CommonRuby - Feature #8271

Proposal for moving to a more visible, formal process for feature requests

04/16/2013 05:35 AM - headius (Charles Nutter)

Status:	Assigned
Priority:	Normal
Assignee:	matz (Yukihiro Matsumoto)
Target version:	

Description

Proposal for moving to a more visible, formal process for feature requests.

1. Introduction

In order to make it clear that an issue or change to MRI is a visible feature change all implementations will need to consider implementing, I propose that we move all feature requests into a separate Redmine project. I see the following benefits to this arrangement:

- Features are always in one place. One-stop-shopping for implementers to track changes to "Ruby" that are not specific to MRI.
- No confusion about where feature requests should be filed. Currently, people usually file feature requests against "trunk", but sometimes against version-specific projects. It's also valid to say that a feature improvement or clarification is not specific to trunk. Tracking features separate from "trunk" and version-specific Redmine projects keeps the process localized to one Redmine project.
- Ability to add fields to "feature" issues that do not have relevance for "bugs". For example, bugs do not usually need approval from matz, but features could have an "approved by matz" field. We could also have other metadata tracking other implementations, such as "approved by implementations" or "affects implementations" with drop-downs for known impls. One-stop-shopping to know whether a given impl is affected and/or has agreed to add the feature.
- More visible process for folks in the community that can't follow the current process or don't believe there's a process in place.

I propose that the project be called CommonRuby (already created and under some use) and be a top-level entry on bugs.ruby-lang.org.

1. Processes

For issues that are obviously new features (i.e. user knows to select "feature" in the current tracker), issues would be filed directly in CommonRuby. Discussion proceeds exactly as the current process, perhaps with additional issue fields added that allow tracking matz approval, etc, as stated in §1.

Issues that are approved for a Ruby version will have fields/metadata to indicate at which version the feature is available. This may mean specifying multiple releases if, for example, 2.0.1 and 1.9.3p400 would both see a feature added (just saying 1.9.3p400 is insufficient since the feature does not exist in 2.0.0. This avoids having to track features through the backport process to know if there are multiple releases that contain the feature.

For issues that start out as bugs, but later become features or feature changes, those issues would be tranferred into CommonRuby at the point where it's obvious they're feature-related.

1. Detriments

Benefits are stated in the introduction above.

Possible detriments with mitigation:

- Confusion by users about where to file features.

This would be mitigated by adding more information to bug-related home pages about the CommonRuby project. The "feature" value in current "trunk" project could either be removed (after migrating features to CommonRuby) or modified to error/warn or switch the issue to CommonRuby programmatically.

- More complex process.

I believe this process is no more complicated than the current process. It also makes the process of evolving "common Ruby" more isolated from MRI development and may make it easier for users to track that evolution.

1. Further justification

A lot of noise has been made over the past several months about Ruby lacking a process for new and changing features. The design process proposed by Brian Shirai (née Ford) gained some measure of popularity, but requires a complete retooling and reworking of current processes, making it infeasible for short-term implementation. Other process-change proposals have been kicked around on ruby-core, but the truth is that there *is* a current process, even if it's not particularly visible. By implementing my proposal, the process would become more obvious and transparent without major impact to MRI's development or Ruby's evolutionary processes.

1. Prior art

The PEP (Python Enhancement Proposal) and JSR (Java Specification Request) processes are partial inspiration for this proposal. The latter governs all visible feature changes to Python independent of bug reports to the main "CPython" implementation. The latter governs (through a heavy and overly-strict process) changes to "Java" independent of individual JVM implementations. Both processes have been very successful at isolating spec changes from implementation changes, although the JSR process tends to move very slowly and be less transparent than it should be.

1. Conclusion

Ruby does not lack a process for adding or changing features, but it does lack visibility into that process and in many cases fails to provide tools to non-MRI implementations to participate. Moving feature requests and discussion to a CommonRuby project independent of MRI will make the process more transparent and easier to follow (for users and implementers) while having minimal impact on the current process.

Subtasks:

Feature # 8272: Transfer feature tracking to CommonRuby	**Open**

Related issues:

Related to CommonRuby - Feature #7549: A Ruby Design Process	**Rejected**	**12/12/2012**

History

#1 - 04/16/2013 05:37 AM - headius (Charles Nutter)

For reference, the current CommonRuby project in Redmine is here: https://bugs.ruby-lang.org/projects/common-ruby

#2 - 04/16/2013 05:40 AM - headius (Charles Nutter)

A correction...

The latter governs all visible feature changes to Python independent of bug reports to the main "CPython" implementation.

should read

The former governs all visible feature changes to Python independent of bug reports to the main "CPython" implementation.

#3 - 04/16/2013 06:17 AM - enebo (Thomas Enebo)

I wonder if issues for Ruby can have a simple front-end wizard where new users will get funneled to the correct system. Once you are experienced you can go straight to the proper redmine project but new users will find the issues link on ruby-lang and get their handheld so they go to the proper place?

My second hope is that mis-reported issues should be easily converted to CommonRuby from ruby-trunk. From experience users do not read documentation and submit to the first thing they find.

#4 - 04/16/2013 09:44 AM - naruse (Yui NARUSE)

- Status changed from Open to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

- No confusion about where feature requests should be filed. Currently, people usually file feature requests against "trunk", but sometimes against version-specific projects. It's also valid to say that a feature improvement or clarification is not specific to trunk. Tracking features separate from "trunk" and version-specific Redmine projects keeps the process localized to one Redmine project.

People can't make feature request other than trunk.
There's only backport request in version-specific projects, and backports can't include new features on current policy.

- Ability to add fields to "feature" issues that do not have relevance for "bugs". For example, bugs do not usually need approval from matz, but features could have an "approved by matz" field. We could also have other metadata tracking other implementations, such as "approved by implementations" or "affects implementations" with drop-downs for known impls. One-stop-shopping to know whether a given impl is affected and/or has agreed to add the feature.

It is already available. You can see "ruby -v" field is only available on "Bug" tracker.

As I asked in [ruby-core:54246], how about bundled libraries?
Note that like rubygems many of them are owned by other than matz,
and some of them's main repository is other than ruby's.

#5 - 04/16/2013 01:49 PM - headius (Charles Nutter)

naruse (Yui NARUSE) wrote:

> People can't make feature request other than trunk.
> There's only backport request in version-specific projects, and backports can't include new features on current policy.

Sorry, I think I was confused by the old Ruby 1.8 project, which did have "Feature".

> - Ability to add fields to "feature" issues that do not have relevance for "bugs". For example, bugs do not usually need approval from matz, but features could have an "approved by matz" field. We could also have other metadata tracking other implementations, such as "approved by implementations" or "affects implementations" with drop-downs for known impls. One-stop-shopping to know whether a given impl is affected and/or has agreed to add the feature.

> It is already available. You can see "ruby -v" field is only available on "Bug" tracker.

I stand corrected.

Perhaps my objection here is more semantic... why would features for "Ruby" be filed against MRI "trunk", when they affect all implementations equally? It comes back to my point about tracking Ruby features in JRuby's tracker or Rubinius's tracker. At least a separate project in Redmine could be in spirit shared by all, rather than owned by MRI and sitting alongside bugs that other implementations probably don't care about.

> As I asked in [ruby-core:54246], how about bundled libraries?
> Note that like rubygems many of them are owned by other than matz,
> and some of them's main repository is other than ruby's.

As now, feature requests for those libraries would go to through those libraries' processes. Nothing changes.

#6 - 04/16/2013 01:50 PM - headius (Charles Nutter)

I should say...if feature requests for bundled libraries are currently marked as "feature" in trunk, they'd just be in CommonRuby now. If they're not, they would not be.

#7 - 04/16/2013 03:29 PM - duerst (Martin Dürst)

On 2013/04/16 5:35, headius (Charles Nutter) wrote:

> Issue #8271 has been reported by headius (Charles Nutter).
>
> ---
>
> Feature #8271: Proposal for moving to a more visible, formal process for feature requests
> https://bugs.ruby-lang.org/issues/8271
>
> Proposal for moving to a more visible, formal process for feature requests.

I wanted to oppose a more formal process because I think it's
unnecessary overkill, but reading the proposal, I can't really see much
in terms of more formality, so I think the title of this proposal should
be changed to reflect that. (Sorry if I missed something.)

> 1. Introduction
>
> In order to make it clear that an issue or change to MRI is a visible feature change all implementations will need to consider implementing, I propose that we move all feature requests into a separate Redmine project. I see the following benefits to this arrangement:
>
> - Features are always in one place. One-stop-shopping for implementers to track changes to "Ruby" that are not specific to MRI.

That was the case before we had the CommonRuby project, as Yui has shown.

> - No confusion about where feature requests should be filed. Currently, people usually file feature requests against "trunk", but sometimes against version-specific projects. It's also valid to say that a feature improvement or clarification is not specific to trunk. Tracking features separate from "trunk" and version-specific Redmine projects keeps the process localized to one Redmine project.

Comparing

trunk (bugs and features)
2.0.0 (backport)
1.9.3 (backport)
1.9.2 (backport)
...

and

New feature
Bug trunk
2.0.0 (backport)
1.9.3 (backport)
1.9.2 (backport)
...

it's indeed quite possible that the later is easier to understand and
follow.

> Issues that are approved for a Ruby version will have fields/metadata to indicate at which version the feature is available. This may mean specifying multiple releases if, for example, 2.0.1 and 1.9.3p400 would both see a feature added (just saying 1.9.3p400 is insufficient since the feature does not exist in 2.0.0. This avoids having to track features through the backport process to know if there are multiple releases that contain the feature.

I think this is only feasible if it can be automated. One way to
automate it is to add feature numbers to tests, but I'm not sure if this
will work well.

> 1. Detriments
>
> Possible detriments with mitigation:
>
> - Confusion by users about where to file features.
>
> This would be mitigated by adding more information to bug-related home pages about the CommonRuby project. The "feature" value in current "trunk" project could either be removed (after migrating features to CommonRuby) or modified to error/warn or switch the issue to CommonRuby programmatically.

I agree that this is mostly a documentation issue.

> - More complex process.
>
> I believe this process is no more complicated than the current process.

I agree. It's mainly about people getting used, and if it's possible to
move issues around between projects, then it's not too bad if there are
a few mistakes from time to time.

> 1. Further justification
>
> A lot of noise has been made over the past several months about Ruby lacking a process for new and changing features.

I'd rather prefer Ruby to be the best language with a "lacking" process
than the language with the best process.

> The design process proposed by Brian Shirai (née Ford)

[That should be "(né Ford)" because the former would be grammatically
female (see http://en.wikipedia.org/wiki/Name_at_birth).]

> gained some measure of popularity,

A very modest measure as far as I understand, and quite a bit of
well-founded opposition, too.

> but requires a complete retooling and reworking of current processes, making it infeasible for short-term implementation.

And also has some very serious shortcomings, such as "specify everything completely before implementing anything", which makes it infeasible even in the long term.

Regards, Martin.

#8 - 04/16/2013 03:41 PM - headius (Charles Nutter)

duerst (Martin Dürst) wrote:

> I wanted to oppose a more formal process because I think it's unnecessary overkill, but reading the proposal, I can't really see much in terms of more formality, so I think the title of this proposal should be changed to reflect that. (Sorry if I missed something.)

I agree. This is more a cosmetic change. But I think it's an important one.

It also feels to me like it would be easier to clean up gaps in the current process (not strictly formalize, but tidy up) if we had a separate feature tracker.

> That was the case before we had the CommonRuby project, as Yui has shown.

Correct. They're just interspersed with bugs that are MRI-specific, making it harder to know which issues alternative implementers should track.

> Comparing
>
> trunk (bugs and features)
> 2.0.0 (backport)
> 1.9.3 (backport)
> 1.9.2 (backport)
> ...
>
> and
>
> New feature
> Bug trunk
> 2.0.0 (backport)
> 1.9.3 (backport)
> 1.9.2 (backport)
> ...
>
> it's indeed quite possible that the later is easier to understand and follow.

Indeed. The former throws features in alongside MRI-specific details. That's wrong, in my mind.

> I think this is only feasible if it can be automated. One way to automate it is to add feature numbers to tests, but I'm not sure if this will work well.

The fact that all items in Redmine are numbered the same is possibly confusing here. I don't imagine it would be hard to add a separate "Feature ID" field to redmine, allowing a simple report of features that have been accepted.

There are many JSRs and PEPs that never see the light of day, but overall the JSR and PEP processes have their own numbers assigned to things.

> I believe this process is no more complicated than the current process.

> I agree. It's mainly about people getting used, and if it's possible to move issues around between projects, then it's not too bad if there are a few mistakes from time to time.

That is indeed a key point. The fact that Redmine supports different bug schemas in the *same* project makes me believe it should be just as trivial to move a bug to CommonRuby (as a feature) as it is to flip the feature toggle now.

Again, mostly a cosmetic change, but an important one: we're making it known that Ruby feature changes are a top-level concern...not a mixed-priorty concern stirred in with MRI bugs.

> 1. Further justification

> A lot of noise has been made over the past several months about Ruby lacking a process for new and changing features.

I'd rather prefer Ruby to be the best language with a "lacking" process
than the language with the best process.

Agreed. I feel like the current process works...it's just hard to follow with MRI bugs and Ruby features in the same place.

> The design process proposed by Brian Shirai (née Ford)

[That should be "(né Ford)" because the former would be grammatically
female (see http://en.wikipedia.org/wiki/Name_at_birth).]

Thanks! I went out on a limb using that word for the first time.

#9 - 04/16/2013 04:37 PM - naruse (Yui NARUSE)

I still don't understand why headius want to do this and it is reasonable, but I understand what he want to do.
So I can try to do it.

How do you think, matz?

#10 - 04/16/2013 09:02 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

Just for the record, I'm +1 and I'm not an implementer. I'm only interested in the Ruby language discussion itself and the bug reports are noise to me as well. I'd love if it would be possible for me to follow only issues requesting changes to the Ruby language. Reading a list like that would take so much less time than reading ruby-core currently does...

I guess there is more people like me who would like to be involved in the Ruby language changing but aren't willing to take time reading all bugs discussions in the ruby-core mailing list. Unfortunately I'm also aware that if I do so, I would end up missing some discussions sent directly to that list instead of through a ticket... But maybe I could simply add some filter to delete all e-mails whose subjects starts with Bug...

#11 - 04/17/2013 12:13 AM - headius (Charles Nutter)

naruse (Yui NARUSE): I will try to summarize what implementation of this proposal will involve:

Required changes:

- CommonRuby becomes the project for feature requests.
- It moves to top-level on bugs.ruby-lang.org.
- Documentation about how to file feature requests points users to CommonRuby.
- Necessary plumbing to make it easy to transfer "bug" issues from ruby-trunk to CommonRuby if they become features.

Possible changes:

- Existing (open?) features from ruby-trunk migrate to CommonRuby.
- New attempts to file feature requests against ruby-trunk redirect users to CommonRuby or automatically get filed in CommonRuby.

Future "to be discussed" changes:

- More/better metadata on CommonRuby issues to track matz approval, implementer approval, spec/test coverage, implementation completeness (done in JRuby? done in Rubinius?), etc.
- Reporting interface against CommonRuby similar to top-level PEP pages. An easy way for folks to track (rss/atom/simple web page) features as they're filed without going to Redmine.

#12 - 04/17/2013 03:21 AM - naruse (Yui NARUSE)

headius (Charles Nutter) wrote:

> naruse (Yui NARUSE): I will try to summarize what implementation of this proposal will involve:
>
> Required changes:
>
> - CommonRuby becomes the project for feature requests.
> - It moves to top-level on bugs.ruby-lang.org.
> - Documentation about how to file feature requests points users to CommonRuby.

Whether this documentation is a content of www.ruby-lang.org or wiki page of CommonRuby?
And could you write it?

- Necessary plumbing to make it easy to transfer "bug" issues from ruby-trunk to CommonRuby if they become features.

You can already transfer it.

> Possible changes:
>
> - Existing (open?) features from ruby-trunk migrate to CommonRuby.

I think it should do.

> - New attempts to file feature requests against ruby-trunk redirect users to CommonRuby or automatically get filed in CommonRuby.

Redirection requires customizing Redmine, so I hesitate it.
So it should be done by removing feature tracker from ruby-trunk project.
But this depends you allow MRI-specific features exist on CommonRuby.

> Future "to be discussed" changes:
>
> - More/better metadata on CommonRuby issues to track matz approval, implementer approval, spec/test coverage, implementation completeness (done in JRuby? done in Rubinius?), etc.

If you add such column, you should describe status path of proposal like PEP 1.
http://www.python.org/dev/peps/pep-0001/

Anyway initially we can write those status as text in the ticket and editing it when the status is changed.
If we want to filter with such status, then add such fields.

> - Reporting interface against CommonRuby similar to top-level PEP pages. An easy way for folks to track (rss/atom/simple web page) features as they're filed without going to Redmine.

Do you know Redmine already has atom feeds?
https://bugs.ruby-lang.org/projects/common-ruby/issues.atom

Or you can make such page with Redmine REST API.

#13 - 04/17/2013 03:33 AM - headius (Charles Nutter)

naruse (Yui NARUSE) wrote:

> headius (Charles Nutter) wrote:
>
> > - Documentation about how to file feature requests points users to CommonRuby.
> >
> > Whether this documentation is a content of www.ruby-lang.org or wiki page of CommonRuby?
> > And could you write it?
>
> Yes, I can write it. I'll try to figure out all appropriate places and let you know if I don't have access.
>
> I'll write *something* in CommonRuby wiki page that describes what we've talked about here (targeted at normal users), but help would be appreciated.
>
> > - Necessary plumbing to make it easy to transfer "bug" issues from ruby-trunk to CommonRuby if they become features.
> >
> > You can already transfer it.
>
> Confirmed...I transferred one of my features earlier today.
>
> > - Existing (open?) features from ruby-trunk migrate to CommonRuby.
> >
> > I think it should do.
>
> Agreed.
>
> > - New attempts to file feature requests against ruby-trunk redirect users to CommonRuby or automatically get filed in CommonRuby.
> >
> > Redirection requires customizing Redmine, so I hesitate it.

> So it should be done by removing feature tracker from ruby-trunk project.
> But this depends you allow MRI-specific features exist on CommonRuby.

Hmm...that's a good point. I think it's ok to leave it so MRI features can still live in ruby-trunk. I recognize that we'll get CommonRuby features filed as ruby-trunk features forever...but I don't see it as a big problem.

I think we *will* need to audit all current open features before migrating to ensure we don't dump MRI-specific features into CommonRuby. I will help with this, since there's a lot of them. Maybe we can clean up old ones that are not relevant anymore too.

> Future "to be discussed" changes:
>
> - More/better metadata on CommonRuby issues to track matz approval, implementer approval, spec/test coverage, implementation completeness (done in JRuby? done in Rubinius?), etc.

> If you add such column, you should describe status path of proposal like PEP 1.
> http://www.python.org/dev/peps/pep-0001/
>
> Anyway initially we can write those status as text in the ticket and editing it when the status is changed.
> If we want to filter with such status, then add such fields.

Ok. We'll take this one step at a time.

> - Reporting interface against CommonRuby similar to top-level PEP pages. An easy way for folks to track (rss/atom/simple web page) features as they're filed without going to Redmine.

> Do you know Redmine already has atom feeds?
> https://bugs.ruby-lang.org/projects/common-ruby/issues.atom
>
> Or you can make such page with Redmine REST API.

That's great to hear! I would like to see a better/prettier/friendlier report page, but this is another case we can take slowly. I also don't do web dev, so I'd need help here :-)