Ruby - Feature #10084

Add Unicode String Normalization to String class

07/23/2014 10:04 AM - duerst (Martin Dürst)

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

Description

Unicode string normalization is a frequent operation when comparing or normalizing strings.

This should be available directly on the String class.

The proposed syntax is:

```
'string'.normalize       # normalize 'string' according to NFC (most frequent on the Web)
'string'.normalize :nfc  # normalize 'string' according to NFC; :nfd, :nfkc, :nfkd also usable
'string'.nfc            # shorter variant, but maybe too many methods
```

There are several "unofficial" but convenient normalization variants that could be offered, e.g.:

'string'.normalize :mac # use MacIntosh file system normalization variant

Implementations are already available in pure Ruby (easy for other Ruby implementations; e.g. eprun: https://github.com/duerst/eprun) and in C (unf,…, http://bibwild.wordpress.com/2013/11/19/benchmarking-ruby-unicode-normalization-alternatives/)

Related issues:

Related to Ruby - Feature #2034: Consider the ICU Library for Improving and E...	**Rejected**	
Related to Ruby - Bug #7267: Dir.glob on Mac OS X returns unexpected string e...	**Closed**	**11/02/2012**
Related to Ruby - Feature #9111: Encoding-free String comparison	**Open**	**11/14/2013**

Associated revisions

Revision 96245d6dd37557acd9a066491db6c80b125f0676 - 09/23/2014 07:08 AM - duerst (Martin Dürst)

tool/downloader.rb: added Downloader.download_if_modified_since
to reduce downloads of large files that change only rarely.
[ruby-core:65164] [CommonRuby - Feature #10084]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@47691 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 96245d6d - 09/23/2014 07:08 AM - duerst (Martin Dürst)

tool/downloader.rb: added Downloader.download_if_modified_since
to reduce downloads of large files that change only rarely.
[ruby-core:65164] [CommonRuby - Feature #10084]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@47691 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 3f8211a1133a401dd8e4cb6ca56b0dae6a09793a - 10/19/2014 02:34 PM - nagachika (Tomoyuki Chikanaga)

- lib/unicode_normalize.rb: (unicode_normalize!): change method name.
  catch up the method name change at r48014. [Feature #10084]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@48027 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 3f8211a1 - 10/19/2014 02:34 PM - nagachika (Tomoyuki Chikanaga)

- lib/unicode_normalize.rb: (unicode_normalize!): change method name.
  catch up the method name change at r48014. [Feature #10084]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@48027 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 07/23/2014 10:10 AM - duerst (Martin Dürst)

- Related to Feature #2034: Consider the ICU Library for Improving and Expanding Unicode Support added

#2 - 07/23/2014 10:11 AM - duerst (Martin Dürst)

- Related to Bug #7267: Dir.glob on Mac OS X returns unexpected string encodings for unicode file names added

#3 - 07/23/2014 10:11 AM - duerst (Martin Dürst)

- Related to Feature #9111: Encoding-free String comparison added

#4 - 07/23/2014 10:11 AM - duerst (Martin Dürst)

- File Normalization.pdf added

#5 - 07/23/2014 01:44 PM - nobu (Nobuyoshi Nakada)

What will happen for a non-unicode string, raising an exception?

#6 - 07/24/2014 06:58 AM - duerst (Martin Dürst)

Nobuyoshi Nakada wrote:

> What will happen for a non-unicode string, raising an exception?

This is a very good question. I'm okay with whatever Matz and the community think is best.

There are many potential approaches. In general, these will be:

1. Make the operation a no-op.
2. Convert to UTF-8, normalize, then convert back.
3. Implement normalization directly in the encoding.
4. Raise an exception.

There is also the question of what a "non-unicode" or "unicode" string is.

UTF-8 is the preferred way to handle Unicode in Ruby, and is where normalization is really needed and will be used.

For the other encodings, unless we go with 1) or 4), the following considerations apply.

UTF8-Mac, UTF8-DoCoMo, UTF8-KDDI and UTF8-Softbank are essentially UTF-8 but with slightly different character conversions. For these encodings, the easiest thing to do is force_encoding to UTF-8, normalize, and force_encoding back. A C-level implementation may not actually need force_encoding, but a Ruby implementation does. There are some questions about what normalizing UTF8-Mac means, so that may have to be treated separately. The DoCoMo/KDDI/Softbank variants are mostly about emoji, which as far as I know are not affected by normalization.

Then there are UTF-16LE/BE and UTF-32LE/BE. For these, it depends on the implementation. A Ruby-level implementation (unless very slow) may want to convert to UTF-8 and back. A C-level implementation may not need to do this.

Then there is also GB18030. Conversion to UTF-8 and back seems to be the best solution. Doing normalization directly in GB18030 will need too much data.

For other, truely non-unicode encodings, implementing noramlization directly in the encoding would mean the following: Analyze to what extent the normalization applies to the encoding in question, and apply this part. As an example, '①'.nfkc produces '1' in UTF-8, it could do the same in Windows-31J. The analysis might take some time (but can be automated), and the data needed for each encoding would mostly be just very small.

#7 - 07/26/2014 06:18 AM - matz (Yukihiro Matsumoto)

First of all, I don't think normalize is the best name.
I propose unicode_normalize instead, since this normalization is sort of unicode specific.

We still need to define the detail.

It should raise an exception for non Unicode strings. It shouldn't convert to UTF-8 implicitly inside.

Matz.

#8 - 07/28/2014 05:32 AM - duerst (Martin Dürst)

copying notes from 2014/7/26 developer's meeting (Google docs):

Proposed method names by Matz: unicode_normalize or normalize_kd,... (not too short)
How to deal with non-Unicode encodings: Matz: raise Exception

Other than UTF-8: UTF8-Mac: return type should be UTF-8, or deal with it as legacy (not really Unicode). UTF8-DoCoMo,..? Yui should decide.
UTF-16/32: Needed data,... differs by whether implementation is internal (C) or pure Ruby.
Todo (for eprun): measure load time, compare with unf, avoid Module Normalize

require “unicode_normalize”
method name: String#unicode_normalize(form)
form: :nfc, :nfd, :nfkc, :nfkd
encodng: UTF-32BE/LE, UTF-16BE/LE, UTF-8
allow UTF8-MAC is confusing.

#9 - 09/20/2014 05:46 AM - duerst (Martin Dürst)

- Target version set to Ruby 2.2.0

#10 - 09/20/2014 07:13 AM - duerst (Martin Dürst)

- Assignee set to nobu (Nobuyoshi Nakada)

For Nobu:
These are the three files that should be downloaded during build time
(with "If-Modified-Since" to avoid repeated downloads of the same large files):

http://www.unicode.org/Public/UCD/latest/ucd/UnicodeData.txt
http://www.unicode.org/Public/UCD/latest/ucd/CompositionExclusions.txt
http://www.unicode.org/Public/UCD/latest/ucd/NormalizationTest.txt

The first two will be used for code generation, the last one for testing.

For file locations, I suggest
enc/unicode/UnicodeData.txt
enc/unicode/CompositionExclusions.txt
test/unicode-normalize/NormalizationTest.txt (needs new directory)
but I leave the decision to you (or somebody else).

I set you as assignee; please set it to me when this is done or when you have a question.

#11 - 09/23/2014 07:08 AM - duerst (Martin Dürst)

- Status changed from Open to Closed

- % Done changed from 0 to 100

Applied in changeset ruby-trunk:r47691.

---

tool/downloader.rb: added Downloader.download_if_modified_since
to reduce downloads of large files that change only rarely.
[ruby-core:65164] [CommonRuby - Feature #10084]

#12 - 09/23/2014 07:13 AM - duerst (Martin Dürst)

- Status changed from Closed to Open

Changeset ruby-trunk:r47691 only completes about 5% or 10% of this bug, therefore reopening.

#13 - 10/07/2014 11:00 AM - duerst (Martin Dürst)

- Assignee changed from nobu (Nobuyoshi Nakada) to matz (Yukihiro Matsumoto)

This feature is going to add one or more methods to class String (String#unicode_normalize and probably String#unicode_normalize! and String#unicode_normalized?).

The implementation also internally needs various additional methods and constants that an end user should not ever want or need to use. Just adding them to class String is the easiest solution, but this may confuse a user when e.g. calling String.instance_methods(false). In the current implementation, these methods and constants are put in module Normalize (see https://github.com/duerst/eprun/blob/master/lib/normalize.rb).

In order to proceed with the implementation of this feature, I'd like to get advice from Matz (and others) on the best alternative. I have thought through the following alternatives:

1. Use a standalone module (probably better to change the name, e.g. to UnicodeNormalizeImplementation or so)

2. Use a module inside String, e.g. String::Normalize. Advantage: module name can be shorter, because local.

3. Use an anonymous module. This is possible, but it requires that all the related code and data is in the same physical file, which restricts potential memory optimizations. Also, it requires that all the code is re-written e.g. replacing 'def' with 'define_method', which will look rather clumsy.

4. Just add the necessary methods and constants to String, but use longer, more explicit names. This should be slightly faster, because currently many of the methods take an explicit string parameter, but this would just be the receiver. We can also make the methods private to reduce user temptation.

5. Use a refinement. The advantages are that this can be distributed over more than one file, and that we can directly call the methods on Strings (see 4). The disadvantages are that we still need a public module as the refinement container.

I personally would like to avoid 3) if at all possible. I don't have much preferences among the other solutions. There may also be other solutions that I haven't thought about yet.

I would like to get Matz's preference(s) as soon as possible to proceed with the implementation. Any advice from others, e.g. with respect to similar cases, performance tradeoffs, other ideas, and so on, are also greatly appreciated.

#14 - 10/13/2014 12:49 AM - duerst (Martin Dürst)

- Assignee changed from matz (Yukihiro Matsumoto) to duerst (Martin Dürst)

Not getting any feedback on implementation details, I'm assuming that nobody cares too much, and will therefore proceed. I have tried a refinement (proposal 5); I didn't see any effects on performance. But using a refinement would make it more difficult to backport this to earlier versions or make it available as a gem.

I'm therefore going to take the easiest way forward and use solution 1), with a module name of UnicodeNormalize (exactly corresponding to primary method name on string). If anybody still has comments, please don't hesitate to add them here, so that we can discuss them.

#15 - 10/19/2014 02:34 PM - nagachika (Tomoyuki Chikanaga)

- Status changed from Open to Closed

Applied in changeset ruby-trunk:r48027.

- lib/unicode_normalize.rb: (unicode_normalize!): change method name.
  catch up the method name change at r48014. [Feature #10084]

#16 - 10/20/2014 01:25 AM - nagachika (Tomoyuki Chikanaga)

- Status changed from Closed to Assigned

Sorry, I accidentally close this ticket.

#17 - 10/21/2014 07:34 AM - naruse (Yui NARUSE)

```
class Unicode < self
def self.download(name, *rest)
super("http://www.unicode.org/Public/UCD/latest/ucd/#{name}", name, *rest)
end
end
```

"latest" is not acceptable because released Ruby's table must be a specific version.

Moreover generated lib/unicode_normalize/tables.rb is only 200MB. How about committing it to the repo like other conversion tables?

#18 - 10/21/2014 08:32 AM - duerst (Martin Dürst)

Hello Yui,

On 2014/10/21 16:34, naruse@airemix.jp wrote:

> Issue #10084 has been updated by Yui NARUSE.
>
> ```
> class Unicode < self
> def self.download(name, *rest)
> super("http://www.unicode.org/Public/UCD/latest/ucd/#{name}", name, *rest)
> end
> end
> ```

> "latest" is not acceptable because released Ruby's table must be a specific version.

[I disagree with this policy, but I will of course respect it until I
can convince others that a more dynamic policy is better.]

> Moreover generated lib/unicode_normalize/tables.rb is only 200MB. How about committing it to the repo like other conversion tables?

I came to the same conclusion, and I have just done so at r48072.

Nobu and I have tried to make the update of the Unicode data files
automatic and unobtrusive, but we had to find out that it is difficult
to get all of the following:

- Use already downloaded Unicode data files if no network connection.
- Check for updates dynamically.
- Make sure that this happens regularly (I think currently it is done
  with "make up", but not everybody packaging Ruby is using "make up").

I hope we can try to keep the makefile logic for automatic update of
Unicode data files and lib/unicode_normalize/tables.rb but change it so
that it is triggered only on request.

Regards,   Martin.

> ---
>
> Feature #10084: Add Unicode String Normalization to String class
> https://bugs.ruby-lang.org/issues/10084#change-49562
>
> - Author: Martin Dürst
> - Status: Assigned
> - Priority: Normal
> - Assignee: Martin Dürst
> - Category:
> - Target version: Ruby 2.2.0
>
> ---
>
> Unicode string normalization is a frequent operation when comparing or normalizing strings.
>
> This should be available directly on the String class.
>
> The proposed syntax is:
>
> ```
> 'string'.normalize        # normalize 'string' according to NFC (most frequent on the Web)
> 'string'.normalize :nfc   # normalize 'string' according to NFC; :nfd, :nfkc, :nfkd also usable
> 'string'.nfc              # shorter variant, but maybe too many methods
> ```
>
> There are several "unofficial" but convenient normalization variants that could be offered, e.g.:
>
> ```
> 'string'.normalize :mac   # use MacIntosh file system normalization variant
> ```
>
> Implementations are already available in pure Ruby (easy for other Ruby implementations; e.g. eprun: https://github.com/duerst/eprun) and in C (unf,..., http://bibwild.wordpress.com/2013/11/19/benchmarking-ruby-unicode-normalization-alternatives/)
>
> ---Files--------------------------------
> Normalization.pdf (576 KB)

#19 - 10/21/2014 08:56 AM - Eregon (Benoit Daloze)

Yui NARUSE wrote:

>> class Unicode < self
>> def self.download(name, *rest)
>> super("http://www.unicode.org/Public/UCD/latest/ucd/#{name}", name, *rest)
>> end
>> end
>
> "latest" is not acceptable because released Ruby's table must be a specific version.
>
> Moreover generated lib/unicode_normalize/tables.rb is only 200MB. How about committing it to the repo like other conversion tables?

You probably meant 200 KB.

#20 - 10/22/2014 10:53 AM - duerst (Martin Dürst)

- Status changed from Assigned to Closed

Closing because the tests (test/test_unicode_normalize.rb) run correctly
(45 tests, 531826 assertions). There are some problems remaining with the
build logic, but I'll create separate bugs for them.

#21 - 12/23/2021 11:41 PM - hsbt (Hiroshi SHIBATA)

- Project changed from 14 to Ruby

- Target version deleted (Ruby 2.2.0)

Files

Normalization.pdf	576 KB	07/23/2014	duerst (Martin Dürst)