Ruby trunk - Feature #14624

#{nil} allocates a fresh empty string each time

03/23/2018 03:49 AM - bumblingbear (Dillon Welch)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

This causes a bunch of unnecessary string allocations in the following scenario: "#{'rails' unless boolean_condition} is great". Each time this line is called when boolean_condition is true, it evaluates to nil and when nil is interpolated into a string it allocates an empty string. Ideally, the behavior of nil.to_s would reference a frozen empty string that wouldn't need to be reallocated each time.

History

#1 - 03/23/2018 09:22 AM - Hanmac (Hans Mackowiak)

i have a problem with that, what if someone might do:

```
x = obj.to_s
x << obj2.to_s
```

then it might be nil, return a frozen string and it would do a RuntimeError

better imo would be that if "#{nil}" would do a check if to_s is overwritten or redefined and if not do other magic

#2 - 03/27/2018 02:50 AM - bumblingbear (Dillon Welch)

That sounds fine to me!

Hanmac (Hans Mackowiak) wrote:

> i have a problem with that, what if someone might do:
>
> ```
> x = obj.to_s
> x << obj2.to_s
> ```
>
> then it might be nil, return a frozen string and it would do a RuntimeError
>
> better imo would be that if "#{nil}" would do a check if to_s is overwritten or redefined and if not do other magic

#3 - 03/27/2018 04:05 AM - phluid61 (Matthew Kerwin)

I'm confused; are you proposing that nil.to_s returns the same String object every time, or that string interpolation detects a nil object and optimises it?

A lot of times I rely on "#{foo}" returning a new String object that contains a copy of the #to_s of foo, so it seems to me like spec that "#{nil}" returns a new empty string every time.

#4 - 03/27/2018 04:17 AM - bumblingbear (Dillon Welch)

Closer to the string interpolation optimization part. The problem with "#{nil}" is that it allocates two empty strings each time. I feel like it would be possible to do this with zero/one allocation. I don't know what the underlying code looks like, but the logic I'm thinking of is something like

```
def interpolate(obj)
  if obj == nil
    ''
  else
    obj.to_s # assuming this is current behavior
  end
end
```

phluid61 (Matthew Kerwin) wrote:

> I'm confused; are you proposing that nil.to_s returns the same String object every time, or that string interpolation detects a nil object and optimises it?

> A lot of times I rely on "#{foo}" returning a new String object that contains a copy of the #to_s of foo, so it seems to me like spec that "#{nil}" returns a new empty string every time.

#5 - 03/27/2018 08:34 AM - nobu (Nobuyoshi Nakada)

- Backport deleted (2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN)

- Description updated

- Tracker changed from Bug to Feature

#6 - 03/31/2018 03:43 PM - nobu (Nobuyoshi Nakada)

https://github.com/nobu/ruby/tree/feature/opt_to_s

#7 - 04/21/2018 11:12 AM - normalperson (Eric Wong)

nobu@ruby-lang.org wrote:

> https://github.com/nobu/ruby/tree/feature/opt_to_s

Btw, I also had [ruby-core:81905] [Feature #13715] from last
year but forgot about it :x