falcon-bench

```

Note that benchmark results for the same code will vary between runs based on a number of factors, including overall system load and CPU scheduling. These factors may be somewhat mitigated by running the benchmarks on a Linux server dedicated to this purpose, and pinning the benchmark process to a specific CPU core.

### Documentation

To check documentation changes (including docstrings), before submitting a PR, ensure the tox job builds the documentation correctly:

```bash

$ tox -e docs

# OS X

$ open docs/\_build/html/index.html

# Gnome

$ gnome-open docs/\_build/html/index.html

# Generic X Windows

$ xdg-open docs/\_build/html/index.html

```

### Code style rules

\* Docstrings are required for classes, attributes, methods, and functions.

\* Docstrings should utilize the [napolean style][docstrings] in order to make them read well, regardless of whether they are viewed through `help()` or on [Read the Docs][rtd].

\* Please try to be consistent with the way existing docstrings are formatted. In particular, note the use of single vs. double backticks as follows:

\* Double backticks

\* Inline code

\* Variables

\* Types

\* Decorators

\* Single backticks

\* Methods

\* Params

\* Attributes

\* Format non-trivial comments using your GitHub nick and one of these prefixes:

\* TODO(riker): Damage report!

\* NOTE(riker): Well, that's certainly good to know.

\* PERF(riker): Travel time to the nearest starbase?

\* APPSEC(riker): In all trust, there is the possibility for betrayal.

\* When catching exceptions, name the variable `ex`.

\* Use whitespace to separate logical blocks of code and to improve readability.

\* No single-character variable names except for trivial indexes when looping,

or in mathematical expressions implementing well-known formulas.

\* Heavily document code that is especially complex and/or clever.

\* When in doubt, optimize for readability.

### Changelog

We use [towncrier](https://towncrier.readthedocs.io/en/actual-freaking-docs/index.html) to manage the changelog. Each PR that modifies the functionality of Falcon should include a short description in a news fragment file in the `docs/\_newsfragments` directory.

The newsfragment file name should have the format `{issue\_number}.{fragment\_type}.rst`, where the fragment type is one of `breakingchange`, `newandimproved`, `bugfix`, or `misc`. If your PR closes another issue, then the original issue number should be used for the newsfragment; otherwise, use the PR number itself.

### Commit Message Format

Falcon's commit message format uses [AngularJS's style guide][ajs], reproduced here for convenience, with some minor edits for clarity.

Each commit message consists of a \*\*header\*\*, a \*\*body\*\* and a \*\*footer\*\*. The header has a special format that includes a \*\*type\*\*, a \*\*scope\*\* and a \*\*subject\*\*:

```

<type>(<scope>): <subject>

<BLANK LINE>

<body>

<BLANK LINE>

<footer>

```

No line may exceed 100 characters. This makes it easier to read the message on GitHub as well as in various git tools.

##### Type

Must be one of the following:

\* \*\*feat\*\*: A new feature

\* \*\*fix\*\*: A bug fix

\* \*\*docs\*\*: Documentation only changes

\* \*\*style\*\*: Changes that do not affect the meaning of the code (white-space, formatting, missing semi-colons, etc)

\* \*\*refactor\*\*: A code change that neither fixes a bug or adds a feature

\* \*\*perf\*\*: A code change that improves performance

\* \*\*test\*\*: Adding missing tests

\* \*\*chore\*\*: Changes to the build process or auxiliary tools and libraries such as documentation generation

##### Scope

The scope could be anything specifying place of the commit change. For example: `$location`, `$browser`, `$compile`, `$rootScope`, `ngHref`, `ngClick`, `ngView`, etc...

##### Subject

The subject contains succinct description of the change:

\* use the imperative, present tense: "change" not "changed" nor "changes"

\* don't capitalize first letter

\* no dot (.) at the end

##### Body

Just as in the \*\*subject\*\*, use the imperative, present tense: "change" not "changed" nor "changes"The body should include the motivation for the change and contrast this with previous behavior.

##### Footer

The footer should contain any information about \*\*Breaking Changes\*\* and is also the place to reference GitHub issues that this commit \*\*Closes\*\*.

[ajs]: https://github.com/angular/angular.js/blob/master/DEVELOPERS.md#-git-commit-guidelines

[docstrings]: https://sphinxcontrib-napoleon.readthedocs.io/en/latest/example\_google.html#example-google-style-python-docstrings

[goog-style]: http://google-styleguide.googlecode.com/svn/trunk/pyguide.html#Comments

[rtd]: https://falcon.readthedocs.io

[coc]: https://github.com/falconry/falcon/blob/master/CODEOFCONDUCT.md

[freenode]: https://www.freenode.net/

[gitter]: https://gitter.im/falconry/dev

[ml-join]: mailto:users-join@mail.falconframework.org?subject=join

[ml-archive]: https://mail.falconframework.org/archives/list/users@mail.falconframework.org/

[ms]: https://github.com/falconry/falcon/milestones