Contributing

We're always looking for contributions! Here are some ways to participate in Cargo's development:

- By sending feedback to the user or dev mailing lists. The feedback could be about something that does not work, something that could be improved, a feature you'd like to see, etc. Or simply it could be that you're a happy user. Letting us know helps a lot!
- By answering emails from others on the mailing lists.
- By sending code patches. In that case there are a few rules you need to know.
- By spreading the word about Cargo!

Creating and submitting a patch

When you have either completed an issue or just want some feedback on the work you have done, create a patch and attach the patch to the issue in question. We have a couple of guidelines when creating patches:

- Always create the patch from the root of the maven project, i.e. where the pom.xml file is.
- Always provide a single patch and not several diff files. If you're adding new files, svn add them first so that they appear within the single patch file.
- If this was a new piece of work without a JIRA issue, create a JIRA issue for it now.
- Attach the patch to the JIRA issue you were working on (do not paste its content in as a comment though). When adding the patch add a comment to the issue explaining what it does. Shortly after, someone will apply the patch and close the issue.

An example on how to create a patch from the command line:

```
$ svn diff > CARGO-123-something.patch
```

If you are picking up an issue with a existing patch attached to the issue you can apply the patch to your working directory directly from JIRA like this. The `wget` and `patch` commands will only be available if you are on a UNIX platform or using Cygwin on Windows.

```
$ wget -O - -q <URL to the patch from JIRA>
| patch -p0
```

If the patch is in a local file `CARGO-123.patch` and you want to apply that use this command:

```
$ patch -p0 < CARGO-123.patch
```

A couple of notes:

- If you are using another tool for creating patches, make sure that the patch doesn't include absolute paths. Including absolute paths in the patch will make the useless for us as we most likely don't have the same directory structure as you.

- Make sure that you follow our code style.

Patch acceptance criteria

There are a number of criteria that a patch will be judged on:

- Whether it works and does what is intended. This one is probably obvious!
- Whether it fits the spirit of the project. Some patches may be rejected as they take the project in a different direction to that which the current development community has chosen. This is usually discussed on an issue well before a patch is contributed, so if you are unsure, discuss it there or on the mailing lists first. Feel free to continue discussing it (with new justification) if you disagree, or appeal to a wider audience on the mailing lists.
- Whether it contains tests. It is expected that any patches relating to functionality will be accompanied by unit tests and/or integration tests. It is strongly desired (and will be requested) for bug fixes too, but will not be the basis for not applying it. At a bare minimum, the change should not decrease the amount of automated test coverage. As a community, we are focusing on increasing the current coverage, as there are several areas that do not receive automated testing.
- Whether it contains documentation. All new functionality needs to be documented for users, even if it is very rough for someone to expand on later. While rough is acceptable, incomplete is not. As with automated testing, as a community we are striving to increase the current coverage of documentation.

Above all, don't be discouraged. These are the same requirements the current commiters should hold each other to as well.
And remember, your contributions are always welcome!

Coding rules

If you submit a patch you need to follow these rules:

- Copyright your code to Vincent Massol (see license explanations)
- Do not use an @author tag (there was a big discussion on this in Apache land - We need to put the link here)
- Ensure that your code passes the build. Note that the build contains some Checkstyle checks that your code must pass.
- Ensure that you have unit tests and/or integration tests (as part of the existing Cargo test suites)
- Use the same code formatting as the existing code.
- Create a JIRA issue and attach your patch to it.
- Create documentation for what you have added (Ask on the list and you'll get access to Cargo's wiki).
- Add your name on the Credits page.

In addition if you plan to contribute big patches that impact existing code, we recommend discussing it on the mailing list first.

Thanks!