## Ways to Contribute

There are many ways to contribute to Spock:

\* Spreading the word by talking, tweeting, blogging, presenting, submitting talks, writing tutorials or articles (or a book!), sharing success stories, adding your project/company to [Who is Using Spock](http://wiki.spockframework.org/WhoIsUsingSpock), etc.

\* Helping other users by participating in [forum](http://forum.spockframework.org) discussions, answering [Spock questions](https://stackoverflow.com/questions/tagged/spock) on Stack Overflow, etc.

\* Providing early feedback on [development snapshots](https://oss.sonatype.org/content/repositories/snapshots/org/spockframework/) and new features

\* Improving and extending the [reference documentation](http://docs.spockframework.org) [(source)](https://github.com/spockframework/spock/tree/master/docs)

\* Translating blog posts, articles, and the reference documentation to other languages (with permission of the authors)

\* Fixing open issues listed in the [issue tracker](http://issues.spockframework.org)

\* Proposing, discussing, and implementing new features

\* Building the next generation of [Spock Web Console](http://webconsole.spockframework.org), or an interactive Spock tutorial

\* Designing a Spock logo and/or website (Disclaimer: Our expectations are high on this one, and obviously we'll need to be heavily involved)

\* Hiring us for Spock/[Geb](https://gebish.org) related training and consulting

\* Donating money to fund further development

\* Suprising us with some other form of contribution!

All forms of contribution are very much appreciated.

## Communication

Good communication makes a big difference. We are always eager to listen, reflect, and discuss. Don't hesitate to get in touch via the [issue tracker](http://issues.spockframework.org), [user forum](http://forum.spockframework.org), or [dev forum](http://dev-forum.spockframework.org). Choose whatever medium feels most appropriate.

## Triage Issues [![Open Source Helpers](https://www.codetriage.com/spockframework/spock/badges/users.svg)](https://www.codetriage.com/spockframework/spock)

You can triage issues which may include reproducing bug reports or asking for vital information, such as version numbers or reproduction instructions. If you would like to start triaging issues, one easy way to get started is to [subscribe to spock on CodeTriage](https://www.codetriage.com/spockframework/spock).

## Contributing Code/Docs

To contribute code or documentation, please submit a pull request to the [GitHub repository](https://github.com/spockframework/spock).

A good way to familiarize yourself with the codebase and contribution process is to look for and tackle low-hanging fruits in the [issue tracker](http://issues.spockframework.org). Before embarking on a more ambitious contribution, please quickly [get in touch](#communication) with us. This will help to make sure that the contribution is aligned with Spock's overall direction and goals, and gives us a chance to guide design and implementation where needed.

\*\*We appreciate your effort, and want to avoid a situation where a contribution requires extensive rework (by you or by us), sits in the queue for a long time, or cannot be accepted at all!\*\*

### What Makes a Good Pull Request

When reviewing pull requests, we value the following qualities (in no particular order):

\* Carefully crafted, clean and concise code

\* Small, focused commits that tackle one thing at a time

\* New code blends in well with existing code, respecting established coding standards and practices

\* Tests are updated/added along with the code, communicate intent and cover important cases (see [Tests](#tests) for additional information)

\* Documentation (Javadoc, Groovydoc, reference documentation) is updated/added along with the code

\* A good commit message that follows the [seven rules](https://chris.beams.io/posts/git-commit/)

Don't be intimidated by these words. Pull requests that satisfy Spock's overall direction and goals (see above), are crafted carefully, and aren't aiming too high, have a good chance of getting accepted. Before doing so, we may ask for some concrete improvements to be made, in which case we hope for your cooperation.

### Implementation Language

The implementation language for the [spock-core](https://github.spockframework.org/spock/tree/master/spock-core) module is Java. Java is also the default language for all other modules (except `spock-specs`), but it's fine to use Groovy when there is a concrete reason. As a general guideline, use the same language as the code around you.

### Compatibility

Spock supports JRE 1.6 and higher. Therefore, language features and APIs that are only available in Java 1.7 or higher cannot be used. Exceptions to this rule need to be discussed beforehand. The same goes for changes to user-visible behavior.

### Tests

All tests are written in Spock. Tests for `spock-core` are located in the `spock-specs` project; all other projects have co-located tests. For each user-visible behavior, a functional test is required. Functional tests for `spock-core` are located under [`spock-specs/src/test/groovy/org/spockframework/smoke`](https://github.spockframework.org/spock/tree/master/spock-specs/src/test/groovy/org/spockframework/smoke).

## Development Tools

### Command Line Build

Spock is built with [Gradle](https://www.gradle.org). The only prerequsite for executing the build is an installation of JDK 1.6 (or higher). After cloning the [GitHub repository](https://github.com/spockframework/spock), cd into the top directory and execute `./gradlew build` (Windows: `gradlew build`). The build should succeed without any errors. `gradlew tasks` lists the available tasks. Always use the Gradle Wrapper (`gradlew` command) rather than your own Gradle installation.

### CI Build

Each push to the official GitHub repository triggers a [Linux CI build](http://builds.spockframework.org) and [Windows CI build](http://winbuilds.spockframework.org). Pull requests are built as well.

### IDE Setup

Using an IDE is recommended but not mandatory. Whether or not you use an IDE, please make sure that `gradlew build` succeeds before submitting a pull request.

#### IntelliJ IDEA

IntelliJ IDEA 14+ is the preferred IDE for developing Spock. To generate an IDEA project configuration:

\* `./gradlew cleanIdea idea`, followed by

\* `File -> Open` in Intellij, then

\* select the appropriate `spock-2.0.ipr`.

Note: You can provide an optional 'variant' flag to the gradle build `-Dvariant=[2.0|2.3|2.4]` that specifies the version of groovy you wish to use. For example, if we wish to use groovy 2.4, we would run `./gradlew cleanIdea idea -Dvariant=2.4` and import the generated `spock-2.4.ipr` into Intellij.

This should result in a fully functional IDE setup where:

\* Git VCS integration is configured

\* Important formatter settings are configured (e.g. two spaces indent)

\* Both Java and Groovy code compiles without problems

\* All tests can be run without problems

From time to time (e.g. when someone has added a new dependency), it may be necessary to resync the IDEA project with the Gradle build. This is done by rerunning the steps above.

Note: Unfortunately there are currently [some issues](https://github.com/spockframework/spock/issues/70) with Intellij's Gradle support that prevent integration with the Spock project. Please use the method described above rather than importing the top-level `build.gradle` file.

#### Eclipse

Eclipse 3.7+ with latest [Groovy plugin](https://github.com/groovy/groovy-eclipse/wiki) should work reasonably well for developing Spock. To import the Gradle build into Eclipse, either run `gradlew eclipse` and import the generated Eclipse projects via `File->Import->General->Existing Project into Workspace`, or install the [Eclipse Gradle Tooling](https://github.com/spring-projects/eclipse-integration-gradle/) and import via `File->Import->Gradle->Gradle Project`. Either method should result in an IDE setup where:

\* Both Java and Groovy code compiles without problems

\* All tests can be run without problems

From time to time (e.g. when someone has added a new dependency), it may be necessary to resync the Eclipse project with the Gradle build. Depending on the method of import (see above), this is done by re-running `gradlew [cleanEclipse] eclipse` and then refreshing all Eclipse projects (F5), or by selecting `Gradle->Refresh All` in the Package Explorer's context menu.

If you encounter any problems with the IDE setup, or want to make some improvements to it, please [get in touch](#communication) with us.

## The End

Thanks for reading this far. We are looking forward to your contributions!

The Spock Framework Team