Contributing to Guava

Thank you so much for wanting to contribute to Guava! There are a couple ways to help out.

Evangelize!

Just tell people about Guava. We believe that a bigger, more involved community makes for a better library, and that better libraries make the world a better place. We can always use more feedback.

Bug Reports

If you come across a bug in Guava, *please* file a bug report. Guava gets used in production at Google, so it's rare for bugs to survive very long without us noticing, but bugs do happen.

Warning us of a bug is possibly the single most valuable contribution you can make to Guava. If you encounter a bug that hasn't already been filed, please file a report with an SSCCE demonstrating the bug.

If you think something *might* be a bug, but you're not sure, ask on StackOverflow or on guava-discuss.

Join Discussions

There's no Guava community without active, public discussions. Chime in with your opinion of feature requests; say what you think about a potential change.

In particular, a lot of what we look for in feature requests is a variety of real-world use cases for a proposed feature. Maybe we can't think of an application for some feature – but you came across one just yesterday, or this new feature would make your current project massively easier.

Join guava-discuss for general discussion about Guava, and guava-issues to follow Guava issues, feature requests, and discussions.

Feature Requests

Filing feature requests is one of the most popular ways to contribute to Guava.

Be aware, though: most feature requests are not accepted, not even if they're suggested by a full-time Guava team member. The feedback from our users has said that they really appreciate Guava's high power-to-weight ratio. It's important to us to keep Guava as easy to pick up and understand as we can. That means boiling features down to compact but powerful abstractions, and controlling feature bloat carefully.

Guava's main yardstick for evaluating features proposed features can be summed up as utility times ubiquity.

Utility: compare with alternatives There is always *some* alternative to adding this new feature to Guava, even if it's just forking Guava yourself.

We want to see that new features have some significant advantage over the alternatives. These advantages can take many forms, but taking the time to discuss them in detail will make it much clearer why this feature should be added to Guava.

What helps the most is to have the following laid out clearly:

1. What are you trying to do?
2. What's the best code you can write to accomplish that using only today's Guava?
3. What would that same code look like if we added your feature?

Having the two approaches to a use case side by side can make it much easier to pick out the differences between them.

Additionally, it's very useful to us if you can provide a "straw API" – what the method signatures would look like, for example, even if the method and class names are still in flux. This can make the feature you're suggesting much clearer to us.

Ubiquity: provide concrete use cases Did you *actually* encounter the need for this feature in a real-world scenario, or is it just a feature that seems like a sensible addition to Guava?

Before new features get added to Guava, we really want to be sure that it's a use case that actually comes up in the real world. We want to hear the real-world use case so the community can discuss and debate whether this feature is actually the *best* way to address the real use case, or whether or not a different abstraction might be appropriate.

It's okay that you can't always give us complete context on a use case. Not all of you are at liberty to discuss the details of what you're working on.

But Guava aims to provide features that are useful across boundaries of projects, companies, or even industries – utilities useful for a sizable proportion of all Java programmers everywhere. If you can give enough detail that any of us can imagine coming across in our own work, that's extremely helpful in studying how broadly useful the feature will be.

Code Contributions

Contributing code is one of the more difficult ways to contribute to Guava, and is almost never what the project really needs most.

New Features

Is there some feature request that you'd like to code up yourself? Is there a feature you asked for yourself that you'd like to code?

Here's how to contribute code for a new feature to Guava. (A "new feature" is any change to Guava that exposes new methods, classes, fields, or interfaces, or changes method signatures.)

1. File a feature request, if you haven't already.
2. Discuss the feature request until it is marked Accepted.
3. Hammer out the API, if necessary. Method and class names can still be in flux, but method signatures (arguments and return type) should be agreed on.
4. Comment on the issue, saying that you'd like to implement it. Ask for a reviewer to volunteer.
5. Sign the Google Individual CLA before sending us any code! (If you're contributing on behalf of your company, the company must have signed the Google Corporate CLA instead.)
6. Implement the change.
7. Get your change through the (difficult) Guava code review process. Expect to spend as much time on documentation and tests as you do on the source code. Don't get discouraged – this always takes many rounds.
8. Your reviewer will import the change into Google's source control system, and your change will get mirrored out in a day or two. Congratulations!

Note: We know it's tempting to submit code before the feature is accepted, but this is not always a good use of your time. First, the community has to discuss whether the feature is a good fit for Guava, then the API gets hammered out, and *then* code happens. Sending in code *won't* help a feature request get accepted.

If someone else has already started implementing a feature, they'll you know when you request a reviewer. Even so, volunteers are always appreciated to help review and discuss new features.

Optimizations, Tests, and Documentation

The overwhelming majority of changes to Guava don't add new features at all. Optimizations, tests, documentation, refactorings – these are all part of making Guava meet the highest standards of code quality and usability.

Contributing improvements in these areas is much easier, and much less of a hassle, than contributing code for new features.

Just email the `guava-discuss` mailing list with a summary of the improvements you'd like to make and why, and ask for a reviewer. If the community agrees that it's a good change to make, code up the change and send it to your reviewer as above. There's no need to write a feature request or anything.

Getting Started

Are you ready to start coding? See this page for instructions on how to get Guava checked out and building, and how to send in code for review.