Ruby trunk - Feature #12275

String unescape

04/12/2016 07:03 PM - asnow (Andrew Bolshov)

Status:	Closed
Priority:	Normal
Assignee:	tad (Tadashi Saito)
Target version:	

Description

I think it will be usefull to have function that convert input string as it was written in prime qouted string or in double qouted string. It's part of metaprogramming.
Example:

```
class String

# Create new string like it will be writed in qoutes. Optional argument define type of qouting use
d: true - prime qoute, false - double qoute. Default is double qoute.
  def unescape prime = false
    eval( prime ? "'#{self}'" : "\"#{self}\"" )
  end
end

"\\\t".unescape # => "\t"
```

Other requests:
http://www.rubydoc.info/github/ronin-ruby/ronin-support/String:unescape
http://stackoverflow.com/questions/4265928/how-do-i-unescape-c-style-escape-sequences-from-ruby
http://stackoverflow.com/questions/8639642/best-way-to-escape-and-unescape-strings-in-ruby

Realized
http://www.rubydoc.info/github/ronin-ruby/ronin-support/String:unescape

Related issues:

Related to Ruby trunk - Feature #12419: Improve String#dump for Unicode outpu...	**Closed**

Associated revisions

Revision bbec11d3 - 12/14/2017 08:47 AM - tadd

Implement String#undump to unescape String#dump-ed string
[Feature #12275] [close GH-1765]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@61228 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 61228 - 12/14/2017 08:47 AM - tad (Tadashi Saito)

Implement String#undump to unescape String#dump-ed string
[Feature #12275] [close GH-1765]

Revision 61228 - 12/14/2017 08:47 AM - tadd

Implement String#undump to unescape String#dump-ed string
[Feature #12275] [close GH-1765]

Revision 61228 - 12/14/2017 08:47 AM - tadd

Implement String#undump to unescape String#dump-ed string
[Feature #12275] [close GH-1765]

History

#1 - 04/12/2016 07:04 PM - asnow (Andrew Bolshov)

- Description updated

#2 - 04/12/2016 07:05 PM - asnow (Andrew Bolshov)

- Description updated

#3 - 04/20/2016 04:01 AM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Feedback

We looked at this ticket on this month's developer meeting. I then started to think that the "escape" you refer to is not that concrete.

Unescaping cannot work out without escaping. In ruby, there already is a method called String#dump. Is this what you want to negate?

```
irb(main):001:0> puts "\u5b57".encode('CP932').dump
"\x8E\x9A"
=> nil
```

#4 - 05/12/2016 03:43 PM - asnow (Andrew Bolshov)

I think yes, inverse of String#dump. I have user inputed string without qoutes, but it don't metter much.

#5 - 05/13/2016 06:44 AM - shyouhei (Shyouhei Urabe)

- Status changed from Feedback to Open

Thank you. That makes sense to me because String#dump has no corresponding undump method now.

#6 - 07/19/2016 06:00 AM - matz (Yukihiro Matsumoto)

String#undump sounds reasonable. If someone implement, it's OK to add.

Matz.

#7 - 07/19/2016 08:58 AM - duerst (Martin Dürst)

- Related to Feature #12419: Improve String#dump for Unicode output (from "\u{130}" to "\u0130") added

#8 - 11/18/2017 01:27 PM - tad (Tadashi Saito)

Hi, I'm working on this feature for several months.

First of all, I began to implement this as a gem.
https://github.com/tadd/string_undump
https://github.com/tadd/string_undump/blob/master/ext/string_undump/string_undump.c

Comments welcomed. I'll write a patch for trunk soon, as the next step.

#9 - 11/19/2017 10:11 AM - duerst (Martin Dürst)

I think rather than using true/false to distinguish single and double quotes, it would be better to have a keyword parameter, such as quotes: :single (and quotes: :double, but that would be default).

Also, "prime quote" isn't used widely. Please check e.g. "prime quote" and "single quote" on your favorite search engine. In addition, U+2032 (′, PRIME) is a different character. (The official name of U+0027 is APOSTROPHE.)

Also, please think about encodings. Some people may want all non-ASCII characters escaped, but others may not want that at all.

#10 - 11/24/2017 07:05 AM - tad (Tadashi Saito)

Thank you for your comments.

> I think rather than using true/false to distinguish single and double quotes, it would be better to have a keyword parameter, such as quotes: :single (and quotes: :double, but that would be default).

I think we can forget about arguments (i.e. additional quotes), because current implementation never uses eval() internally.

My String#undump takes no argument just like:

```
'"\u00FC"'.undump #=> "ü"
```

I'll write detailed specs when I submit a patch. Basically I focused to does inverse of String#dump.

> Also, please think about encodings. Some people may want all non-ASCII characters escaped, but others may not want that at all.

Unfortunately, I couldn't understand your concern. I think we're discussing about unescaping/undumping, not escaping.

Note that String#dump already escapes all of non-ASCII characters, so I'm trying to unescape them all with undump.

#11 - 11/24/2017 08:04 AM - duerst (Martin Dürst)

tad (Tadashi Saito) wrote:

> > Also, please think about encodings. Some people may want all non-ASCII characters escaped, but others may not want that at all.
>
> Unfortunately, I couldn't understand your concern. I think we're discussing about unescaping/undumping, not escaping.
> Note that String#dump already escapes all of non-ASCII characters, so I'm trying to unescape them all with undump.

Thanks for your explanation. I was confused.

Still, there is the question of what the encoding of the result of #unescape should be.

#12 - 11/25/2017 08:20 AM - tad (Tadashi Saito)

> Still, there is the question of what the encoding of the result of #unescape should be.

Indeed. It is one of few things that I'm still worried about.

For now, undump inherits receiver's encoding:

```
"abc".encode('euc-jp').undump.encoding #=> #<Encoding:EUC-JP>
```

But it may cause some inconvenient errors like:

```
utf8 = "\xE3\x81\x82".force_encoding('utf-8')
dumped = utf8.dump.encode('ascii') # we can treat dumped string as ASCII
dumped.valid_encoding? #=> always true, of course
dumped.undump #=> RangeError: 12354 out of char range
```

dump-ed string may contain any codepoints without original encoding information basically,
and this situation reminds me about Integer#chr(encoding).
Then undump may needs an argument too, to specify encoding of result string, I think.

(Of course dumped.force_encoding('utf-8') before undump solves this problem, but I feel it's little redundant.)

Any thoughts about this?

Although this is another topic, I think that the name of this new method is confirmed as
#undump (not #unescape) by @matz. Please see https://bugs.ruby-lang.org/issues/12275#note-6
and below. (I believe it's a good name because it reminds its spec clearly.)

#13 - 11/27/2017 07:58 PM - tad (Tadashi Saito)

- File v1.patch added

- File benchmark.rb added

Sorry for late, I implemented #undump as v1.patch based on my "string_undump" gem.
Please see https://github.com/ruby/ruby/pull/1765 also.

Spec

Roughly speaking, my implementation follows steps below:

1. If self is wrapped with double quote, just ignore them
2. Parse self and produce new string with concatenating character
    1. If escaped character (begins with backslash) found, unescape and add it to new string
    2. Otherwise, just add the character to the new string
3. Return the produced string

Note that this method does not require the wrapping of double quotes. It will be a help
for the cases such as in the initial proposal like "\\\t".undump .

Supported escaping formats are:

- Backslash itself
    - \\

- Double quote after backslash
  - \" yields double quote itself
- One ASCII character after backslash
  - \n \r \t \f \v \b \a \e
- "u" after backslash (Unicode)
  - \uXXXX form
  - \u{XXXXX} form (number of hex digits is variable)
- "x" and two hex digits after backslash
  - \xXX form
- "#$", "#@" or "#{" after backslash
  - These are embedded-Ruby-variable-like strings

I was careful to cover all escaping cases in String#dump so that s.dump.undump == s
is true as possible. Unfortunately, there are some limitations that shown below.

Testing

I added some testcases in test/ruby/test_string.rb
https://github.com/ruby/ruby/pull/1765/files#diff-25eb856a893dbc53c562f6865b215083
and they passes of course.

Another testcases that based on the original gems also passed.
https://gist.github.com/tadd/634b6e4b09b6dfe7c8b97bca138d31ec

Furthermore, at the RubyKaigi of this year, I knew about AFL (American Fuzzy Lop).
http://lcamtuf.coredump.cx/afl/
(I was fortunate to know that. Thank you shyouhei!)

It can tease my implementation. I checked my original gem (string_undump 0.1.0) with AFL 2.36b,
then I confirmed that:

- It did not cause SEGV during one night, with (about) 9 million times execution
- It did not cause roundtrip error during one night, with (about) 10 million times execution
  - s == s.dump.undump always true
  - I ran it in UTF-8 environment

Performance

It may be a boring result, but I'll also mention about performance. With really-naive
benchmark, undump is about 9 times faster than eval(string).
See and try attached benchmark.rb file, then feel free to experience Ruby 3x3x3 now...

Limitations

Sorry, some limitations exist on current implementation.

- Can't undump non ASCII-compatible string
  - '"abc"'.encode('utf-16le').undump yields Encoding::CompatibilityError for now
  - This is simply due to my lack of impl knowledge. Advice welcomed
- Can't undump dump-ed string correctly that is produced from non ASCII-compatible string
  - String#dump adds .force_encoding("encoding name here") at the end of dump-ed string, but String#undump doesn't parse this. Please check code below:

```
s = '"abc"'.encode('utf-16le')
puts s.dump #=> "a\x00b\x00c\x00".force_encoding("UTF-16LE")
s == s.dump.undump #=> false
```

- I believe this is rare case, and it's convenient enough even in the present situation
- But of course, I will not commit the patch if this limitation is not acceptable

Future work

- Improve support for non ASCII-compatible encodings (eliminate limitations above)
- Optimization for single-byte-optimizable string

Conclusion

I implemented #undump to be "someone" matz said. The code

- covers most practical cases of dump treats
- is enough safe from SEGV
- runs far faster from eval()

but some limitations still exist.

Any comments?

#14 - 11/28/2017 06:37 AM - tad (Tadashi Saito)

- Assignee set to tad (Tadashi Saito)

- Status changed from Open to Assigned

#15 - 12/03/2017 11:07 AM - tad (Tadashi Saito)

A few days ago, I attended at Ruby developers' meeting.
We concluded that the implementation is immature, so I need to improve in several points before commit.

- Encoding of a undumped string which including \uXXXX before undump should be UTF-8 automatically
- '"...".force_encoding("...")' form should be parsed
- self must be wrapped with double quotes
  - We need strict handling to clarify the spec

Improvements must be done in a week or so, then I'll require code reviewing.
After that, I'll mention to the 2.5 release manager, naruse, to get approval to check in.

Yes, I have to hurry...!

#16 - 12/09/2017 06:05 PM - tad (Tadashi Saito)

- File v2.patch added

- File benchmark2.rb added

I updated patch as v2.patch to satisfy 3 points that mentioned in note-15.
(Also https://github.com/ruby/ruby/pull/1765 is updated too.)

I also attached a simple benchmarking script as benchmark2.rb to check performance of newly-supported "...".force_encoding("...") form.

Can anyone review this patch? Or naruse (Yui NARUSE), do you want to nominate somebody?

#17 - 12/13/2017 08:44 PM - tad (Tadashi Saito)

- File v3.patch added

Thanks to shyouhei, mame, and especially naruse, I was able to brush up the patch.
v3.patch is attached. Improvements are diverse.

Spec change:

- use RuntimeError instead of ArgumentError for invalid formed (self) string
  - no arguments are given for this method... :(
- explicitly reject string that contains:
  - non-ascii character
  - NUL \0 character
  - (note that dumped strings do not contain above)

Bug fix:

- reject string that contains double quote in double quotes, like """"""
- prevent compiler's warnings/errors
  - cast explicitly from unsigned long to int
  - remove needless "const"

Misc:

- fix styles
- add more tests for invalid escaping
- remove needless logic
- adjust unescaped expression to parse.y

I'll take a short nap while praying so that AFL will not catch the worm...

#18 - 12/14/2017 08:47 AM - Anonymous

- Status changed from Assigned to Closed

Applied in changeset trunk|r61228.

Implement String#undump to unescape String#dump-ed string
[Feature #12275] [close GH-1765]

#19 - 12/14/2017 08:52 AM - tad (Tadashi Saito)

I committed this under the approval of naruse (Yui NARUSE) . https://github.com/ruby/ruby/pull/1765#pullrequestreview-83409358
Thanks a lot.

Files

benchmark.rb	193 Bytes	11/27/2017	tad (Tadashi Saito)
v1.patch	8.95 KB	11/27/2017	tad (Tadashi Saito)
benchmark2.rb	315 Bytes	12/09/2017	tad (Tadashi Saito)
v2.patch	12.1 KB	12/09/2017	tad (Tadashi Saito)
v3.patch	12.9 KB	12/13/2017	tad (Tadashi Saito)