# Contributing to JHipster

Are you ready to contribute to JHipster? We'd love to have you on board, and we will help you as much as we can. Here are the guidelines we'd like you to follow so that we can be of more help:

- [Questions and help](#question)

- [Issues and Bugs](#issue)

- [Bug bounties](#bounties)

- [Feature Requests](#feature)

- [RFCs](#rfcs)

- [Submission Guidelines](#submit)

- [Generator development setup](#setup)

- [Coding Rules](#rules)

- [Git Commit Guidelines](#commit)

And don't forget we also accept [financial contributions to the project](https://www.jhipster.tech/sponsors/) using OpenCollective.

## <a name="question"></a> Questions and help

This is the JHipster bug tracker, and it is used for [Issues and Bugs](#issue) and for [Feature Requests](#feature). It is \*\*not\*\* a help desk or a support forum.

If you have a question on using JHipster, or if you need help with your JHipster project, please [read our help page](https://www.jhipster.tech/help/) and use the [JHipster tag on StackOverflow](http://stackoverflow.com/tags/jhipster) or join our [Gitter.im chat room](https://gitter.im/jhipster/generator-jhipster).

## <a name="issue"></a> Issues and Bugs

If you find a bug in the source code or a mistake in the documentation, you can help us by [submitting a ticket](https://opensource.guide/how-to-contribute/#opening-an-issue) to our [GitHub issues](https://github.com/jhipster/generator-jhipster/issues). Even better, you can submit a Pull Request to our [JHipster generator project](https://github.com/jhipster/generator-jhipster) or to our [Documentation project](https://github.com/jhipster/jhipster.github.io).

\*\*Please see the Submission Guidelines below\*\*.

## <a name="bounties"></a> Bug bounties

If you submitted a Pull Request that fixes a ticket with the "\$100" tag, then you are eligible to our bug bounty program! Go to our [bug bounties documentation](https://www.jhipster.tech/bug-bounties/) for more information, and claim your money.

## <a name="feature"></a> Feature Requests

You can request a new feature by submitting a ticket to our [GitHub issues](https://github.com/jhipster/generator-jhipster/issues). If you

would like to implement a new feature then consider what kind of change it is:

- \*\*Major Changes\*\* that you wish to contribute to the project should be discussed first. Please open a ticket which clearly states that it is a feature request in the title and explain clearly what you want to achieve in the description, and the JHipster team will discuss with you what should be done in that ticket. You can then start working on a Pull Request. In order to communicate major changes proposals and receive reviews from the core team, you can also submit an RFC.

- \*\*Small Changes\*\* can be proposed without any discussion. Open up a ticket which clearly states that it is a feature request in the title. Explain your change in the description, and you can propose a Pull Request straight away.

## <a name="rfcs"></a> RFCs

Sometimes, major feature requests are "complex" or "substantial". In this case, Github Issues might not be the best tool to to present them because we will need a lot of going back and forth to reach a consensus.

So we ask that these feature request be put through a formal design process and have their specifications described in an "RFC" (request for comments) that will be validated by the team through a Pull Request Review.

The RFC process is intended to provide a consistent and controlled path for major features and directions of the project.

To submit an RFC follow those steps:

1. Discuss the RFC proposal with the core team through Github issues or other channels

2. Create the initial Github issue for the Feature Request if it doesn't already exist

3. Copy the `rfcs/0-jhipster-rfc-template.md` to `rfcs/${featureRequestIssueNumber}-my-feature-request-name.md`

4. Fill in the RFC, make sure to complete every required section

5. Submit the RFC as a Pull Request with the summary of the proposal in the PR description

6. Build consensus and integrate feedback from the reviewers

7. The Pull Request is either accepted (merged), rejected (closed) or postponed (given an "on hold" status)

Note: The JHipster RFC process is inspired by [Rust RFCs](https://rust-lang.github.io/rfcs/).

## <a name="submit"></a> Submission Guidelines

### [Submitting an Issue](https://opensource.guide/how-to-contribute/#opening-an-issue)

Before you submit your issue search the [archive](https://github.com/jhipster/generator-jhipster/issues?utf8=%E2%9C%93&q=is%3Aissue), maybe your question was already answered.

If your issue appears to be a bug, and has not been reported, open a new issue.

Help us to maximize the effort we can spend fixing issues and adding new

features, by not reporting duplicate issues. Providing the following information will increase the

chances of your issue being dealt with quickly:

- \*\*Overview of the issue\*\* - if an error is being thrown a stack trace helps

- \*\*Motivation for or Use Case\*\* - explain why this is a bug for you

- \*\*Reproduce the error\*\* - an unambiguous set of steps to reproduce the error. If you have a JavaScript error, maybe you can provide a live example with

[JSFiddle](http://jsfiddle.net/)?

- \*\*Related issues\*\* - has a similar issue been reported before?

- \*\*Suggest a Fix\*\* - if you can't fix the bug yourself, perhaps you can point to what might be

causing the problem (line of code or commit)

- \*\*JHipster Version(s)\*\* - is it a regression?

- \*\*JHipster configuration, a `.yo-rc.json` file generated in the root folder\*\* - this will help us to replicate the scenario, you can remove the rememberMe key.

- \*\*Entity configuration(s) `entityName.json` files generated in the `.jhipster` directory\*\* - if the error is during an entity creation or associated with a specific entity

- \*\*Browsers and Operating System\*\* - is this a problem with all browsers or only IE8?

You can use `jhipster info` to provide us the information we need.

Click [here](issue-template) to open a bug issue with a pre-filled template. For feature requests and enquiries you can use [this template][feature-template].

You can run `jhipster info` in your project folder to get most of the above required info.

Issues opened without any of these info will be \*\*closed\*\* without any explanation.

### [Submitting a Pull Request](https://opensource.guide/how-to-contribute/#opening-a-pull-request)

Before you submit your pull request consider the following guidelines:

- Search [GitHub](https://github.com/jhipster/generator-jhipster/pulls?utf8=%E2%9C%93&q=is%3Apr) for an open or closed Pull Request

that relates to your submission.

- If you want to modify the JHipster generator, read our [Generator development setup](#setup)

- Make your changes in a new git branch

```shell

git checkout -b my-fix-branch master

```

- Create your patch, \*\*including appropriate test cases\*\*.

- Follow our [Coding Rules](#rules).

- Generate a new JHipster project, and ensure that all tests pass

```shell

mvnw verify -Pprod

```

- Test that the new project runs correctly:

```shell

mvnw spring-boot:run

```

- You can generate our Continuous Integration (with GitHub Actions and Azure Pipelines) by following [this](#local-build)

- Commit your changes using a descriptive commit message that follows our

[commit message conventions](#commit-message-format).

```shell

git commit -a

```

Note: the optional commit `-a` command line option will automatically "add" and "rm" edited files.

- Push your branch to GitHub:

```shell

git push origin my-fix-branch

```

- In GitHub, send a pull request to `jhipster/generator-jhipster:master`.

- If we suggest changes then

- Make the required updates.

- Re-run the JHipster tests on your sample generated project to ensure tests are still passing.

- Rebase your branch and force push to your GitHub repository (this will update your Pull Request):

```shell

git rebase master -i

git push -f

```

That's it! Thank you for your contribution!

#### Resolving merge conflicts ("This branch has conflicts that must be resolved")

Sometimes your PR will have merge conflicts with the upstream repository's master branch. There are several ways to solve this but if not done correctly this can end up as a true nightmare. So here is one method that works quite well.

- First, fetch the latest information from the master

```shell

git fetch upstream

```

- Rebase your branch against the upstream/master

```shell

git rebase upstream/master

```

- Git will stop rebasing at the first merge conflict and indicate which file is in conflict. Edit the file, resolve the conflict then

```shell

git add <the file that was in conflict>

git rebase --continue

```

- The rebase will continue up to the next conflict. Repeat the previous step until all files are merged and the rebase ends successfully.

- Re-run the JHipster tests on your sample generated project to ensure tests are still passing.

- Force push to your GitHub repository (this will update your Pull Request)

```shell

git push -f

```

#### After your pull request is merged

After your pull request is merged, you can safely delete your branch and pull the changes

from the main (upstream) repository:

- Delete the remote branch on GitHub either through the GitHub web UI or your local shell as follows:

```shell

git push origin --delete my-fix-branch

```

- Check out the master branch:

```shell

git checkout master -f

```

- Delete the local branch:

```shell

git branch -D my-fix-branch

```

- Update your master with the latest upstream version:

```shell

git pull --ff upstream master

```

## <a name="setup"></a> Generator development setup

JHipster is a [Yeoman Generator](http://yeoman.io/), so you must follow the [Yeoman authoring documentation](http://yeoman.io/authoring/) in order to be able to run and test your changes.

Here are the most important steps.

### Fork the generator-jhipster project

Go to the [generator-jhipster project](https://github.com/jhipster/generator-jhipster) and click on the "fork" button. You can then clone your own fork of the project, and start working on it.

[Please read the GitHub forking documentation for more information](https://help.github.com/articles/fork-a-repo)

### Set NPM to use the cloned project

In your cloned `generator-jhipster` project, type `npm link`.

This will do a symbolic link from the global `node\_modules` version to point to this folder, so when we run `jhipster`, you will now use the development version of JHipster.

For testing, you will want to generate an application, and there is a specific issue here: for each application, JHipster installs a local version of itself. This is made to enable several applications to each use a specific JHipster version (application A uses JHipster 3.1.0, and application B uses JHipster 3.2.0).

To overcome this you need to run `npm link generator-jhipster` on the generated project folder as well, so that the local version has a symbolic link to the development version of JHipster.

To put it in a nutshell, you need to:

1. run `npm link` on the `generator-jhipster` project

2. run `npm link generator-jhipster` on the generated application folder (you need to do this for each application you create)

Now, running the 'jhipster' command should run your locally installed JHipster version directly from sources. Check that the symbolic link is correct with the following command :

```shell

? ~ ll $(which jhipster)

lrwxr-xr-x 1 username admin 63B May 15 11:03 /usr/local/bin/jhipster -> ../../../Users/username/github/generator-jhipster/cli/jhipster.js

```

You can test your setup by making a small change in your cloned generator, and running again on an existing JHipster project:

```shell

jhipster

```

Depending on which parts of the generator you have changed, do not forget to run jhipster command with the proper arguments e.g. when updating the entity template run:

```shell

jhipster --with-entities

```

You should see your changes reflected in the generated project.

Note: The generated project might not build properly in case the generator is using a

snapshot version of [jhipster/jhipster](https://github.com/jhipster/jhipster). This issue is mentioned in; https://github.com/jhipster/generator-jhipster/issues/9571. In

this case clone the jhipster/jhipster project and build it using:

```shell script

./mvnw clean install -Dgpg.skip=true

```

or on Windows:

```

.\mvnw.cmd clean install -D"gpg.skip=true"

```

### Use a text editor

As modifying the JHipster generator includes modifying Java and JavaScript templates, most IDE will not work correctly. We recommend you use a text editor like [Atom](https://atom.io/) or [VSCode](https://code.visualstudio.com/) to code your changes. The ESLint and EditorConfig extensions are recommended to help with respecting code conventions.

### Use a debugger

It is possible to debug JHipster's code using a Node.js debugger. To achieve this, setup your debugger to launch `cli/jhipster.js`.

#### Debugging with VSCode

To start debugging JHipster with \*\*VSCode\*\*, open the generator code in your workspace and simply press F5 (or click the green arrow in the \*\*Debug\*\* menu reachable with Ctrl+Shift+D). This will start the generator in debug mode and generate files in the `test-integration/samples/app-sample-dev` folder.

It is also possible to debug sub generators by selecting one of the other debug options (for example `jhipster entity`). Those debug configurations are specified in the `.vscode/launch.json` file.

## Local Build

You can run the builds locally by following below commands

Go into the `test-integration` folder with `cd test-integration` from the generator source code root folder

Run `./generate-sample.sh <command\_name> [folder] [sample\_name:optional] [type of entity]`

This will create a folder with configuration and entities. Then, you can generate manually a JHipster project and test it.

Command name can be as below

`list`: List all sample names

`generate`: Generate the sample

## <a name="rules"></a> Coding Rules

To ensure consistency throughout the source code, keep these rules in mind as you are working:

- All features or bug fixes \*\*must be tested\*\* by one or more tests.

- All files must follow the [.editorconfig file](http://editorconfig.org/) located at the root of the JHipster generator project. Please note that generated projects use the same `.editorconfig` file, so that both the generator and the generated projects share the same configuration.

- Java files \*\*must be\*\* formatted using Intellij IDEA default code style.

- Generators JavaScript files \*\*must follow\*\* the eslint configuration defined at the project root, which is based on [Airbnb JavaScript Style Guide](https://github.com/airbnb/javascript).

- Any client side feature/change should be done for both Angular and react clients

- Web apps JavaScript files \*\*must follow\*\* [Google's JavaScript Style Guide](https://google.github.io/styleguide/jsguide.html).

- Angular Typescript files \*\*must follow\*\* the [Official Angular style guide](https://angular.io/styleguide).

- React/Redux Typescript files \*\*may follow\*\* the [React/Redux Typescript guide](https://github.com/piotrwitek/react-redux-typescript-guide).

Please ensure to run `npm run lint` and `npm test` on the project root before submitting a pull request. You can also run `npm run lint-fix` to fix some of the lint issues automatically.

## <a name="templates"></a> Template Guidelines

The template engine used by yeoman is [EJS](http://ejs.co/), its syntax is fairly simple.

For simple code (few lines), logic can be embedded in the main file but if logic becomes more complex it's better to externalise the JS fragment to a sub template included by the first one and located in same folder.

Sub templates should be named with the `ejs` extension because it's the default one, it enables editors to apply correct syntax highlighting and it enables us to use a very concise syntax:

<%- include('field\_validators'); -%>

This statement means that [\\_Entity.java](generators/entity/templates/src/main/java/package/domain/\_Entity.java) template includes [field\_validators.ejs](generators/entity/templates/src/main/java/package/domain/field\_validators.ejs) sub template.

Sub templates can be unit tested.

## <a name="commit"></a> Git Commit Guidelines

We have rules over how our git commit messages must be formatted. Please ensure to [squash](https://help.github.com/articles/about-git-rebase/#commands-available-while-rebasing) unnecessary commits so that your commit history is clean.

If the commit only involves documentation changes you can skip the continuous integration pipelines using `[ci skip]` or `[skip ci]` in your commit message header.

### <a name="commit-message-format"></a> Commit Message Format

Each commit message consists of a \*\*header\*\*, a \*\*body\*\* and a \*\*footer\*\*.

```

<header>

<BLANK LINE>

<body>

<BLANK LINE>

<footer>

```

Any line of the commit message cannot be longer 100 characters! This allows the message to be easier

to read on GitHub as well as in various git tools.

### Header

The Header contains a 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

If your change is simple, the Body is optional.

Just as in the Header, 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 is the place to reference GitHub issues that this commit \*\*Closes\*\*.

You \*\*must\*\* use the [GitHub keywords](https://help.github.com/articles/closing-issues-via-commit-messages) for

automatically closing the issues referenced in your commit.

### Example

For example, here is a good commit message:

```

upgrade to Spring Boot 1.1.7

upgrade the Maven and Gradle builds to use the new Spring Boot 1.1.7,

see http://spring.io/blog/2014/09/26/spring-boot-1-1-7-released

Fix #1234

```

### Regular Contributor Guidelines

These are some of the guidelines that we would like to emphasize if you are a regular contributor to the project

or joined the [JHipster team](https://www.jhipster.tech/team/).

- We recommend not committing directly to master, but always submit changes through PRs.

- Before merging, try to get at least one review on the PR.

- Add appropriate labels to issues and PRs that you create (if you have permission to do so).

- Follow the project's [policies](https://www.jhipster.tech/policies/#-policies).

- Follow the project's [Code of Conduct](https://github.com/jhipster/generator-jhipster/blob/master/CODE\_OF\_CONDUCT.md)

and be polite and helpful to users when answering questions/bug reports and when reviewing PRs.

- We work on our free time so we have no obligation nor commitment. Work/life balance is important, so don't

feel tempted to put in all your free time fixing something.

[issue-template]: https://github.com/jhipster/generator-jhipster/issues/new?template=BUG\_REPORT.md

[feature-template]: https://github.com/jhipster/generator-jhipster/issues/new?template=FEATURE\_REQUEST.md