GettingYourPatchCommitted

How to Submit A Patch

Found a bug? Got an idea for an enhancement? Here's a quick set of guidelines to help you in getting your patch committed.

1. First, before you write the patch - email velocity-user@jakarta.apache.org to see if anyone else has solved the problem you are facing. If you want to discuss the internals of Velocity (or your idea as to how to solve the problem), email velocity-dev@jakarta.apache.org.

2. Second, download the latest source code from the Subversion repository and write your patch. Create the patch by generating a diff with "svn diff". If you use a different diff program to generate the diff, be sure to specify unified diff format (typically -u option).

3. Third, test the patch by writing unit tests with JUnit. Look at the existing unit tests for examples of how to do this.

4. Submit a JIRA entry. If you include a patch, add the keyword PatchAvailable. Please include a clear specification of the problem the patch is supposed to solve, how to replicate it (for a bug), and the approach you've taken or recommend to take in order to solve it.

Criteria for Committing

An enhancement or bug fix is likely to be accepted if it:

- Includes a patch.
- Includes unit tests for new functionality.
- States that the submitter has run "ant test" and the patch has passed all existing tests.
- Follows Velocity Coding Standards.
- Is useful to multiple members of the Velocity community.
- The submitter is willing to support it in the future
- Is consistent with the general functionality of Velocity.
- Is backwards-compatible

The last two points are a little broad. Here's a bit more detail.

Consistency

Velocity has a fairly narrow focus. In a nutshell, it's a light-weight library designed to create text output based on a text with markup statements (Velocity Template Language). Anything that's outside of that scope is probably a poor candidate for a Velocity enhancement.

If you feel that your enhancement would be useful for Velocity developers but is not a core functionality (for example, a library to send Velocity-formatted email, or a Tool object to be used in the Velocity Context to handle i18n), consider submitting to the VelocityTools subproject. The submission process is similar, except preface the Bugzilla issue title with [Velocity-Tools].

Before submitting your patch, look at similar examples in the Velocity source code. For example, if you are writing a resource loader, look at the source for other resource loaders and create yours in a similar manner.

Finally, consider whether your patch duplicates existing functionality. If the problem you are trying to solve can be accomplished in a different way, consider whether your patch might not be needed. (or, you might be better off submitting a patch that modifies existing code rather than adding new code).

Backwards-Compatibility

For most changes to Velocity, the committers will look to see if the patch is backwards-compatible. Historically, the committers have been very conservative on this issue. The basic idea is that users with existing installations will be able to use the updated version of Velocity with no changes. (This applies to both the templates and the Velocity library itself). There's been talk of a 2.0 version for which this restriction might be eased. See the RoadMap for discussion.

In addition, the goal is for Velocity to compile and run under JDK 1.3.

Alternatives to Patching Velocity

If you would like to see changes in the functionality of Velocity for your application but do not think your change is suitable for a patch, you might consider creating custom plugins. Specific customizations you can do without changing the Velocity source code include:

- Writing your own resource loader to load templates in a custom manner
- Writing your own custom directive to provide custom VTL syntax
- Writing your own introspector to call methods following your specific rules
- Writing an event handler to modify basic Velocity behaviors.

More info can be found in the Developer's Guide or in the HackingVelocity code examples.