Ruby master - Feature #6875

Make test/unit default gem

08/15/2012 09:07 PM - kou (Kouhei Sutou)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>sorah (Sorah Fukumori)</td>
</tr>
<tr>
<td>Target version:</td>
<td>2.0.0</td>
</tr>
</tbody>
</table>

**Description**

test/unitをdefault gemにするパッチです。
http://bugs.ruby-lang.org/projects/ruby/wiki/StdlibGem に従ってバージョンは2.0.0.0にしています。

私はtest-unitというgemを作っています。test-unit gemがインストールされるとバンドルされているtest/unitよりも優先してtest-unit gemの方が使われると嬉しいのでこのパッチを書きました。

**Associated revisions**

Revision ed931252 - 08/31/2012 12:50 PM - kou (Kouhei Sutou)

- lib/test/unit/test-unit.gemspec: Make test/unit default gem. [Feature #6875] [ruby-dev:46051]
  git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@36872 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 36872 - 08/31/2012 12:50 PM - kou (Kouhei Sutou)

- lib/test/unit/test-unit.gemspec: Make test/unit default gem. [Feature #6875] [ruby-dev:46051]

Revision 36872 - 08/31/2012 12:50 PM - kou (Kouhei Sutou)

- lib/test/unit/test-unit.gemspec: Make test/unit default gem. [Feature #6875] [ruby-dev:46051]

Revision 36872 - 08/31/2012 12:50 PM - kou (Kouhei Sutou)

- lib/test/unit/test-unit.gemspec: Make test/unit default gem. [Feature #6875] [ruby-dev:46051]

Revision 36872 - 08/31/2012 12:50 PM - kou (Kouhei Sutou)

- lib/test/unit/test-unit.gemspec: Make test/unit default gem. [Feature #6875] [ruby-dev:46051]

Revision 36872 - 08/31/2012 12:50 PM - kou (Kouhei Sutou)

- lib/test/unit/test-unit.gemspec: Make test/unit default gem. [Feature #6875] [ruby-dev:46051]

Revision 36872 - 08/31/2012 12:50 PM - kou (Kouhei Sutou)

- lib/test/unit/test-unit.gemspec: Make test/unit default gem. [Feature #6875] [ruby-dev:46051]

**History**

#1 - 08/15/2012 09:08 PM - kou (Kouhei Sutou)

- Description updated

#2 - 08/16/2012 09:08 PM - naruse (Yui NARUSE)

-  test-allがきちんと完走するならわたしは異論無いんですが、どうなんでしょう。
  それではtest-all用に別名を与えたほうがいいのかしらん。

#3 - 08/16/2012 10:42 PM - kou (Kouhei Sutou)

- あぁ、すみません、説明が足りませんでした。。。

09/05/2021 1/10
Rubyのテストにtest-unit gemを使いたいという話ではなく、Rubyをインストールしたユーザーがgem install test-unitとすればrequire "test/unit"でtest-unitが使えるようになってほしいということです。

#4 - 08/17/2012 10:23 AM - usa (Usaku NAKAMURA)

こんにちは、なかむら(う)です。

In message "[ruby-dev:46056] [ruby-trunk - Feature #6875] Make test/unit default gem" on Aug.16,2012 21:08:38, naruse@airemix.jp wrote:

test-all --disable-gems

---

U.Nakamura usa@garbagecollect.jp

#5 - 08/22/2012 09:55 AM - sorah (Sorah Fukumori)

test-allが通って、特にほかに問題がなければ入れちゃって良いと思います

#6 - 08/22/2012 11:52 AM - kou (Kouhei Sutou)

Running tests:

[ 9418/11373] TestRubyMode::TestIndent#test_singleton_class = 0.00 s

1) Skipped:

Finished tests in 413.081272s, 27.5321 tests/s, 5482.1125 assertions/s.

11373 tests, 2264558 assertions, 0 failures, 0 errors, 29 skips

ruby -v: ruby 2.0.0dev (2012-08-22 trunk 36709) [x86_64-linux]

#7 - 08/31/2012 09:50 PM - kou (Kouhei Sutou)

- Status changed from Open to Closed
- % Done changed from 0 to 100

This issue was solved with changeset r36872.
Kouhei, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- lib/test/unit/test-unit.gemspec: Make test/unit default gem. [Feature #6875] [ruby-dev:46051]

#8 - 10/18/2012 04:10 PM - vo.x (Vit Ondruch)

Hi guys,

Could you please enlighten what this issue solves? Adding .gemspec is not enough to create valid gem. The gem typically consist from following properties:

1) The /usr/bin stub allow to load specified version of gem, e.g. calling $ testrb 2.5.2 should call the latest version of test-unit but your stub does not do so
2) The gem usually contains code. It does not depend on some code lying somewhere in the StdLib
3) The gem could be usually found on rubygems.org. While there is test-unit available on rubygems.org, it is definitely different test-unit than the one shipped with Ruby.

Could you please stop confusing Ruby users and stop to make life harder to us, packagers? I really love to have rubygem-test-unit in Fedora which is well defined and I’d really love to be able to break StdLib into independent rubygems, which can be packaged separately and updated separately, but this is not the way. Could you please reconsider your steps? Thank you.

#9 - 10/20/2012 04:53 PM - kou (Kouhei Sutou)
https://bugs.ruby-lang.org/issues/6875

I’m the test-unit gem maintainer and working on default gem for RubyGems library.
I want to make Ruby users (especially test-unit gem users) life easier.

Could you please enlighten what this issue solves?

OK. I’ll describe it.

Here is a problem that I want to solve by this issue:

`test_XXX.rb`:

```ruby
require "test/unit"
class TestXXX < Test::Unit::TestCase
def test_XXX
  assert_equal(3, 1 + 2)
end
test_XXX_with_explicit_gem_test-unit.rb:
  gem "test-unit"
  require ".,/test_XXX"
```

Current:

- `% ruby test_XXX.rb` # -> uses test/unit in StdLib
- `% gem install test-unit`  
- `% ruby test_XXX.rb` # -> still uses test/unit in StdLib
- `% ruby test_XXX_with_explicit_gem_test-unit.rb` # -> uses test-unit gem

Expected:

- `% ruby test_XXX.rb` # -> uses test/unit in StdLib (same)
- `% gem install test-unit`  
- `% ruby test_XXX.rb` # -> still uses test/unit in StdLib (different)
- `% ruby test_XXX_with_explicit_gem_test-unit.rb` # -> uses test-unit gem (same)

Adding .gemspec is not enough to create valid gem.

It is right. There are still some tasks to implement default gem feature.

1) The /usr/bin stub allow to load specified version of gem, e.g. calling $ testrb 2.5.2 should call the latest version of test-unit but your stub does not do so

It is right. But it doesn’t cause the problem in testrb case. Because test-unit gem dropped testrb to avoid conflicting testrb that is installed by Ruby.

(It is not needed soon. I will implement the feature later.)

2) The gem usually contains code. It does not depend on some code lying somewhere in the StdLib

It is right. (.gemspec for test/unit StdLib doesn’t have any problems for it, doesn’t it?)

3) The gem could be usually found on rubygems.org. While there is test-unit available on rubygems.org, it is definitely different test-unit than the one shipped with Ruby.

It is right. (.gemspec for test/unit StdLib doesn’t have any problems for it, doesn’t it?)
Could you please stop confusing Ruby users and stop to make life harder to us, packagers?

I don't want to confusing Ruby users and make life harder to packages. If they have problems, I want to resolve them.

I'd really love to be able to break StdLib into independent rubygems

The "rubygems" means that gem packages on RubyGems.org not RubyGems library itself, doesn't it?
If it is true, the change doesn't depend on additional gem packages.

  can be packaged separately and updated separately, but this is not the way.

You can do both of them.
You just put test-unit.gemspec to Ruby's package. It doesn't add any additional dependencies.
RubyGems library does all other works. (It is work in progress. https://github.com/rubygems/rubygems/pull/377)

Again, I want to implement default gem feature without bothering packagers.
I don't know packagers' tasks well. Could you please tell me your problems? I'll resolve them.

Thanks.

#10 - 10/24/2012 08:22 PM - vo.x (Vit Ondruch)
kou (Kouhei Sutou) wrote:
https://bugs.ruby-lang.org/issues/6875

I'm the test-unit gem maintainer and working on default gem for RubyGems library.
I want to make Ruby users (especially test-unit gem users) life easier.

Thank you. I appreciate your effort.

  Could you please enlighten what this issue solves?

OK. I'll describe it.

Here is a problem that I want to solve by this issue:

  test_xxx.rb:

    require "test/unit"

    class TestXXX < Test::Unit::TestCase
      def test_xxx
        assert_equal(3, 1 + 2)
      end
    end

  test_xxx_with_explicit_gem_test-unit.rb:

    gem "test-unit"
    require ".test_xxx"

Current:

  % ruby test_xxx.rb          # -> uses test/unit in StdLib
  % gem install test-unit
  % ruby test_xxx.rb          # -> still uses test/unit in StdLib
  % ruby test_xxx_with_explicit_gem_test-unit.rb # -> uses test-unit gem

Expected:

  % ruby test_xxx.rb          # -> uses test/unit in StdLib (same)
  % gem install test-unit
  % ruby test_xxx.rb          # -> still uses test/unit in StdLib (different)
  % ruby test_xxx_with_explicit_gem_test-unit.rb # -> uses test-unit gem (same)
Sorry, I am not sure I understand this example. I expect that current is Ruby 1.9.3 and expected is Ruby 2.0, but what is the difference?

Adding .gemspec is not enough to create valid gem.

It is right. There are still some tasks to implement default gem feature.

1) The /usr/bin stub allow to load specified version of gem, e.g. calling $ testrb 2.5.2 should call the latest version of test-unit but your stub does not do so

It is right. But it doesn't cause the problem in testrb case. Because test-unit gem dropped testrb to avoid conflicting testrb that is installed by Ruby.

(It is not needed soon. I will implement the feature later.)

Good to know. Thank you.

2) The gem usually contains code. It does not depend on some code lying somewhere in the StdLib

It is right. (.gemspec for test/unit StdLib doesn't have any problems for it, doesn't it?)

StdLib doesn't, but I cannot package such gem, because it is not gem.

3) The gem could be usually found on rubygems.org. While there is test-unit available on rubygems.org, it is definitely different test-unit then the one shipped with Ruby.

It is right. (.gemspec for test/unit StdLib doesn't have any problems for it, doesn't it?)

It is not problem for StdLib, but how should I package something I don't have source for?

Could you please stop confusing Ruby users and stop to make life harder to us, packagers?

I don't want to confusing Ruby users and make life harder to packages. If they have problems, I want to resolve them.

Thank you. I appreciate your effort.

I'd really love to be able to break StdLib into independent rubygems

The "rubygems" means that gem packages on RubyGems.org not RubyGems library itself, doesn't it?
If it is true, the change doesn't depend on additional gem packages.

No I really mean that StdLib should consist from real gems. Since we need them for packaging and easy updates.

 can be packaged separately and updated separately, but this is not the way.

You can do both of them. You just put test-unit.gemspec to Ruby's package. It doesn't add any additional dependencies. RubyGems library does all other works. (It is work in progress. https://github.com/rubygems/rubygems/pull/377 )

Again, I want to implement default gem feature without bothering packagers.
I don't know packagers' tasks well. Could you please tell me your problems? I'll resolve them.

Thanks.

Thank you for your effort. I'll try to elaborate a bit.

In Fedora, we unbundled every gem from StdLib, i.e. the rake is not anymore part of StdLib. There is no reason to have it in StdLib, since we need only one version of Rake on system. That is the purpose of RPM and YUM, to manage your packages. It allows easy updates when new rake version is available as well as it prevents multiple copies around the system.

No for your 'gem', since there is no test-unit, just some test-unit shim, there is no way how to extract it from StdLib. We are already providing in Fedora full featured rubygem-test-unit, but how it is supposed the original part of StdLib?
For the rest, see my feedback in your pull request and thank you for pointing it out, since I am obviously highly interested in possible output.

**#11 - 10/26/2012 10:11 PM - kou (Kouhei Sutou)**

I comment on this issue.
I think that this issue is the right issue for talking about only test-unit.gemspec. If you want to talk about "default gem", [#5481](#5481) is the right issue. If the latter is true, please use [#5481](#5481).

vo.x (Vít Ondruch) wrote:

kou (Kouhei Sutou) wrote:

Sorry, I am not sure I understand this example. I expect that current is Ruby 1.9.3 and expected is Ruby 2.0, but what is the difference?

Sorry. Here is a correct example:

test_xxx.rb:

```ruby
require "test/unit"

class TestXXX < Test::Unit::TestCase
  def test_xxx
    assert_equal(3, 1 + 2)
  end
end
```

test_xxx_with_explicit_gem_test-unit.rb:

```ruby
gem "test-unit"
require "./test_xxx"
```

Current:

```ruby
% ruby test_xxx.rb          # -> uses test/unit in StdLib
% gem install test-unit
% ruby test_xxx.rb          # -> still uses test/unit in StdLib
% ruby test_xxx_with_explicit_gem_test-unit.rb # -> uses test-unit gem
```

Expected:

```ruby
% ruby test_xxx.rb          # -> uses test/unit in StdLib (same)
% gem install test-unit
% ruby test_xxx.rb          # -> uses test-unit gem (different)
% ruby test_xxx_with_explicit_gem_test-unit.rb # -> uses test-unit gem (same)
```

2) The gem usually contains code. It does not depend on some code lying somewhere in the StdLib

It is right. (.gemspec for test/unit StdLib doesn't have any problems for it, doesn't it?)

StdLib doesn't, but I cannot package such gem, because it is not gem.

You don't need to package it. It should be included in Ruby's package itself.

3) The gem could be usually found on rubygems.org. While there is test-unit available on rubygems.org, it is definitely different test-unit then the one shipped with Ruby.

It is right. (.gemspec for test/unit StdLib doesn't have any problems for it, doesn't it?)

It is not problem for StdLib, but how should I package something I don't have source for?

You don't need to package it. It should be included in Ruby's package itself because the source is included in Ruby.

I'd really love to be able to break StdLib into independent rubygems

The "rubygems" means that gem packages on RubyGems.org not RubyGems library itself, doesn't it?
If it is true, the change doesn't depend on additional gem packages.
No I really mean that StdLib should consist from real gems. Since we need them for packaging and easy updates.

I don't think that "packaging as RPM" equal to "easy updates". For example, users can update by "gem update". It is also easy update.

In Fedora, we unbundled every gem from StdLib, i.e. the rake is not anymore part of StdLib. There is no reason to have it in StdLib, since we need only one version of Rake on system.

There is a reason. We can use rake with "ruby -disable-gems" if rake is part of StdLib.
See also https://bugs.ruby-lang.org/projects/ruby/wiki/StdlibGem#ToDo. The page describes advantages and drawbacks.

Why "we need only one version of Rake on system"? Is it only for Rake or all libraries/softwares on Fedora?
Fedora has python and python3 packages. Is it not related about "we need only one version of Rake on system"?

It seems that some (or many) Rails users have one or more Rails versions on system. So I don't understand "we need only one version of Rake on system".

That is the purpose of RPM and YUM, to manage your packages. It allows easy updates when new rake version is available as well as it prevents multiple copies around the system.

Does "multiple copies" means "rake 0.9.2 and rake 0.9.2.2 are on the same system"?
I don't think that it is not "multiple copies"...

No for your 'gem', since there is no test-unit, just some test-unit shim, there is no way how to extract it from StdLib. We are already providing in Fedora full featured rubygem-test-unit, but how it is supposed the original part of StdLib?

I will not extract it from StdLib.

For the rest, see my feedback in your pull request and thank you for pointing it out, since I am obviously highly interested in possible output.

Thanks for your feedback.

#12 - 10/30/2012 11:26 PM - vo.x (Vit Ondruch)
kou (Kouhei Sutou) wrote:

I comment on this issue.
I think that this issue is the right issue for talking about only test-unit.gemspec. If you want to talk about "default gem", #5481 is the right issue. If the latter is true, please use #5481.

Well, yes, it would be probably better to discuss some points there :)

vo.x (Vit Ondruch) wrote:

kou (Kouhei Sutou) wrote:

Sorry, I am not sure I understand this example. I expect that current is Ruby 1.9.3 and expected is Ruby 2.0, but what is the difference?

Sorry. Here is a correct example:

test_xxx.rb:

```ruby
require "test/unit"

class TestXXX < Test::Unit::TestCase
  def test_xxx
    assert_equal(3, 1 + 2)
  end
end

test_xxx_with_explicit_gem_test-unit.rb:

  gem "test-unit"
  require ".:/test_xxx"

Current:

% ruby test_xxx.rb  # -> uses test/unit in StdLib
Yes, that make sense. Actually it is one issues I complained in #6124

2) The gem usually contains code. It does not depend on some code lying somewhere in the StdLib

It is right. (.gemspec for test/unit StdLib doesn't have any problems for it, doesn't it?)

StdString doesn't, but I cannot package such gem, because it is not gem.

You doesn't need to package it. It should be included in Ruby's package itself.

No, that is not true. Not on Fedora and similar guidelines apply to other distributions as well:

https://fedoraproject.org/wiki/Packaging:Guidelines#Bundling_of_multiple_projects
https://fedoraproject.org/wiki/Packaging:Treatment_Of_Bundled_Libraries
https://fedoraproject.org/wiki/Packaging:No_Bundled_Libraries

Actually, see also some comments from Lucas Nussbaum, who has the same interest as me, just for Debian.

3) The gem could be usually found on rubygems.org. While there is test-unit available on rubygems.org, it is definitely different test-unit then the one shipped with Ruby.

It is right. (.gemspec for test/unit StdLib doesn't have any problems for it, doesn't it?)

It is not problem for StdLib, but how should I package something I don't have source for?

You doesn't need to package it. It should be included in Ruby's package itself because the source is included in Ruby.

I'd really love to be able to break StdLib into independent rubygems

The "rubygems" means that gem packages on RubyGems.org not RubyGems library itself, doesn't it? If it is true, the change doesn't depend on additional gem packages.

No I really mean that StdLib should consist from real gems. Since we need them for packaging and easy updates.

I don't think that "packaging as RPM" equal to "easy updates". For example, users can update by "gem update". It is also easy update.

You have a view of Ruby developer. Ruby developer is used to use gem command. But I am defending interest of Fedora users. They don't really care about gems, they just want to have their application working and they want to use system tools to install the application. And in Fedora, that is RPM/YUM.

In Fedora, we unbundled every gem from StdLib, i.e. the rake is not anymore part of StdLib. There is no reason to have it in StdLib, since we need only one version of Rake on system.

There is a reason. We can use rake with "ruby -disable-gems" if rake is part of StdLib. See also https://bugs.ruby-lang.org/projects/ruby/wiki/StdLibGem#ToDo . The page describes advantages and drawbacks.

Yes, you chosen the option 2, while only the option 1 is the correct approach. What would be the purpose of --disable-gems flag, if you load gems? If you are using --disable-gems, it is probably for some reason. If you are using such flag, you should know what are you doing and since it is not that easy to find it in documentation, you have to be definitely more experienced Rubyst
Why "we need only one version of Rake on system"? Is it only for Rake or all libraries/softwares on Fedora?

Why would you need more versions of Rake? If it is ensured that all the applications are compatible with the Rake, why would I want to have more Rakes on my system? How many versions of Rake do you have on your system?

It applies of course to every library/application.

Fedora has python and python3 packages. Is it not related about "we need only one version of Rake on system"?

Yes, this is unfortunate exception and you would be able to find more. But anyway, the ultimate target is to have only one library/application.

It seems that some (or many) Rails users have one or more Rails versions on system. So I don't understand "we need only one version of Rake on system".

Here you are again speaking about Rails developers, not users. Users don't care about Rails, but about Redmine or Aeolus, which are build using Rails.

That is the purpose of RPM and YUM, to manage your packages. It allows easy updates when new rake version is available as well as it prevents multiple copies around the system.

Does "multiple copies" means "rake 0.9.2 and rake 0.9.2.2 are on the same system"?
I don't think that it is not "multiple copies"...

Yes, this is what I mean by multiple copies.

No for your 'gem', since there is no test-unit, just some test-unit shim, there is no way how to extract it from StdLib. We are already providing in Fedora full featured rubygem-test-unit, but how it is supposed the original part of StdLib?

I will not extract it from StdLib.

Yes, because there is nothing to extract. That is the point. In Ruby, there is currently Minitest and test-unit, which is actually not test unit. If there would be something to extract into gem, I would already did it for Fedora.

#13 - 11/20/2012 11:00 PM - kou (Kouhei Sutou)
Sorry for my late response...

vo.x (Vit Ondruch) wrote:

kou (Kouhei Sutou) wrote:

I comment on this issue.
I think that this issue is the right issue for talking about only test-unit.gemspec. If you want to talk about "default gem", #5481 is the right issue. If the latter is true, please use #5481.

Well, yes, it would be probably better to discuss some points there :)

Thanks for your agreement.
I'll comment on only test-unit.gemspec things.

You have a view of Ruby developer. Ruby developer is used to use gem command. But I am defending interest of Fedora users. They don't really care about gems, they just want to have their application working and they want to use system tools to install the application. And in Fedora, that is RPM/YUM.

I understand. You're right. I have a view of Ruby developer.
(I understand that "Ruby developer" means developers who use Ruby for programming. It doesn't mean that developers who develope Ruby itself.)

I misunderstood that Fedora users are both Ruby developers and users that use applications that written by Ruby.

No for your 'gem', since there is no test-unit, just some test-unit shim, there is no way how to extract it from StdLib. We are already providing in Fedora full featured rubygem-test-unit, but how it is supposed the original part of StdLib?

I will not extract it from StdLib.
Yes, because there is nothing to extract. That is the point. In Ruby, there is currently Minitest and test-unit, which is actually not test unit. If there would be something to extract into gem, I would already did it for Fedora.

OK. I understand that test/unit in the Ruby package needs to do nothing.

I hope that NaHi comments on #5481...

It seems that he thought about it in RubyConf. He tweets summary on twitter:

https://twitter.com/nahi/status/264807829527986176

@drbrain imports RubyGems 2.0 w/ @ktou's fakegem fix (default gem) into CRuby 2.0.0. And we help @hone02 for bundler compat. /cc evanphx (Evan Phoenix)

https://twitter.com/nahi/status/264808442626179072

nahi (Hiroshi Nakamura) And real fix (install stdlib as a real gem) would come after 2.0. drbrain (Eric Hodel) @ktou @hone02 Thanks for discussing! /cc evanphx (Evan Phoenix)

I hope that he describes about details of it on #5481...

Files

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>test-unit-default-gem.diff</td>
<td>699 Bytes</td>
<td>08/15/2012</td>
<td>kou (Kouhei Sutou)</td>
</tr>
</tbody>
</table>