Ruby master - Feature #9111
Encoding-free String comparison
11/14/2013 10:15 PM - sawa (Tsuyoshi Sawada)

| Status:    | Open         |
| Priority:  | Normal       |
| Assignee:  |              |
| Target version: |          |

**Description**

```ruby
=end
currently, strings with the same content but with different encodings count as different strings. This causes strange behaviour as
below (noted in stackoverflow question
```n
```ruby
[128].pack("C")    #=> "\x80"
[128].pack("C")  == "\x80"  #=> false
```

Since `[128].pack("C")` has the encoding ASCII-8BIT and "\x80" (by default) has the encoding UTF-8, the two strings are not equal.

Also, comparison of strings with different encodings may end up with a messy, unintended result.

I suggest that the comparison String#<=> should not be based on the respective encoding of the strings, but all the strings should be
internally converted to UTF-8 for the purpose of comparison.

=end

**Related issues:**

Related to CommonRuby - Feature #10084: Add Unicode String Normalization to S...

Closed 07/23/2014

**History**

**#1 - 11/14/2013 11:17 PM - nobu (Nobuyoshi Nakada)**

sawa (Tsuyoshi Sawada) wrote:

> I suggest that the comparison String#<=> should not be based on the respective encoding of the strings, but all the strings should be internally converted to UTF-8 for the purpose of comparison.

It's unacceptable to always convert all strings to UTF-8, should restrict to comparison with an ASCII-8BIT string.

**#2 - 11/15/2013 12:04 AM - sawa (Tsuyoshi Sawada)**

Following nobu's suggestion, I came up with the following several possibilities:

When two strings with different encodings are to be compared by String#<=>, then one of the following options should be taken:

- Raise a Warning message
- Raise an error
- Convert one of the strings to the other one.

I am not sure which option would be the best, but feel the feature should not be left as is now.

**#3 - 11/15/2013 05:20 AM - Hanmac (Hans Mackowiak)**

What about strings with the same encoding, but different content, but that is turned the same?

like "à" can be made from "a" + '̀' somehow, should they also treated as equal?

**#4 - 11/15/2013 02:41 PM - sawa (Tsuyoshi Sawada)**

Hanmac: "à" can be made from "a" + '̀'

Treating them the same is too much, I think. There are various marking methods. For example, à would have a different marking in TeX. Assuming them equal is going too much. They should be treated differently.
#5 - 11/15/2013 05:15 PM - Hanmac (Hans Mackowiak)

its not about treating "â" or "a" the same as "ä" but there is a way to clue the chars together

i think thats also a reason for [http://api.rubyonrails.org/classes/String.html#method-i-mb_chars](http) ?

i found another interesting gems [http://rubygems.org/gems/unicode_utils](http) with that is also possible to do something like this: "â".upcase => "Ä"

there is another page about combining character: [http://sbp.so/supercombiner](http)

---

#6 - 11/21/2013 04:35 PM - naruse (Yui NARUSE)

Hanmac (Hans Mackowiak) wrote:

> what about strings with the same encoding, but different content, but that is turned the same?
> like "â" can be maked from "a" + "¨" somehow, should they also treated as equal?

The standard practice is  \text{NFD}("â") == \text{NFD}("a" + "¨")
To NFD, you can use some libraries, see also [http://bibwild.wordpress.com/2013/11/19/benchmarking-ruby-unicode-normalization-alternatives/](http)

---

#7 - 07/23/2014 10:11 AM - duerst (Martin Dürst)

- Related to Feature #10084: Add Unicode String Normalization to String class added