Ruby master - Feature #12306

Implement String #blank? #present? and improve #strip and family to handle unicode

04/20/2016 11:33 PM - sam.saffron (Sam Saffron)

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

Description

Time and again there have been rejected feature requests to Ruby core to implement blank and present protocols across all objects as ActiveSupport does. I am fine with this call and think it is fair.

However, for the narrow case of String having #blank? and #present? makes sense.

- Provides a natural extension over #strip, #lstrip and #rstrip. (" ".strip.length == 0) == " ".blank?
- Plays nicely with ActiveSupport, providing an efficient implementation in Ruby core: see: https://github.com/SamSaffron/fast_blank, implementing blank efficiently requires a c extension.

However, if this work is to be done, #strip and should probably start dealing with unicode blanks, eg:

```
irb(main):008:0> [0x3000].pack("U")
=> "　"
irb(main):009:0> [0x3000].pack("U").strip.length
=> 1
```

So there are 2 questions / feature requests here

1. Can we add blank? and present? to String?
2. Can we amend strip and family to account for unicode per: https://github.com/SamSaffron/fast_blank/blob/master/ext/fast_blank/fast_blank.c#L43-L74

Related issues:

Related to Ruby master - Feature #8110: Regex methods not changing global var...	**Closed**
Related to Ruby master - Feature #12403: Optimise Regexp#match?	**Open**
Is duplicate of Ruby master - Feature #8206: Should Ruby core implement Strin...	**Open**

History

#1 - 04/21/2016 12:20 AM - shyouhei (Shyouhei Urabe)

What about non-Unicode strings?

#2 - 04/21/2016 12:52 AM - matz (Yukihiro Matsumoto)

I agree with making strip etc. encoding aware, as we did for upcase etc., although priority is low.

I don't see the benefit of blank? and present?, so it has little chance to have them in standard Ruby.
But if String#blank? checks if the string only contains spaces, it means something, but I still don't see real-world use-case.

Matz.

#3 - 04/21/2016 02:12 AM - nobu (Nobuyoshi Nakada)

I think that String#blank? equals to /\P{space}/ !~ self.
Is it slow, especially than strip?

#4 - 04/21/2016 02:29 AM - sam.saffron (Sam Saffron)

Nobuyoshi Nakada wrote:

> I think that String#blank? equals to /\P{space}/ !~ self.
> Is it slow, especially than strip?

strip would force allocation of a duplicate string, so is a bit of a non starter.

Yes, in Rails apps this blank very quickly becomes a bottleneck see: http://tmm1.net/ruby21-profiling/ for a real world example. I have seen Rails applications spend 2-5% of the time in Active Support String#blank? it is used super aggressively due to the "style" of programming used.

Recently Richard Schneems has been micro optimizing blank? due to this discovery see: https://github.com/rails/rails/pull/24658

The regex approach will never come close to a native implementation see bench here: https://github.com/SamSaffron/fast_blank

matz (Yukihiro Matsumoto) as to why this tends to happen so much in Rails world, this is a super common scenario

- Allow a textinput with User name
- Then from controller run

```
# one letter name is fine, especially for Chinese names, but blank is not
raise YouGotToAddAName if params[:name].blank?
```

In Rails programming it turns out that people are checking for "blankness" of strings in tons and tons of very very common scenario. The speed boost of having a fast native implementation would be very welcome.

I am fine with not adding present? (unless s.blank?) fills the need just fine

Regarding timing, would a PR adding blank? and unicode support to strip be welcome for 2.4 timeframe?

Regarding non-unicode strings I think strip should just keep working the way it does now.

#5 - 04/21/2016 02:50 AM - rafaelfranca (Rafael França)

Other real wold examples:

You want to validate if a username is provided in the form and your user submitted three whitespaces ("   "). This is obvious not a valid username so your code checks for it.

In fact every single Rails application that uses:

```
validates_presense_of :my_attribute
```

Uses String#blank?.

Other real example is the mail gem. It also implements blank?. It is used, for example, to check if the from or the to address is filled. An string with only whitespaces is not a valid email address.

This example shows that this is not specific for Rails applications. Any ruby script that will accept user supplied parameters and can't accept strings with only whitespaces will need to have an implementation of blank?.

In my opinion, a shared implementation of blank? in the Ruby standard library would be beneficial to the Ruby community. Right now as it stands, all performance improvements made in blank? for Rails will not be shared by the mail gem or other Ruby programs.

#6 - 04/21/2016 03:52 AM - nobu (Nobuyoshi Nakada)

Rafael França wrote:

> You want to validate if a username is provided in the form and your user submitted three whitespaces ("   "). This is obvious not a valid username so your code checks for it.

Is "' || 1=1" a valid username?

> Other real example is the mail gem. It also implements blank?. It is used, for example, to check if the from or the to address is filled. An string with only whitespaces is not a valid email address.

Validating mail address is a hard job.
"@@@" is not only whitespaces but is it a valid email address?

Your examples don't seem reasonable.

#7 - 04/21/2016 04:07 AM - sam.saffron (Sam Saffron)

Nobu,

There is a subtle change validation wise with user names:

"Please enter a username, username must not be blank"

vs

"Please enter a valid username, username must not contain the letter E"

Checking for blank is a distinct class of errors (user forgot or entered spaces in one of the fields)

#8 - 04/21/2016 04:09 AM - rafaelfranca (Rafael França)

> Your examples don't seem reasonable.

Both are real examples, being used in a lot of applications for years, so they are reasonable.

> Validating mail address is a hard job. "@@@" is not only whitespaces but is it a valid email address?

Indeed, but I was not entering in the merit of email address validation. The validation is a simple check to raise argument error if the email address (or any other field not only email address) is blank. Perhaps I could link to the places were the method is used instead of were it is defined so you can see that it is reasonable:

https://github.com/mikel/mail/blob/df48a05a7fb5a4271e6df12da7afb26a53494a18/lib/mail/check_delivery_params.rb#L5
https://github.com/mikel/mail/blob/df48a05a7fb5a4271e6df12da7afb26a53494a18/lib/mail/fields/content_disposition_field.rb#L41
https://github.com/mikel/mail/blob/df48a05a7fb5a4271e6df12da7afb26a53494a18/lib/mail/fields/mime_version_field.rb#L24

> Is "' || 1=1" a valid username?

No, it is not, but username was only one example. That validation (that is real and is being used in a lot of application) can be used to any attribute, from usernames to comments body. In fact, this same application that we are using to track Ruby's issues is using String#blank?:

https://github.com/redmine/redmine/search?utf8=%E2%9C%93&q=validates_presence_of

#9 - 04/21/2016 04:10 AM - sam.saffron (Sam Saffron)

Also as a point of reference, have a look at all the calls to String #blank? made from Redmine, the very software powering the bug tracker (present boils down to a call on blank most times)

https://github.com/redmine/redmine/search?utf8=%E2%9C%93&q=present

https://github.com/redmine/redmine/search?utf8=%E2%9C%93&q=blank&type=Code

#10 - 04/21/2016 04:17 AM - rafaelfranca (Rafael França)

Seems that even Ruby itself would benefits from String#blank?

https://github.com/ruby/ruby/search?utf8=%E2%9C%93&q=%22strip.empty%3F%22&type=Code
https://github.com/ruby/ruby/search?utf8=%E2%9C%93&q=%22strip.length%22&type=Code

#11 - 04/21/2016 07:16 AM - shyouhei (Shyouhei Urabe)

Rafael's comment about non-Rails use case of .strip.empty? is very interesting. This shows that adding blank method can benefit wild situations where programmers have no other choice than .strip.empty? which doesn't fluently describe what is wanted.

I know there is blank method in Rails and everybody uses that. However that specific use-case is already treated by ActiveSupport. I see no need to add this in core to save your Rails application, because that solves no problem (apart from being slow). Non-Rails use case seems much more important for people who want this method, especially when they persuade Matz.

#12 - 04/21/2016 03:29 PM - schneems (Richard Schneeman)

I think this is useful outside of Rails. The Active Support module has 87 million downloads on rubygems.org while Railties has only 53 million downloads. So 34 million times people have needed to use Active Support without Railties, this is a huge number. Granted not all of them will only be for present? but some of them will be. In many cases people will want this logic and maybe they cannot or do not wish to pull in Active Support. I find it quite common to either check for presence of a variable and then to see if it is empty? or to strip.empty?.

Here is a quick grep of the examples on my computer (it is by no means comprehensive).

Nil and empty? checks together

```
./bundler/lib/bundler/cli/lock.rb:      if gems && !gems.empty?
./bundler/lib/bundler/cli/open.rb:      editor = [ENV["BUNDLER_EDITOR"], ENV["VISUAL"], ENV["EDITOR"]].find {|
e| !e.nil? && !e.empty? }
./bundler/lib/bundler/cli.rb:      if ENV["RUBYGEMS_GEMDEPS"] && !ENV["RUBYGEMS_GEMDEPS"].empty?
./carrierwave/lib/carrierwave/uploader/download.rb:            return match[1] unless match.nil? || match[1].e
```

```
mpty?
./fog/lib/fog/clodo/core.rb:        :path     =>  (uri.path and not uri.path.empty?) ? uri.path : 'v1.0'./fog/
lib/fog/clodo/models/compute/server.rb:          pubaddrs && !pubaddrs.empty? ? pubaddrs.first['ip'] : nil]
./fog/lib/fog/cloudstack/models/compute/servers.rb:          if servers.nil? || servers.empty?
./fog/lib/fog/cloudstack/models/compute/servers.rb:          unless servers.nil? || servers.empty?
./git_hub_bub/lib/git_hub_bub/request.rb:      self.options[:query]   = query if query && !query.empty?
./puma/lib/puma/rack/urlmap.rb:        next unless !rest || rest.empty? || rest[0] == ?/
./rack/lib/rack/mock.rb:      env[PATH_INFO]       = (!uri.path || uri.path.empty?) ? "/" : uri.path
./rack/lib/rack/multipart/parser.rb:            elsif !filename && data.empty?
./rack/lib/rack/multipart/parser.rb:          if name.nil? || name.empty?
./rack/lib/rack/multipart/parser.rb:        if content.nil? || content.empty?
./rack/lib/rack/server.rb:        options[:config] = args.last if args.last && !args.last.empty?
./rack/lib/rack/session/abstract/id.rb:          value && !value.empty?
./rake/lib/rake/task.rb:      add_comment(comment) if comment && ! comment.empty?
./rdoc/lib/rdoc/code_object.rb:                if comment and not comment.empty? then
./rdoc/lib/rdoc/parser/c.rb:      if no_match and no_match.empty? then
./rest-client/lib/restclient/request.rb:      if (!body) || body.empty?
./rpm/lib/new_relic/agent/agent.rb:          if data && !data.empty?
./rpm/lib/new_relic/agent/javascript_instrumentor.rb:        value.nil? || value.empty?
./rpm/lib/new_relic/agent/obfuscator.rb:        if key.nil? || key.empty?
./rubinius/library/rubygems/command.rb:    if args.nil? or args.empty? then
./ruby/.ext/common/psych/class_loader.rb:      return nil if !klassname || klassname.empty?
./ruby/.ext/common/tk/menu.rb:    configure(keys) if keys && !keys.empty?
./ruby/.ext/common/tkextlib/tktable/tktable.rb:    obj.configure(keys) if keys && ! keys.empty?
./ruby/.ext/common/tkextlib/tktable/tktable.rb:    configure(keys) if keys && ! keys.empty?
./ruby/lib/optparse.rb:        elsif !opt or opt.empty?
./ruby/lib/rubygems/command.rb:    if args.nil? or args.empty? then
./ruby/lib/rubygems/command.rb:    return if option_list.nil? or option_list.empty?
./sass/lib/sass/engine.rb:      if content.first && content.first.strip.empty?
./sprockets/lib/sprockets/loader.rb:          if cached_asset[:metadata][:included] && !cached_asset[:metadata
][:included].empty?
./sprockets/lib/sprockets/loader.rb:          if cached_asset[:metadata][:links] && !cached_asset[:metadata][:
links].empty?
./sprockets/lib/sprockets/loader.rb:          if cached_asset[:metadata][:stubbed] && !cached_asset[:metadata]
[:stubbed].empty?
./sprockets/lib/sprockets/loader.rb:          if cached_asset[:metadata][:required] && !cached_asset[:metadata
][:required].empty?
./sprockets/lib/sprockets/loader.rb:          if cached_asset[:metadata][:dependencies] && !cached_asset[:meta
data][:dependencies].empty?
./www.ruby-lang.org/_plugins/posted_by.rb:      if author.nil? || author.empty? || author == 'Unknown Author'
./yard/lib/yard/cli/yri.rb:        if @name.nil? || @name.strip.empty?
./yard/lib/yard/parser/ruby/ruby_parser.rb:              if comment && !comment.empty?
```

strip.empty? checks

```
./concurrent-ruby/doc/actor/format.rb:        unless chunk.strip.empty? || chunk =~ /\A *#/
./concurrent-ruby/examples/format.rb:        unless chunk.strip.empty? || chunk =~ /\A *#/
./fog/lib/fog/rackspace/service.rb:          !response.body.strip.empty? &&
./passenger/lib/phusion_passenger/platform_info/compiler.rb:        if source.strip.empty?
./passenger/lib/phusion_passenger/platform_info.rb:      indent = str.split("\n").select{ |line| !line.strip.e
mpty? }.map{ |line| line.index(/[^\s]/) }.compact.min || 0
./rdoc/lib/rdoc/context.rb:        known.value.nil? or known.value.strip.empty?
./rpm/lib/new_relic/cli/commands/deployments.rb:      @description = nil if @description && @description.strip
.empty?
./rubinius/rakelib/instruction_parser.rb:        elsif line.strip.empty?
./rubinius/tools/rubuildius/bin/pastie.rb:    return if body.strip.empty?
./rubinius/vm/codegen/field_extract.rb:    return '' if out.strip.empty?
./ruby/ext/ripper/tools/strip.rb:  if line.strip.empty?
./ruby/ext/tk/extconf.rb:  defs.map{|ary| s = ary.join(''); (s.strip.empty?)? "": "-D" << s}
./ruby/ext/tk/extconf.rb:        !TkConfig_Info['TK_XINCLUDES'].strip.empty?) ||
./ruby/ext/tk/extconf.rb:        (TkConfig_Info['TK_XLIBSW'] && !TkConfig_Info['TK_XLIBSW'].strip.empty?)
./ruby/ext/tk/extconf.rb:      !TkConfig_Info['TK_XINCLUDES'].strip.empty?
./ruby/ext/tk/extconf.rb:          TkConfig_Info['TK_XLIBSW'] && !TkConfig_Info['TK_XLIBSW'].strip.empty?
./ruby/ext/tk/extconf.rb:    !TclConfig_Info['TCL_STUB_LIB_SPEC'].strip.empty? &&
./ruby/ext/tk/extconf.rb:    !TkConfig_Info['TK_STUB_LIB_SPEC'].strip.empty?
./ruby/ext/tk/extconf.rb:          !TclConfig_Info['TCL_BUILD_STUB_LIB_SPEC'].strip.empty?
./ruby/ext/tk/extconf.rb:          !TclConfig_Info['TCL_BUILD_LIB_SPEC'].strip.empty?
./ruby/ext/tk/extconf.rb:          !TclConfig_Info['TK_BUILD_STUB_LIB_SPEC'].strip.empty?
./ruby/ext/tk/extconf.rb:          !TclConfig_Info['TK_BUILD_LIB_SPEC'].strip.empty?
./ruby/lib/net/http/header.rb:        .reject {|str| str.strip.empty? }\
./ruby/lib/rdoc/context.rb:        known.value.nil? or known.value.strip.empty?
./sass/lib/sass/engine.rb:        if line.strip.empty?
./sass/lib/sass/engine.rb:      if value.strip.empty?
./sass/lib/sass/engine.rb:      if value.strip.empty? && line.children.empty?
./sass/lib/sass/engine.rb:      if content.first && content.first.strip.empty?
```

```
./yard/lib/yard/cli/yri.rb:        if @name.nil? || @name.strip.empty?
./yard/lib/yard/i18n/text.rb:              if line.strip.empty?
./yard/lib/yard/tags/directives.rb:          (tag.text && !tag.text.strip.empty?)
./yard/templates/default/docstring/setup.rb:  if text.strip.empty? && object.tags(:return).size == 1 && object
.tag(:return).text
```

In many of these examples the desired result is not clear at a glance. It would simplify code and comprehension to have a common short hand that can be used when it is not appropriate to pull in Active Support.

#13 - 04/21/2016 04:37 PM - PSchambacher (Pierre Schambacher)

Just my 2 cents here but any time I've been writing a pure ruby application, I ended up including active support or copy-pasting the blank? method.

There's a lot of applications of pure ruby code to know if the string that we have is just blank space: parsing a CSV, reading the response from a web request, receive something from a socket, reading a file, ... empty? is not always enough since there's scenarii where having a couple blank spaces make no more sense than nothing at all.

I also see the point from Nobu about using regular expressions to validate the input but that's a sword with 2 edges. In theory it's great but in practice having the perfect regular expression is pretty difficult (see http://www.kalzumeus.com/2010/06/17/falsehoods-programmers-believe-about-names/ for instance). In lots of cases, having "something that's not blank" is just the simplest and surest thing to do.

#14 - 04/24/2016 03:14 PM - naruse (Yui NARUSE)

Richard Schneeman wrote:

> I think this is useful outside of Rails. The Active Support module has 87 million downloads on rubygems.org while Railties has only 53 million downloads. So 34 million times people have needed to use Active Support without Railties, this is a huge number. Granted not all of them will only be for present? but some of them will be. In many cases people will want this logic and maybe they cannot or do not wish to pull in Active Support. I find it quite common to either check for presence of a variable and then to see if it is empty? or to strip.empty?.
>
> Here is a quick grep of the examples on my computer (it is by no means comprehensive).

Nil and empty? checks together

you can already write str&.empty?.

strip.empty? checks

```
./concurrent-ruby/doc/actor/format.rb:        unless chunk.strip.empty? || chunk =~ /\A *#/
./concurrent-ruby/examples/format.rb:        unless chunk.strip.empty? || chunk =~ /\A *#/
./fog/lib/fog/rackspace/service.rb:          !response.body.strip.empty? &&
./passenger/lib/phusion_passenger/platform_info/compiler.rb:        if source.strip.empty?
./passenger/lib/phusion_passenger/platform_info.rb:      indent = str.split("\n").select{ |line| !line.str
ip.empty? }.map{ |line| line.index(/[^\s]/) }.compact.min || 0
./rdoc/lib/rdoc/context.rb:        known.value.nil? or known.value.strip.empty?
./rpm/lib/new_relic/cli/commands/deployments.rb:      @description = nil if @description && @description.s
trip.empty?
./rubinius/rakelib/instruction_parser.rb:        elsif line.strip.empty?
./rubinius/tools/rubuildius/bin/pastie.rb:    return if body.strip.empty?
./rubinius/vm/codegen/field_extract.rb:    return '' if out.strip.empty?
./ruby/ext/ripper/tools/strip.rb:  if line.strip.empty?
./ruby/ext/tk/extconf.rb:  defs.map{|ary| s = ary.join(''); (s.strip.empty?)? "": "-D" << s}
./ruby/ext/tk/extconf.rb:        !TkConfig_Info['TK_XINCLUDES'].strip.empty?) ||
./ruby/ext/tk/extconf.rb:        (TkConfig_Info['TK_XLIBSW'] && !TkConfig_Info['TK_XLIBSW'].strip.empty?)
./ruby/ext/tk/extconf.rb:      !TkConfig_Info['TK_XINCLUDES'].strip.empty?
./ruby/ext/tk/extconf.rb:          TkConfig_Info['TK_XLIBSW'] && !TkConfig_Info['TK_XLIBSW'].strip.empty?
./ruby/ext/tk/extconf.rb:    !TclConfig_Info['TCL_STUB_LIB_SPEC'].strip.empty? &&
./ruby/ext/tk/extconf.rb:    !TkConfig_Info['TK_STUB_LIB_SPEC'].strip.empty?
./ruby/ext/tk/extconf.rb:          !TclConfig_Info['TCL_BUILD_STUB_LIB_SPEC'].strip.empty?
./ruby/ext/tk/extconf.rb:          !TclConfig_Info['TCL_BUILD_LIB_SPEC'].strip.empty?
./ruby/ext/tk/extconf.rb:          !TclConfig_Info['TK_BUILD_STUB_LIB_SPEC'].strip.empty?
./ruby/ext/tk/extconf.rb:          !TclConfig_Info['TK_BUILD_LIB_SPEC'].strip.empty?
./ruby/lib/net/http/header.rb:         .reject {|str| str.strip.empty? }\
./ruby/lib/rdoc/context.rb:        known.value.nil? or known.value.strip.empty?
./sass/lib/sass/engine.rb:        if line.strip.empty?
./sass/lib/sass/engine.rb:      if value.strip.empty?
./sass/lib/sass/engine.rb:      if value.strip.empty? && line.children.empty?
./sass/lib/sass/engine.rb:      if content.first && content.first.strip.empty?
./yard/lib/yard/cli/yri.rb:        if @name.nil? || @name.strip.empty?
./yard/lib/yard/i18n/text.rb:              if line.strip.empty?
./yard/lib/yard/tags/directives.rb:          (tag.text && !tag.text.strip.empty?)
./yard/templates/default/docstring/setup.rb:  if text.strip.empty? && object.tags(:return).size == 1 && ob
ject.tag(:return).text
```

Most of them looks different from ActiveSupport's String#blank? which is Unicode aware.
If you replace them with String#blank, it will break those code.
It is what we wonder about.

#15 - 04/24/2016 07:01 PM - shevegen (Robert A. Heiler)

Just one comment, not related to the suggestion itself but to one
other comment made above.

Richard Schneeman wrote:

> I think this is useful outside of Rails. The Active Support module has
> 87 million downloads on rubygems.org while Railties has only 53 million
> downloads. So 34 million times people have needed to use Active Support
> without Railties, this is a huge number.

This is not a good comparison to show the "most downloads" statistics alone.

Many downloads are bundled together with prior dependencies, e. g. rails.
All the Active* gems are, in most cases, tied to the usage of rails. Most
people who will do a "gem install rails" won't know what the individual
active* gems do or how they work.

Not everyone who uses ruby, also uses rails. And idioms that are available
to ruby itself, should primarily make and see fit within ruby itself,
not to external gems/projects.

A lot of rails has also influenced ruby itself too, but rails is not ruby.

String#blank? may fit well into the active* world but for a regular
string object, it is not entirely clear what it should mean. Should
it mean whether the String contains one or more "blanks"? In this case,
it would have to check for at least one ' ' in that string object.

But this is not what the method does. The documentation is:

"A string is blank if it's empty or contains whitespaces only"

So the name of the method is actually wrong in my opinion, since it also
checks whether the string is empty. Is an empty string a blank string
or does it include a "blank" string or character? In an empty string,
there surely is no space character.

It sure enough does not include any ' ' so why would it be considered
blank?

And for string objects such as:

"This cat is a happy cat.".blank?

This would return false in the above definition, but it sure enough
does include the ' ' character.

I assume the method .blank? here more does a .strip.empty? check
combined. Calling this operation ".blank?" still does not seem to
be right IMO.

An advantage, and a huge point in favour of ninja patching (also called
monkey patching), you model ruby within the particular domain thinking.
(And with refinements, perhaps one day we have means to make changes to
core classes of ruby, limited to only that particular domain. Like, to
retain the "default" behaviour of ruby core/stdlib classes and refer to
these, but anyway, this is for another discussion.)

#16 - 04/24/2016 11:03 PM - sam.saffron (Sam Saffron)

Nobu,

Regarding:

> Most of them looks different from ActiveSupport's String#blank? which is Unicode aware.

I think this change (if it happens) must come with unicode aware strip/rstrip/lstrip (and underlying is_space) for UTF-8 encoded strings. It is clear that there is a desire to make string operations encoding aware so this would have to be lockstep. All of the strip.length==0 in core can safely be changed to a unicode aware blank? function with zero regression.

#17 - 04/25/2016 02:37 AM - shyouhei (Shyouhei Urabe)

Sam Saffron wrote:

> All of the strip.length==0 in core can safely be changed to a unicode aware blank? function with zero regression.

Can you elaborate with this? For instance this example

```
./ruby/ext/tk/extconf.rb: !TkConfig_Info['TK_XINCLUDES'].strip.empty?
```

is clear that the XINCLUDES indicates this string is passed to a shell script. The call of strip here surely intends stripping shell's definition of whitespaces, not the Unicode one. Do you think Unicode-aware strip cannot introduce regressions for this kind of situations?

#18 - 04/25/2016 10:05 AM - duerst (Martin Dürst)

- Assignee set to matz (Yukihiro Matsumoto)

Several comments, all in one post:

1) .blank? definitely cannot check for all problems in an input field, but it seems to be used very often because it very easily catches a frequent user mistake. More precise checking is more difficult, more application-dependent, and will catch less mistakes. So I'm not surprised that may programmers use it, even if it's just only a first step.

2) If we want to make Ruby 3 times faster by Ruby 3.0 (see Matz's keynote at last year's Ruby Kaigi), then ignoring a request to speed up functionality that may take up to 2% to 5% of time in the most widely used application of Ruby seems to be rather counter-purpose.

3) Regarding Unicode awareness, there are many ways to extend the definition of white space. See e.g. https://discourse.wicg.io/t/whitespace-is-hard-and-buggy-can-we-normalize-it/1436. The experience of Rails may be very valuable, but we'll have to look at it in detail.

4) Regarding backwards compatibility, for upcase/downcase/..., Matz has said that he's willing to make it backwards-incompatible in edge cases (if you have non-ASCII data, but *really* only want ASCII to change case). That might apply here, too. But we have to discuss it.

#19 - 04/25/2016 11:32 AM - duerst (Martin Dürst)

Hello Sam,

Just in private for the moment. I'm trying to find the definition of
blank? in the rails source code. I expected it somewhere around
here:https://github.com/rails/rails/tree/master/activesupport/lib/active_support/core_ext/string
but I didn't find anything.

Regards, Martin.

#20 - 04/25/2016 11:38 AM - jbilbo (Jonathan Hernandez)

https://github.com/rails/rails/blob/master/activesupport/lib/active_support/core_ext/object/blank.rb

Martin Dürst wrote:

> Hello Sam,
>
> Just in private for the moment. I'm trying to find the definition of
> blank? in the rails source code. I expected it somewhere around
> here:https://github.com/rails/rails/tree/master/activesupport/lib/active_support/core_ext/string
> but I didn't find anything.
>
> Regards, Martin.

#21 - 04/25/2016 02:12 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

Yui NARUSE wrote:

> Richard Schneeman wrote:
>
> ## Nil and empty? checks together

> you can already write str&.empty?.

It's not useful. The logic we are looking for is basically !(nil? || empty?). str&.empty? would return nil for nil and true for '' (empty).

If Ruby provided "non_empty?" we could use it like str&.non_empty? and it would make sense.

#22 - 04/25/2016 06:57 PM - naruse (Yui NARUSE)

Rodrigo Rosenfeld Rosas wrote:

> Yui NARUSE wrote:
>
> > Richard Schneeman wrote:
> >
> > ## Nil and empty? checks together
> >
> > you can already write str&.empty?.
>
> It's not useful. The logic we are looking for is basically !(nil? || empty?). str&.empty? would return nil for nil and true for '' (empty).
>
> If Ruby provided "non_empty?" we could use it like str&.non_empty? and it would make sense.

Ah, yes true.
I just remind old my proposal https://bugs.ruby-lang.org/issues/12075

#23 - 04/26/2016 10:12 AM - sam.saffron (Sam Saffron)

Shyouhei Urabe wrote:

> is clear that the XINCLUDES indicates this string is passed to a shell script. The call of strip here surely intends stripping shell's definition of whitespaces, not the Unicode one. Do you think Unicode-aware strip cannot introduce regressions for this kind of situations?

In this particular case I fail to see how this can cause any problem.

```
sam@ubuntu ruby % ruby -e 'p ARGV[0]' -test
"-test-"
```

Most shells these days allow you to pass in UTF-8 strings, but saying that there is one real case in the universe where someone really wanted to include the library -hacky– as opposed to the lib called hacky when building tk is not true. blank? and utf8 aware strip would work fine here and not break a single real example in this usage.

That said, internal uses of strip while dealing with shell params passed to ARGV may need some care, but requiring people start quoting leading and trailing unicode spaces is not really a major breaking change imo.

#24 - 04/27/2016 01:40 AM - shyouhei (Shyouhei Urabe)

Sam Saffron wrote:

> In this particular case I fail to see how this can cause any problem.

You failed to see the problem because you could not imagine a file path containing U+3000. That is not very rare in cultures with ideographics. No, I'm not against categorizing such path being insane. But they ARE there.

> Most shells these days allow you to pass in UTF-8 strings, but saying that there is one real case in the universe where someone really wanted to include the library -hacky– as opposed to the lib called hacky when building tk is not true. blank? and utf8 aware strip would work fine here and not break a single real example in this usage.

U+3000 inside of a path should not be stripped. That should break existing working codes.

> That said, internal uses of strip while dealing with shell params passed to ARGV may need some care, but requiring people start quoting leading and trailing unicode spaces is not really a major breaking change imo.

So you admit this change do emit regressions, not zero. You just say this is minor.

#25 - 04/27/2016 06:51 AM - naruse (Yui NARUSE)

I just come back tmm1's bloc: http://tmm1.net/ruby21-profiling/
You guys are talking on this.

But I'm wondering whether it is still true after https://github.com/rails/rails/pull/24658 is merged.

#26 - 04/27/2016 06:57 AM - sam.saffron (Sam Saffron)

Shyouhei Urabe wrote:

> You failed to see the problem because you could not imagine a file path containing U+3000. That is not very rare in cultures with ideographics. No, I'm not against categorizing such path being insane. But they ARE there.

I can see that ... but I can not see someone trying to build tk and expecting that including –would work.

Nobu,

Yes the micro optimisations in blank only yield a 10-30% improvement see the benches in https://github.com/SamSaffron/fast_blank native is_space loop is often 10x faster. feel free to rerun the benches against latest code.

#27 - 04/27/2016 07:59 AM - sam.saffron (Sam Saffron)

note this method exists in .NET String

https://msdn.microsoft.com/en-us/library/system.string.isnullorwhitespace(v=vs.110).aspx

and in Python

http://www.tutorialspoint.com/python/string_isspace.htm

#28 - 04/28/2016 12:08 AM - sam.saffron (Sam Saffron)

Just to expand on how hard this is to get right without the framework providing it

See:

https://gist.github.com/SamSaffron/d1a9cc8e141e7415e06306369fdedfe5

/[[:^space:]]/ === str can cause significantly more data to allocate including invisible MatchData and Strings vs

/\A[[:space:]]*\z/ === str

(depending on the string being tested)

There is no way to invoke the regex engine without it magically setting a pile of globals, making it very inefficient to do lots of things with regex. 3 years ago when I brought this up, Nobu suggested allowing String#include? to accept a regex and set no globals, that may be a way to get a bunch of perf out of the regex engine. Or simply stop with all the globals in Ruby 3 and have specific methods for getting match data always used. I don't know.

My point is, doing something even trivial here (test if string is blank) is practically impossible to do fast in Ruby today.

#29 - 04/28/2016 07:42 AM - nobu (Nobuyoshi Nakada)

Sam Saffron wrote:

> There is no way to invoke the regex engine without it magically setting a pile of globals, making it very inefficient to do lots of things with regex. 3 years ago when I brought this up, Nobu suggested allowing String#include? to accept a regex and set no globals, that may be a way to get a bunch of perf out of the regex engine. Or simply stop with all the globals in Ruby 3 and have specific methods for getting match data always used. I don't know.

I'm sorry that I don't remember it.
Can't you show the pointer?

#30 - 04/28/2016 08:06 AM - sam.saffron (Sam Saffron)

Sure Nobu,

https://bugs.ruby-lang.org/issues/8206

very close to this feature request that I created years ago.

#31 - 04/28/2016 02:07 PM - nobu (Nobuyoshi Nakada)

- Is duplicate of Feature #8206: Should Ruby core implement String#blank? added

#32 - 05/06/2016 05:58 AM - shyouhei (Shyouhei Urabe)

Anyways, given the real-world use-case, I now think it's a good idea to have something that does strip.empty? -equivalent method for String. Maybe shell-related use cases can be migrated to shellwords standard library, like by creating Shellwords.shellstrip method or something.

#33 - 05/17/2016 08:44 AM - mrkn (Kenta Murata)

- Related to Feature #8110: Regex methods not changing global variables added

#34 - 05/19/2016 06:55 PM - naruse (Yui NARUSE)

Experimentally implemented an String#blank? with Intel STTNI (SSE 4.2; Nehalem or later)
https://github.com/ruby/ruby/commit/e6bc209abf81d53c2e3374dc52c2a128570c6055

#35 - 05/20/2016 12:23 AM - nobu (Nobuyoshi Nakada)

The instruction doesn't seem worth specializing, at least.

#36 - 05/20/2016 09:25 AM - naruse (Yui NARUSE)

- Related to Feature #12403: Optimise Regexp#match? added

#37 - 01/16/2018 03:03 PM - ana06 (Ana Maria Martinez Gomez)

I would also like to have blank? and present? in Ruby. matz (Yukihiro Matsumoto) says he doesn't see the benefit of having them, but I think they improve readability.

#38 - 03/22/2018 09:03 PM - busterb (Brent Cook)

This seems to be the number one time-wasters in my project (metasploit-framework), where nobody can agree if we should use it, shouldn't, whether it affects performance, behavior on nil, etc.

It would be great if this just got added to the standard library for nil and string. It's practically part of the Ruby standard anyway.

#39 - 06/04/2018 03:03 AM - sam.saffron (Sam Saffron)

matz (Yukihiro Matsumoto) ... is there any way we can revise this and act on it? I really want to kill off my "fast_blank" gem. The 2 decisions that need to be made are:

1. Will MRI accept String#blank? which returns exactly the same as String#strip.empty?
2. Will MRI accept changing #strip to be unicode space aware so it is consistent with Regexp?

#match? helps but it is not as fast as a native implementation.

```
sam@ubuntu rails % irb
irb(main):001:0> str = "\u202f"
=> " "
irb(main):002:0> str = "\u202f".strip.length
=> 1
irb(main):005:0> str = "\u202f".match?(/\A[[:space:]]*\z/)
=> true
```

The real world use case of this is slowly moving us to Ruby 3x3, by adding #blank? we can have an optimised implementation of #strip#empty? this is used widely in MRI codebase and other projects (non Rails including)

An alternative may be erasing #strip#empty? in insns or something but I wonder if this is both risky and simply adding #blank? is easier.

#40 - 06/04/2018 07:50 AM - nobu (Nobuyoshi Nakada)

As String#upcase and family are Unicode case aware now, so the second feels somewhat reasonable.

#41 - 07/13/2018 03:43 PM - shevegen (Robert A. Heiler)

Sam suggested the topic to be discussed at the next upcoming ruby
developer meeting here recently:

https://bugs.ruby-lang.org/issues/14861

I'll add a few comments to the issue here as a consequence.

I am mostly neutral to the whole issue at hand, though up to slightly in
disfavour of adding #blank? and respectively #present?.

It's not that I do not understand the use case; I simply don't think
the method names make a lot of sense. What is a "blank" string really?

If it is an empty string, then it should be called an empty string. But rails also considers an empty array to be "blank", so I am confused - see lateron in this comment.

I understand that we want to query the whitespace status via #blank? but then why not call it #whitespace? or something similar? And why does ActiveSupport use #blank? across several different classes with a different meaning/behaviour - such as on arrays?

So, my problem is primarily with the name, only secondarily the functionality in itself.

Note also that while active* is big, active* is not ruby as a whole. For core functionality, the reasoning should be "because lots of ruby people may need this or that" rather than "because it is used in active*, we should make it part of ruby", in my opinion. These two can often be synonymous but the reasoning should still come from a general user's perspective.

As for #present?, I have even less of an idea what this is supposed to do. I had a look at the docu for ActiveSupport:

http://guides.rubyonrails.org/active_support_core_extensions.html#blank-questionmark-and-present-questionmark

"The method present? is equivalent to !blank?."

I think that is also weird from a design point of view.

It's a bit like .select versus .reject, except just for a (forward or backwards) query. Would you have guessed that "#present?" is the opposite to "#blank?"? I would not have.

"The following values are considered to be blank in a Rails application"
" empty arrays and hashes, and"
" any other object that responds to empty? and is empty."

Is also a weird design choice. In the latter blank? is suddenly equivalent to .empty? for general objects - but we already have that method name for e. g. Strings. So I don't think this is a good design decision that Rails did, but that is my opinion and I am sure many rails people think it's a great design decision.

It's good that ruby is that flexible though.

As for #strip handling unicode, if it is possible I think it would be nice. People could then assume that calling .strip on a unicode string would remove e. g. newlines and whitespace on the ends/sides.

I don't know how difficult it would be to have this; guess Martin Dürst knows more here. Do all languages support this behaviour by the way? Perhaps some languages have no whitespace, like some symbols-based languages perhaps.

I should however had also add that I am not really that against the proposal. It's no real problem at all for me if it is added, so I am mostly neutral, only with a slight inching towards against the first part of the proposal. (The unicode extension is I think perfectly fine; I think it may have been better to split this issue into two separate ones though.)

#42 - 07/18/2018 06:24 AM - matz (Yukihiro Matsumoto)

If the requirement is adding a predicate method to check if a string contains only characters with space property (defined by Unicode), it's OK to add the method. I am not fully satisfied with the name blank? because it can cause confusion whether it checks a character in a string or all characters in a string. blank? is still acceptable though it's not the best.

Regarding present?, I am not going to agree.

Matz.

#43 - 07/20/2018 10:53 AM - nobu (Nobuyoshi Nakada)

I've proposed String#space_only?, like String#ascii_only?.

#44 - 07/22/2018 10:55 PM - sam.saffron (Sam Saffron)

> I've proposed String#space_only?, like String#ascii_only?.

I very much support #space_only?, it still fills the original goal of the ticket which is to eliminate the the fast_blank gem!