Ruby trunk - Feature #12786

String#casecmp?

09/24/2016 02:34 AM - rringler (Ryan Ringler)

Status:	Closed	
Priority:	Normal	
Assignee:	duerst (Martin Dürst)	
Target version:		

Description

Description

I would find String#casecmp? convenience method handy. I don't believe I've ever called String#casecmp without chaining #zero? to the result.

```
'abc'.casecmp?('ABC') #=> true
'abc'.casecmp?('DEF') #=> false
```

Related issues:

Related to Ruby trunk - Feature #14055: String#casecmp should use Unicode fol...	**Rejected**

Associated revisions

Revision ad619e02 - 11/28/2016 08:37 AM - duerst (Martin Dürst)

implement String/Symbol#casecmp? including Unicode case folding

- string.c: Implement String#casecmp? and Symbol#casecmp? by using String#downcase :fold for Unicode case folding. This does not include options such as :turkic, because these currently cannot be combined with the :fold option. This implements feature #12786.
- test/ruby/test_string.rb/test_symbol.rb: Tests for above.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@56912 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 56912 - 11/28/2016 08:37 AM - duerst (Martin Dürst)

implement String/Symbol#casecmp? including Unicode case folding

- string.c: Implement String#casecmp? and Symbol#casecmp? by using String#downcase :fold for Unicode case folding. This does not include options such as :turkic, because these currently cannot be combined with the :fold option. This implements feature #12786.
- test/ruby/test_string.rb/test_symbol.rb: Tests for above.

Revision 56912 - 11/28/2016 08:37 AM - duerst (Martin Dürst)

implement String/Symbol#casecmp? including Unicode case folding

- string.c: Implement String#casecmp? and Symbol#casecmp? by using String#downcase :fold for Unicode case folding. This does not include options such as :turkic, because these currently cannot be combined with the :fold option. This implements feature #12786.
- test/ruby/test_string.rb/test_symbol.rb: Tests for above.

Revision 56912 - 11/28/2016 08:37 AM - duerst (Martin Dürst)

implement String/Symbol#casecmp? including Unicode case folding

- string.c: Implement String#casecmp? and Symbol#casecmp? by using String#downcase :fold for Unicode case folding. This does not include options such as :turkic, because these currently cannot be combined with the :fold option. This implements feature #12786.

- test/ruby/test_string.rb/test_symbol.rb: Tests for above.

Revision 56912 - 11/28/2016 08:37 AM - duerst (Martin Dürst)

implement String/Symbol#casecmp? including Unicode case folding

- string.c: Implement String#casecmp? and Symbol#casecmp? by using String#downcase :fold for Unicode case folding. This does not include options such as :turkic, because these currently cannot be combined with the :fold option. This implements feature #12786.

- test/ruby/test_string.rb/test_symbol.rb: Tests for above.

Revision 7a480ae8 - 11/29/2016 03:31 PM - kazu

NEWS: Add String/Symbol#casecmp? [Feature #12786]

[ci skip]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@56932 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 56932 - 11/29/2016 03:31 PM - znz (Kazuhiro NISHIYAMA)

NEWS: Add String/Symbol#casecmp? [Feature #12786]

[ci skip]

Revision 56932 - 11/29/2016 03:31 PM - kazu

NEWS: Add String/Symbol#casecmp? [Feature #12786]

[ci skip]

Revision 56932 - 11/29/2016 03:31 PM - kazu

NEWS: Add String/Symbol#casecmp? [Feature #12786]

[ci skip]

Revision 56932 - 11/29/2016 03:31 PM - kazu

NEWS: Add String/Symbol#casecmp? [Feature #12786]

[ci skip]

History

#1 - 09/24/2016 02:53 AM - duerst (Martin Dürst)

Can you describe exactly what String#casecmp? method would do? Does it only compare for equality (after case folding)?

This proposal is interesting, because String#upcase/downcase/capitalize/swapcase now work for all of Unicode. But String#casecmp doesn't, because it would be a lot of additional work (and would depend on natural language) as to which string is larger or smaller.

But that doesn't apply to String#casecmp?, so it could be made to work rather easily with all of Unicode. Essentially, the implementation would be something like

```
class String
  def casecomp? (other)
    downcase(:fold) == other.downcase(:fold)
  end
end
```

#2 - 09/24/2016 02:57 AM - rringler (Ryan Ringler)

- File deleted (string_casecmp_.patch)

#3 - 09/24/2016 02:58 AM - rringler (Ryan Ringler)

- File string_casecmp_.patch added

#4 - 09/24/2016 03:02 AM - rringler (Ryan Ringler)

Martin Dürst wrote:

> Does it only compare for equality (after case folding)?

Yep. All it would do is check if the return value of #casecmp is equal to zero. I'm sure there are plenty of clever uses for #casecmp, but I've only ever used it for case-insensitive string comparision. I often find myself monkey-patching this method into String.

#5 - 09/24/2016 03:55 AM - rringler (Ryan Ringler)

- File deleted (string_casecmp_.patch)

#6 - 09/24/2016 03:55 AM - rringler (Ryan Ringler)

- File string_casecmp_.patch added

#7 - 09/24/2016 08:35 AM - rringler (Ryan Ringler)

Some thoughts on method naming...

- New method: #casecmp?
  - **PROS:** Nice symmetry with Fixnum's #<=> & #equal? methods. 'casecmp' seems to imply case-insensitivity (see String#casecmp or C's strcasecmp)
  - **CONS:** String#casecmp returns 0 for case-insensitive matches, which while not falsey, seems less truthy than 1 or -1.

```
# Proposed implementation
> 'abc'.casecmp?('abc') # true
> 'abc'.casecmp?('ABC') # false
> 'abc'.casecmp?('DEF') # false
> 'abc'.casecmp?(:abc)  # TypeError
```

- New method: #case_equal?, #case_insensitive_equal?, #insensitive_equal?, #iequal?
  - **PROS:** More expressive method name.
  - **CONS:** New method to learn. Method names without some flavor of 'insensitive' may be unintuitive.

```
# Proposed implementation
> 'abc'.case_equal?('abc') # true
> 'abc'.case_equal?('ABC') # false
> 'abc'.case_equal?('DEF') # false
> 'abc'.case_equal?(:abc)  # false
```

- Repurpose #eql?. Per the string.c method description: "Two strings are equal if they have the same length and content". Is 'content' case-insensitive?
  - **PROS:** Seems to align with the description of the method. #eql? is currently redundant with #==.
  - **CONS:** New context to learn. Backwards incompatible.

```
# Current implementation (Using ruby 2.3.1p112 (2016-04-26 revision 54768) [x86_64-darwin15])
> 'abc'.eql?('abc') # true
> 'abc'.eql?('ABC') # false
> 'abc'.eql?('DEF') # false
> 'abc'.eql?(:abc)  # false

> 'abc' == 'abc'    # true
> 'abc' == 'ABC'    # false
> 'abc' == 'DEF'    # false
> 'abc' == :abc     # false

# Proposed implemntation
> 'abc'.eql?('abc') # true
> 'abc'.eql?('ABC') # true
> 'abc'.eql?('DEF') # false
> 'abc'.eql?(:abc)  # false
```

I currently like #casecmp? or #eql? but am willing to be convinced otherwise.

#8 - 09/30/2016 05:41 AM - shevegen (Robert A. Heiler)

I think .eql? is not a good name because it chops away characters.

Imagine if we would have .sz or .sz? rather than .size. Or .lngth
rather than .length. :D

.casecmp? does chop away characters too but at least it is a bit
easier to read. Though the name could be .casecompare?() or

.equal() or .case_equal? - actually the last one may be better.

You gave some more examples for names too:

```
#case_equal?
#case_insensitive_equal?
#insensitive_equal?
#iequal?
```

I think .iequal? is not good for the same reasons; right now I
like .case_equal? the most but perhaps there may be even better
names, who knows.

Anyway I guess the name can be decided at a later point too - guess
you have to see whether matz considers the described behaviour
useful to have for ruby. If so then I am sure the name can be
easily chosen. :)

(You need to keep in mind that, once implemented and available,
it will eventually be used in ruby scripts so it should "fit"
into ruby itself too.)

#9 - 11/25/2016 08:40 AM - matz (Yukihiro Matsumoto)

casecmp? accepted.

Matz.

#10 - 11/25/2016 08:45 AM - duerst (Martin Dürst)

I gave an implementation in Ruby, but can somebody provide an equivalent patch in C?

#11 - 11/25/2016 08:47 AM - duerst (Martin Dürst)

- Status changed from Open to Feedback

I gave an implementation in Ruby, but can somebody provide an equivalent patch in C?

#12 - 11/26/2016 09:09 PM - rringler (Ryan Ringler)

Yukihiro Matsumoto wrote:

> casecmp? accepted.
>
> Matz.

Thank you!

#13 - 11/26/2016 09:13 PM - rringler (Ryan Ringler)

Martin Dürst wrote:

> I gave an implementation in Ruby, but can somebody provide an equivalent patch in C?

The attached patch implements #casecmp? by calling #casecmp and checking whether the return value is zero. I think this provides a clean separation of concerns (keeping all the comparison logic in #casecmp.)

#14 - 11/27/2016 01:53 AM - duerst (Martin Dürst)

Ryan Ringler wrote:

> The attached patch implements #casecmp? by calling #casecmp and checking whether the return value is zero. I think this provides a clean separation of concerns (keeping all the comparison logic in #casecmp.)

Thanks for the patch. "Separation of concerns" is in general a good idea, but in this case, it will lead to a suboptimal result. In particular, as discussed in https://bugs.ruby-lang.org/issues/12786#note-1, we can easily make sure that casecmp? works with all of Unicode, so that e.g.

```
assert_equal(true, 'äöü'.casecmp? 'ÄÖÜ')
```

works. But your patch doesn't do this. Making sure that casecmp itself works for all of Unicode is extremely difficult, because it needs full Unicode sorting.

Btw, this reminds me that we need options such as :turkic on casecmp?. The simplest set of tests would be:

```
assert_equal(true, 'I'.casecmp? 'i')
assert_equal(true, 'I'.casecmp? 'ı', :turkic)
assert_equal(true, 'i'.casecmp? 'İ', :turkic)
```

So my implementation above would change to

```
class String
  def casecmp? (other, *args)
    downcase(:fold, *args) == other.downcase(:fold, *args)
  end
end
```

Given that casecmp? can easily be made to work across Unicode soon, whereas for casecmp, this may take another few years or so, I wonder whether it might not be better to use a slightly different name, e.g. case_equal? or so.

#15 - 11/27/2016 03:23 AM - nobu (Nobuyoshi Nakada)

String#casecmp also should have those options, I think.

#16 - 11/27/2016 04:02 AM - rringler (Ryan Ringler)

Nobuyoshi Nakada wrote:

> String#casecmp also should have those options, I think.

I agree, and feel strongly that the logic to support those options should live there. Someone smarter than I should probably make the changes Martin has suggested to String#casecmp; I just want a predicate method that indicates whether String#casecmp indicates a match or not.

#17 - 11/27/2016 05:32 AM - duerst (Martin Dürst)

Nobuyoshi Nakada wrote:

> String#casecmp also should have those options, I think.

I agree that once String#casecmp supports full Unicode, it should have these options (and more, because for Unicode sorting, you need/can have a lot more options than for simple casing operations). But I think we can add these options once we need them.

#18 - 11/28/2016 08:30 AM - duerst (Martin Dürst)

- Assignee set to duerst (Martin Dürst)

- Status changed from Feedback to Open

#19 - 11/28/2016 08:37 AM - duerst (Martin Dürst)

- Status changed from Open to Closed

Applied in changeset r56912.

---

implement String/Symbol#casecmp? including Unicode case folding

- string.c: Implement String#casecmp? and Symbol#casecmp? by using String#downcase :fold for Unicode case folding. This does not include options such as :turkic, because these currently cannot be combined with the :fold option. This implements feature #12786.
- test/ruby/test_string.rb/test_symbol.rb: Tests for above.

#20 - 11/29/2016 03:42 AM - rringler (Ryan Ringler)

Thank you for incorporating this!

I see the changeset implements the new method by calling String#downcase on both strings and compares the results for equality. While this does allow unicode support in advance of String#casecmp, it sacrifices the performance of character-by-character comparison that #casecmp provides. It's also going to lead to some inconsistent behavior:

```
'äöü'.casecmp('ÄÖÜ').zero? # false
'äöü'.casecmp?('ÄÖÜ')      # true
```

I hope this was considered, and perhaps when String#casecmp provides better support for unicode we can update #casecmp? to use it.

#21 - 11/29/2016 10:35 AM - duerst (Martin Dürst)

Ryan Ringler wrote:

> I see the changeset implements the new method by calling String#downcase on both strings and compares the results for equality.

Yes, this is as proposed at https://bugs.ruby-lang.org/issues/12786#note-1.

> While this does allow unicode support in advance of String#casecmp, it sacrifices the performance of character-by-character comparison that #casecmp provides.

Yes, I realized this as I implemented it. For the moment, my take is "better slow than fast and wrong".

> It's also going to lead to some inconsistent behavior:
>
> ```
> 'äöü'.casecmp('ÄÖÜ').zero? # false
> 'äöü'.casecmp?('ÄÖÜ')      # true
> ```

Yes. I added some additional explanations to the documentation today to make this clearer.

> I hope this was considered, and perhaps when String#casecmp provides better support for unicode we can update #casecmp? to use it.

It would definitely be great to implement Unicode sorting, and in that case, we will make sure that casecmp and casecmp? are consistent again. But that will be a LOT of work. Also, please note that implementing Unicode sorting will make casecmp slower, too.

#22 - 10/26/2017 01:57 AM - nobu (Nobuyoshi Nakada)

- Related to Feature #14055: String#casecmp should use Unicode folding added

Files

string_casecmp_.patch	4.97 KB	09/24/2016	rringler (Ryan Ringler)