Contributing to stxx

by Don Brown

Table of contents

1. Introduction

The stxx project is an Open Source volunteer project released under a very open license. This means there are many ways to contribute to the project - either with direct participation (coding, documenting, answering questions, proposing ideas, reporting bugs, suggesting bug-fixes, etc..) or by resource donations (money, time, publicity, hardware, software, conference presentations, speeches, etc...).

To begin with, we suggest you to subscribe to the stxx mailing list. Listen-in for a while, to hear how others make contributions.

You can get your local working copy of the latest and greatest code (which you find in the stxx module in the CVS code repository. Review the todo list, choose a task (or perhaps you have noticed something that needs patching). Make the changes, do the testing, generate a patch, and post to the mailing list or email the committers.

2. Help Wanted Here

The rest of this document is mainly about contributing new or improved code and/or documentation, but we would also be glad to have extra help in any of the following areas:

- Answering questions on the mailing list.
- Testing stxx (especially its less-frequently-used features) on various configurations and reporting back.
- Debugging - producing reproduceable test cases and/or finding causes of bugs.
- Specifying/analysing/designing new features - and beyond.
- ... and there is just one other thing - don't forget to tell everyone who asks, how great stxx is! ;-) The more people that know about and start to use stxx, the larger the pool of potential contributors there will be.

3. CVS Overview

This is an overview of how to use CVS to participate in stxx development. Do not be afraid - you cannot accidently destroy the actual code repository, because you are working with a local copy as an anonymous user. Therefore, you do not have the system permissions to change anything. You can only update your local repository and compare your revisions with the real repository.

(Further general CVS usage information is at www.cvshome.org and your local `info cvs` pages or `man cvs` pages or user documentation.)

For example, using the commandline version of cvs, to checkout a copy of stxx you would enter:

```
cvs -d
:pserver:anonymous@cvs.stxx.sourceforge.net:/cvsroot/stxx co
stxx
```

4. CVS Committer with Secure Shell access

After a developer has consistently provided contributions (code, documentation and discussion), then the rest of the dev community may vote to grant this developer commit access to CVS.

You will need secure access to the repository to be able to commit patches. Here are some resources that help to get your machine configured to use the repository over SSH.

- The CVS Book
- www.cvshome.org
- - See the bottom of the page for links to tips for UNIX and Windows. Even if you are on UNIX, the Windows page will also help.

5. Contribution Notes and Tips

This is a collection of tips for contributing to the project in a manner that is productive for all parties.

- Every contribution is worthwhile. Even if the ensuing discussion proves it to be off-beam, then it may jog ideas for other people.
- Use sensible and concise email subject headings. Search engines, and humans trying to browse a voluminous list, will respond favourably to a descriptive title.
- Start new threads with new Subject for new topics, rather than reusing the previous Subject line.
- Keep each topic focused. If some new topic arises then start a new discussion. This leaves the original topic to continue uncluttered.
- Whenever you decide to start a new topic, then start with a fresh new email message window. Do not use the "Reply to" button, because threaded mail-readers get confused (they utilise the `In-reply-to` header). If so, then your new topic will get lost in the previous thread and go unanswered.
- Prepend your email subject line with a marker when that is appropriate, e.g. `[Patch]`, `[Proposal]`, `[RT]` (Random Thought which quickly blossom into research topics :-), `[STATUS]` (development status of a certain facility).
- When making changes to XML documentation, or any XML document for that matter, use a validating parser (one that is tried and true is OpenSP/onsgmls). This procedure will detect errors without having to go through the whole `build docs` process to find them.

Do not expect Forrest or the build system to detect the validation errors for you - they can do it, but that is not their purpose. (Anyway, nsgmls validation error messages are more informative.)

- Remember that most people are participating in development on a volunteer basis and in their "spare time". These enthusiasts will attempt to respond to issues. It may take a little while to get your answers.
- Research your topic thoroughly before beginning to discuss a new development issue. Search and browse through the email archives - your issue may have been discussed before. Do not just perceive a problem and then rush out with a question - instead, delve.
- Try to at least offer a partial solution and not just a problem statement.
- Take the time to clearly explain your issue and write a concise email message. Less confusion facilitates fast and complete resolution.
- Do not bother to send an email reply that simply says "thanks". When the issue is resolved, that is the finish - end of thread. Reduce clutter.
- You would usually do any development work against the HEAD branch of CVS.
- When sending a patch, you usually do not need to worry about which CVS branch it should be applied to. The maintainers of the repository will decide.
- If an issue starts to get bogged down in list discussion, then it may be appropriate to go into private off-list discussion with a few interested other people. Spare the list from the gory details. Report a summary back to the list to finalise the thread.
- Become familiar with the mailing lists. As you browse and search, you will see the way other people do things. Follow the leading examples.

This page is based off the Contributing to Forrest page