# Contributing to SpecFlow

This contributing guide is based on the contributing for ElixirScript (which was based on the guide for contributing to Elixir), with changes suitable for this project.

Contributing can have many faces. It can be code, but also answering questions and issues, writing documentation or trying out preview versions are contributions.

## Using the issue tracker

Use the issues tracker for:

\* [bug reports](#bug)

\* [pull requests](#pull-requests)

\* [feature requests](#feature-request)

### Bug reports

A bug is a \_demonstrable problem\_ that is caused by the code in the repository.

Good bug reports are extremely helpful - thank you!

Guidelines for bug reports:

1. \*\*Use the GitHub issue search\*\* &mdash; [check if the issue has already been

reported](https://github.com/techtalk/SpecFlow/search?type=Issues).

2. \*\*Check if the issue has been fixed\*\* &mdash; try to reproduce it using the

`master` branch in the repository.

3. \*\*Isolate and report the problem\*\* &mdash; ideally create a reduced test

case. Fill out the provided template.

4. \*\*Provide a screenshot or example code\*\* &mdash; you might in a situation that

is very tied to your specific use-case, so provide as much information as

possible.

Please try to be as detailed as possible in your report. Please provide steps to

reproduce the issue as well as the outcome you were expecting! And fill out the issue template. If you don't provide this information, we have to ask them again from you. All these details

will help developers to fix any potential bugs.

We move issues that need help, but may not be of a criticial nature or require

intensive SpecFlow knowledge, to [Up For Grabs](https://github.com/techtalk/SpecFlow/labels/up-for-grabs).

This is a list of easier tasks that anybody who wants to get into SpecFlow

development can try. Thank you!

### Feature requests

Feature requests are welcome. But please take a moment to find

out whether your idea fits with the scope and aims of the project. It's up to \*you\*

to make a strong case to convince the community of the merits of this feature.

Since much of the work is done be volunteers, someone who believes in the

idea will have to write the code. Please provide as much detail and context as possible.

If active work is not made on your feature request within 2-3 months, we may close your

feature request. Anybody is free to pick up on the idea, and we welcome the work! But in

the interest of keeping the issue list down to a maintainable level, we have to do some

pruning from time-to-time.

### Pull requests

Good pull requests - patches, improvements, new features - are a fantastic

help. They should remain focused in scope and avoid containing unrelated

commits.

All needed informations about developing SpecFlow can be found at [Development.md](/docs/Development.md).

\*\*NOTE\*\*: Do not send code style changes as pull requests like changing

the indentation of some particular code snippet or how a function is called.

Those will not be accepted as they pollute the repository history with non

functional changes and are often based on personal preferences.

\*\*IMPORTANT\*\*: By submitting a patch, you agree that your work will be

licensed under the license used by the project.

If you have any large pull request in mind (e.g. implementing features,

refactoring code, etc), \*\*please ask first\*\* otherwise you risk spending

a lot of time working on something that the project's developers might

not want to merge into the project.

Please adhere to the coding conventions in the project (indentation,

accurate comments, etc.) and don't forget to add your own tests and

documentation. When working with Git, we recommend the following process

in order to craft an excellent pull request:

1. [Fork](https://help.github.com/fork-a-repo/) the project, clone your fork,

and configure the remotes:

```sh

# Clone your fork of the repo into the current directory

git clone https://github.com/<your-username>/SpecFlow

# Navigate to the newly cloned directory

cd SpecFlow

# Assign the original repo to a remote called "upstream"

git remote add upstream https://github.com/techtalk/SpecFlow

```

2. If you cloned a while ago, get the latest changes from upstream:

```sh

git checkout master

git pull upstream master

```

3. Create a new topic branch (off of `master`) to contain your feature, change,

or fix.

\*\*IMPORTANT\*\*: Making changes in `master` is discouraged. You should always

keep your local `master` in sync with upstream `master` and make your

changes in topic branches.

```sh

git checkout -b <topic-branch-name>

```

4. Commit your changes in logical chunks. Keep your commit messages organized,

with a short description in the first line and more detailed information on

the following lines. Feel free to use Git's

[interactive rebase](https://help.github.com/articles/interactive-rebase)

feature to tidy up your commits before making them public.

5. Make sure all the tests are still passing.

This is needed to ensure your changes can

pass all the tests.

6. Push your topic branch up to your fork:

```sh

git push origin <topic-branch-name>

```

7. [Open a Pull Request](https://help.github.com/articles/using-pull-requests/)

with a clear title and description.

8. If you haven't updated your pull request for a while, you should consider

rebasing on master and resolving any conflicts.

\*\*IMPORTANT\*\*: \_Never ever\_ merge upstream `master` into your branches. You

should always `git rebase` on `master` to bring your changes up to date when

necessary.

```sh

git checkout master

git pull upstream master

git checkout <your-topic-branch>

git rebase master

```

Thank you for your contributions!