Contributing to Forrest

1. Introduction

The Forrest Project is an Open Source volunteer project released under a very liberal 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 Forrest mailing lists (follow the link for information on how to subscribe and to access the mail list archives). 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 Forrest module in the SVN 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 dev mailing list. (Do not worry - the process is easy and explained below.)

Document writers are usually the most wanted people so if you like to help but you're not familiar with the innermost technical details, don't worry: we have work for you!

2. Help Wanted Here

We would be glad to have extra help in any of the following areas:

- Assisting to improve documentation.
- Testing Forrest (especially its less-frequently-used features) on various configurations and reporting back.
- Debugging - producing reproduceable test cases and/or finding causes of bugs. Some known bugs are informally listed on To Do, and some are recorded as issues (see explanation below).
- Providing new use-cases and requirements. If you think that Forrest does not quite meet your needs then tell us about it.
- Specifying/analysing/designing new features - and beyond. If you wish to get further involved with this, please join the `forrest-dev` mailing list, install and try out Forrest and read some of the mail archives. You should have a reasonable fluency in XML technologies, some Java and Ant skills, and a basic understanding of the Forrest architecture - don't just say "it should have XYZ" without reading anything first - because chances are, somebody has already thought of that feature!)
- Packaging easy-to-install packages (such as RPMs) for the myriad of possible configurations out there. (The project does not maintain anything but the basic `.zip` and `.tar.gz` packages, but anyone is welcome to build their own specific packages and announce them on the `forrest-dev` list)
- ... and there is just one other thing - don't forget to tell everyone who asks, how great Forrest is! The more people that know about and start to use Forrest, the larger the pool of potential contributors there will be.

3. SVN Usage

An overview of how to use Subversion (SVN) to participate in Forrest 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. The Building Forrest document explains.

4. SVN Committer with Secure Shell access

After a developer has consistently provided contributions (code, documentation and discussion) and demonstrated

committment, then the rest of the dev community may vote to grant this developer commit access to the Subversion repository.

You will need secure access to the repository to be able to commit patches. Commits to the SVN repository must use the https: protocol. If you already have the codebase checked out via the http: protocol, then the following command will convert it.

```
svn sw https://svn.apache.org/repos/asf/forrest/trunk
```

5. Procedure for Raising Development Issues

There are two methods for discussing development and submitting patches. So that everyone can be productive, it is important to know which method is appropriate for a certain situation and how to go about it without confusion. This section explains when to use the developer mailing list and the issue tracker.

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. Prepare your post clearly and concisely.

Most issues will be discovered, resolved, and then patched quickly via the developer mailing list. Larger issues, and ones that are not yet fully understood or are hard to solve, are destined for the issue tracker.

Experienced developers use the issue tracker directly, as they are very sure when they have found a bug and when not. However, less experienced users should first discuss it on the user or developer mailing list (as appropriate). Impatient people always enter everything into the issue tracker without caring if it is a bug of Forrest or their own installation/configuration mistake - please do not do this.

As a rule-of-thumb, discuss an issue on the developers mailing list first to work out any details. After it is confirmed to be worthwhile, and you are clear about it, then submit the bug description or patch via Bug Tracking.

Perhaps you do not get any answer on your first reply, so just post it again until you get one. (But please not every hour - allow a few days for the list to deal with it.) Do not be impatient - remember that the whole world is busy, not just you. Bear in mind that other countries will have holidays at different times to your country and that they are in different time zones. You might also consider rewriting your initial posting - perhaps it was not clear enough and the readers eyes glazed over.

6. 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 XML editor. Here is some assistance with editor configuration.
- 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 trunk of SVN.
- When sending a patch, you usually do not need to worry about which SVN branch it should be applied to. The maintainers of the repository will decide.
- Keep all project-related discussion on the mailing list. It is much better to utilise the wider audience, rather than to break off into private discussion groups. You never know who else will have the answer to your issues, and anyway other people are interested in the outcome.
- Become familiar with the mailing lists. As you browse and search, you will see the way other people do things. Follow the leading examples.