UsingBugzilla

Opening Bugs For Development

The general practice is to open a bug when some task needs to be done, even if there is no "bug" involved; it's really a "task tracker", not a "bug tracker". The idea is to provide a "thread" of discussion and a place to track development.

That way, when a developer is later reading the code, and wondering why a certain database design is used (for example), they can look up a bug number referenced in a comment, and see the entire discussion thread for that design decision.

BTW, this is different from other ASF projects, who prefer discussion on mailing lists and bugs on the bugzilla. That's OK, we're different!

Bugzilla also offers a more persistent way of storing things like patches and votes; these are less likely to get lost than they would be if they were posted on the mailing lists. Plugins that are likely to be frequently downloaded by third parties and users, should probably go on the wiki – e.g. the CustomPlugins page – but stuff that is intended for inclusion in the main distro, Bugzilla is best.

Current Milestones

(consider this an example of how the milestones are used, assuming that in this example we are somewhere between the 3.0.x and 3.1.0 releases. there's no need to update this example for every future release...)

- 3.1.0: next major release (also see ReleaseGoals)
- 3.1.1: next minor release (bug fixes and minor things that we'd like to get fixed, but don't want to hold up 3.1.0 for them)
- 3.2.0: next major release
- Undefined: no milestone set
- Future: maybe at some point in the distant future

Bug Triage

Here is how you handle bug triage:

- mark duplicated bugs as duplicate
- try to confirm bugs
- test submitted rules with promise (see AutoMassChecks)
- ask users for additional information as needed (sooner is better if you want a response), you can use the *moreinfo* keyword to denote bugs awaiting a response
- set the severity (how bad does it affect things, is it an enhancement request, etc.)
- close bugs already fixed (when possible, this is mostly done by developers, but sometimes it's obvious or easy to tell)
- close bugs that are invalid.
- if you can reproduce a bug under the current svn, update the Version flag if not already set to svn.
- when any of the above is being done, and you're not yet ready to set a milestone or close a bug, then flag the bug with a keyword "triage". This will indicate to other triagers that the bug is already undergoing triage.

Severity describes the impact of the bug, and is often set by the submitter. The only really important value here is 'blocker' or 'critical', both of which indicate that the bug should block further releases until it's fixed.

If the developer doesn't agree with the Sev setting, they can go ahead and change it – but it doesn't really matter anyway, since the Priority is what actually affects the operation of Bugzilla (sort order etc.). In theory, the developer working on the bug can use Priority to indicate how important they think the bug is. In practice, we don't use it much.

We do use the milestones a lot. Developers are always going to tweak those, but I think we could come up with a procedure to allow those to be set fairly accurately in a first pass triage. Something like:

```
if (bug)
   set severity appropriately if needed
   if (bug affecting current svn head)
      assign to next release off of svn head
   elsif (bug affecting current stable release)
      if (bug affects both svn head and current stable)
         # I'm still not entirely happy with how we handle this case...
         assign to next release off of svn head, make note in bug
      else
         assign to next stable release

else
   set severity to "enhancement"
   if (pie-in-the-sky)
      assign to "Future"
   else
      assign to next major release
```

Requests for new features should have a Sev of 'enhancement', and optionally have a title beginning with 'RFE:'.

When Taking Bugs...

When taking a bug in Bugzilla (i.e. assigning it to yourself), please go ahead and make sure dev@spamassassin.apache.org is in the CC list. That way bug discussion is out in the open.

Tagging Bugs

Generally in addition to using Keywords, we use the "Status Whiteboard" field to give a quick overview of what the status of the bug is.

Bugs that need peer review are usually tagged with [review] in the subject as this makes it more obvious when reading the dev list that a bug needs review. This should maybe be a keyword also?

Nah, leave it a subject tag, since that appears in mail. no need to have it twice! – JustinMason

We also use 'needs N votes' in the Status Whiteboard field. However, as far as I can see whenever we're in a situation that votes are needed, the voting population are never reading this anyway, so I think it may be superfluous. 😳 – JustinMason

Committing Fixes For Bugs

When you commit a fix for some bug, the number of the bug should be stated in the commit message.

JustinMason: a tip on referring to bugs – the convention is "bug NNNN: blah blah" simply because Bugzilla has the smarts to automatically turn "bug NNNN" into a hyperlink, and it's easily greppable.

JustinMason: also 'bug NNNN comment NN' gets turned into a hyperlink to a comment.

Categories For Closing Bugs

- FIXED: it's now fixed in SVN – this is usually used when a commiter checks in a fix for the bug.
- INVALID: it's not actually a bug report, or not something related to SpamAssassin – "SpamAssassin doesn't work!" or "I can't install SpamAssassin"
- WONTFIX: the report proposes something that we do not want to fix either for technical or philosophical reasons – "SpamAssassin should do anti-virus scanning"
- WORKSFORME: it's not reproducible, or the reported behaviour seems to be as designed
- LATER: this might be something we should look at in the far future, but for now its infeasible or undesirable – seldom used
- REMIND: we don't generally use this

Removing Spam Bugs

Deleting a bug that is spam can only be done by someone who has Administrator access to our Bugzilla. See RemovingBugzillaSpam.