Ruby - Feature #11848

New #to_b method for String, Symbol, Numeric, NilClass, TrueClass and FalseClass.

12/19/2015 05:06 PM - prodis (Fernando Hamasaki de Amorim)

Status:	Rejected
Priority:	Normal
Assignee:	matz (Yukihiro Matsumoto)
Target version:	

Description

New ***to_b*** method converts **strings**, **symbols**, **numbers** and **nil** values in a **boolean** value.

to_b method is available on ***String***, ***Symbol***, ***Numeric***, ***TrueClass***, ***FalseClass*** and ***NilClass*** classes.

String

Returns ***true*** if string is one of **t**, **true**, **on**, **y**, **yes** or **1** values. Returns ***false*** otherwise.
Ignores trailing spaces and letter cases.

```
't'.to_b         # => true
'true'.to_b      # => true
'on'.to_b        # => true
'y'.to_b         # => true
'yes'.to_b       # => true
'1'.to_b         # => true

''.to_b          # => false
'0'.to_b         # => false
'2'.to_b         # => false
'-1'.to_b        # => false
'f'.to_b         # => false
'false'.to_b     # => false
'off'.to_b       # => false
'n'.to_b         # => false
'no'.to_b        # => false
'wherever'.to_b # => false
```

Symbol

Same as ***symbol.to_s.to_b***.

```
:'1'.to_b       # => true
:t.to_b         # => true
:true.to_b      # => true
:on.to_b        # => true
:y.to_b         # => true
:yes.to_b       # => true

:f.to_b         # => false
:false.to_b     # => false
:off.to_b       # => false
:n.to_b         # => false
:no.to_b        # => false
:wherever.to_b # => false
```

Numeric

Returns ***false*** if number is **zero**. Returns ***true*** otherwise.

Integer

```
0.to_b  # => false
```

```
1.to_b  # => true
2.to_b  # => true
-1.to_b # => true
-2.to_b # => true
```

Float

```
0.0.to_b  # => false
0.1.to_b  # => true
1.0.to_b  # => true
-0.1.to_b # => true
-1.0.to_b # => true
```

BigDecimal

```
require 'bigdecimal'

BigDecimal('0.0').to_b  # => false
BigDecimal('0.1').to_b  # => true
BigDecimal('1.0').to_b  # => true
BigDecimal('-0.1').to_b # => true
BigDecimal('-1.0').to_b # => true
```

NilClass

Returns ***false***.

```
nil.to_b # => false
```

TrueClass

Returns ***true***.

```
true.to_b # => true
```

FalseClass

Returns ***false***.

```
false.to_b # => false
```

Related issues:

Has duplicate Ruby - Feature #12012: Add Boolean method	**Rejected**

History

#1 - 12/19/2015 05:06 PM - prodis (Fernando Hamasaki de Amorim)

- Tracker changed from Bug to Feature

#2 - 12/19/2015 07:49 PM - marcandre (Marc-Andre Lafortune)

- Status changed from Open to Feedback

- Assignee set to matz (Yukihiro Matsumoto)

So many decisions in this seem completely arbitrary (and inconsistent), plus you don't give a use case, there's no way this would ever be accepted.

I'd suggest you build your own hash of true/false values and use that.

#3 - 12/19/2015 10:10 PM - matz (Yukihiro Matsumoto)

- Status changed from Feedback to Rejected

You forgot empty string/array/hash to be false. ;-)
But Ruby is not Python.

Matz.

#4 - 12/19/2015 10:43 PM - Eregon (Benoit Daloze)

I like the idea of having explicit conversion between booleans/integers, such that 0.to_b => false and true.to_i => 1.
!int.zero? can be a workaround for the first conversion but I only see a ternary condition for the second case.
Also, nil.to_i is already 0.

#5 - 12/20/2015 02:18 PM - prodis (Fernando Hamasaki de Amorim)

The idea of ***to_b*** method is inspired in Rails and Swift:

- ActiveRecord::Type:Boolean in Rails:
  https://github.com/rails/rails/blob/fc4084fc5bdf60c46824094f4d6a362304c047f6/activerecord/lib/active_record/type/boolean.rb#L10
  https://github.com/rails/rails/blob/fc4084fc5bdf60c46824094f4d6a362304c047f6/activerecord/lib/active_record/connection_adapters/column.rb#L8

- NString#boolValue property in Swift:
  https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSString_Class/index.html#//apple_ref/occ/instp/NSString/boolValue

#6 - 01/22/2016 09:21 AM - naruse (Yui NARUSE)

- Has duplicate Feature #12012: Add Boolean method added

#7 - 01/30/2016 02:04 AM - avit (Andrew Vit)

I've had to do this in a few places over the years myself:

```
TRUTHY_VALUES = [true, 1, '1', 't', 'T', 'true', 'TRUE', 'y', 'Y', 'yes', 'YES']
FALSY_VALUES = [false, 0, '0', 'f', 'F', 'false', 'FALSE', 'n', 'N', 'no', 'NO']
```

> You forgot empty string/array/hash to be false. ;-)

I think the main reason for this is handling user input (from a file like CSV or other), so other types like Array/Hash are not expected there: just basic scalar values.

Still, it probably only makes sense for some specific situations.

#8 - 10/27/2016 11:14 PM - prodis (Fernando Hamasaki de Amorim)

Andrew Vit wrote:

> I've had to do this in a few places over the years myself:
>
> ```
> TRUTHY_VALUES = [true, 1, '1', 't', 'T', 'true', 'TRUE', 'y', 'Y', 'yes', 'YES']
> FALSY_VALUES = [false, 0, '0', 'f', 'F', 'false', 'FALSE', 'n', 'N', 'no', 'NO']
> ```
>
> > You forgot empty string/array/hash to be false. ;-)
>
> I think the main reason for this is handling user input (from a file like CSV or other), so other types like Array/Hash are not expected there: just basic scalar values.
>
> Still, it probably only makes sense for some specific situations.

Andrew Vit, you can use **wannabe_bool** gem: https://github.com/prodis/wannabe_bool

Files

to_b_method.diff	9.63 KB	12/19/2015	prodis (Fernando Hamasaki de Amorim)