# Contributing to REST framework

> The world can only really be changed one piece at a time. The art is picking that piece.

>

> &mdash; [Tim Berners-Lee][cite]

There are many ways you can contribute to Django REST framework. We'd like it to be a community-led project, so please get involved and help shape the future of the project.

## Community

The most important thing you can do to help push the REST framework project forward is to be actively involved wherever possible. Code contributions are often overvalued as being the primary way to get involved in a project, we don't believe that needs to be the case.

If you use REST framework, we'd love you to be vocal about your experiences with it - you might consider writing a blog post about using REST framework, or publishing a tutorial about building a project with a particular JavaScript framework. Experiences from beginners can be particularly helpful because you'll be in the best position to assess which bits of REST framework are more difficult to understand and work with.

Other really great ways you can help move the community forward include helping to answer questions on the [discussion group][google-group], or setting up an [email alert on StackOverflow][so-filter] so that you get notified of any new questions with the `django-rest-framework` tag.

When answering questions make sure to help future contributors find their way around by hyperlinking wherever possible to related threads and tickets, and include backlinks from those items if relevant.

## Code of conduct

Please keep the tone polite & professional. For some users a discussion on the REST framework mailing list or ticket tracker may be their first engagement with the open source community. First impressions count, so let's try to make everyone feel welcome.

Be mindful in the language you choose. As an example, in an environment that is heavily male-dominated, posts that start 'Hey guys,' can come across as unintentionally exclusive. It's just as easy, and more inclusive to use gender neutral language in those situations. (e.g. 'Hey folks,')

The [Django code of conduct][code-of-conduct] gives a fuller set of guidelines for participating in community forums.

# Issues

It's really helpful if you can make sure to address issues on the correct channel. Usage questions should be directed to the [discussion group][google-group]. Feature requests, bug reports and other issues should be raised on the GitHub [issue tracker][issues].

Some tips on good issue reporting:

\* When describing issues try to phrase your ticket in terms of the \*behavior\* you think needs changing rather than the \*code\* you think need changing.

\* Search the issue list first for related items, and make sure you're running the latest version of REST framework before reporting an issue.

\* If reporting a bug, then try to include a pull request with a failing test case. This will help us quickly identify if there is a valid issue, and make sure that it gets fixed more quickly if there is one.

\* Feature requests will often be closed with a recommendation that they be implemented outside of the core REST framework library. Keeping new feature requests implemented as third party libraries allows us to keep down the maintenance overhead of REST framework, so that the focus can be on continued stability, bug fixes, and great documentation.

\* Closing an issue doesn't necessarily mean the end of a discussion. If you believe your issue has been closed incorrectly, explain why and we'll consider if it needs to be reopened.

## Triaging issues

Getting involved in triaging incoming issues is a good way to start contributing. Every single ticket that comes into the ticket tracker needs to be reviewed in order to determine what the next steps should be. Anyone can help out with this, you just need to be willing to:

\* Read through the ticket - does it make sense, is it missing any context that would help explain it better?

\* Is the ticket reported in the correct place, would it be better suited as a discussion on the discussion group?

\* If the ticket is a bug report, can you reproduce it? Are you able to write a failing test case that demonstrates the issue and that can be submitted as a pull request?

\* If the ticket is a feature request, do you agree with it, and could the feature request instead be implemented as a third party package?

\* If a ticket hasn't had much activity and it addresses something you need, then comment on the ticket and try to find out what's needed to get it moving again.

# Development

To start developing on Django REST framework, clone the repo:

git clone https://github.com/encode/django-rest-framework

Changes should broadly follow the [PEP 8][pep-8] style conventions, and we recommend you set up your editor to automatically indicate non-conforming styles.

## Testing

To run the tests, clone the repository, and then:

# Setup the virtual environment

python3 -m venv env

source env/bin/activate

pip install django

pip install -r requirements.txt

# Run the tests

./runtests.py

### Test options

Run using a more concise output style.

./runtests.py -q

Run the tests using a more concise output style, no coverage, no flake8.

./runtests.py --fast

Don't run the flake8 code linting.

./runtests.py --nolint

Only run the flake8 code linting, don't run the tests.

./runtests.py --lintonly

Run the tests for a given test case.

./runtests.py MyTestCase

Run the tests for a given test method.

./runtests.py MyTestCase.test\_this\_method

Shorter form to run the tests for a given test method.

./runtests.py test\_this\_method

Note: The test case and test method matching is fuzzy and will sometimes run other tests that contain a partial string match to the given command line input.

### Running against multiple environments

You can also use the excellent [tox][tox] testing tool to run the tests against all supported versions of Python and Django. Install `tox` globally, and then simply run:

tox

## Pull requests

It's a good idea to make pull requests early on. A pull request represents the start of a discussion, and doesn't necessarily need to be the final, finished submission.

It's also always best to make a new branch before starting work on a pull request. This means that you'll be able to later switch back to working on another separate issue without interfering with an ongoing pull requests.

It's also useful to remember that if you have an outstanding pull request then pushing new commits to your GitHub repo will also automatically update the pull requests.

GitHub's documentation for working on pull requests is [available here][pull-requests].

Always run the tests before submitting pull requests, and ideally run `tox` in order to check that your modifications are compatible on all supported versions of Python and Django.

Once you've made a pull request take a look at the Travis build status in the GitHub interface and make sure the tests are running as you'd expect.

## Managing compatibility issues

Sometimes, in order to ensure your code works on various different versions of Django, Python or third party libraries, you'll need to run slightly different code depending on the environment. Any code that branches in this way should be isolated into the `compat.py` module, and should provide a single common interface that the rest of the codebase can use.

# Documentation

The documentation for REST framework is built from the [Markdown][markdown] source files in [the docs directory][docs].

There are many great Markdown editors that make working with the documentation really easy. The [Mou editor for Mac][mou] is one such editor that comes highly recommended.

## Building the documentation

To build the documentation, install MkDocs with `pip install mkdocs` and then run the following command.

mkdocs build

This will build the documentation into the `site` directory.

You can build the documentation and open a preview in a browser window by using the `serve` command.

mkdocs serve

## Language style

Documentation should be in American English. The tone of the documentation is very important - try to stick to a simple, plain, objective and well-balanced style where possible.

Some other tips:

\* Keep paragraphs reasonably short.

\* Don't use abbreviations such as 'e.g.' but instead use the long form, such as 'For example'.

## Markdown style

There are a couple of conventions you should follow when working on the documentation.

##### 1. Headers

Headers should use the hash style. For example:

### Some important topic

The underline style should not be used. \*\*Don't do this:\*\*

Some important topic

====================

##### 2. Links

Links should always use the reference style, with the referenced hyperlinks kept at the end of the document.

Here is a link to [some other thing][other-thing].

More text...

[other-thing]: http://example.com/other/thing

This style helps keep the documentation source consistent and readable.

If you are hyperlinking to another REST framework document, you should use a relative link, and link to the `.md` suffix. For example:

[authentication]: ../api-guide/authentication.md

Linking in this style means you'll be able to click the hyperlink in your Markdown editor to open the referenced document. When the documentation is built, these links will be converted into regular links to HTML pages.

##### 3. Notes

If you want to draw attention to a note or warning, use a pair of enclosing lines, like so:

---

\*\*Note:\*\* A useful documentation note.

---

[cite]: https://www.w3.org/People/Berners-Lee/FAQ.html

[code-of-conduct]: https://www.djangoproject.com/conduct/

[google-group]: https://groups.google.com/forum/?fromgroups#!forum/django-rest-framework

[so-filter]: https://stackexchange.com/filters/66475/rest-framework

[issues]: https://github.com/encode/django-rest-framework/issues?state=open

[pep-8]: https://www.python.org/dev/peps/pep-0008/

[pull-requests]: https://help.github.com/articles/using-pull-requests

[tox]: https://tox.readthedocs.io/en/latest/

[markdown]: https://daringfireball.net/projects/markdown/basics

[docs]: https://github.com/encode/django-rest-framework/tree/master/docs

[mou]: http://mouapp.com/