Ruby master - Feature #12968

Allow default value via block for Integer(), Float() and Rational()

11/22/2016 09:30 AM - sos4nt (Stefan Schüßler)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

Kernel provides the global conversion functions Integer(), Float() and Rational(). Each one convert its argument to the respective type.

Parsing errors are signaled by raising a TypeError or an ArgumentError.

I often use these methods to parse numeric string values, but I find the need for a rescue ArgumentError block quite annoying an distracting.

I therefore request a new feature: Integer(), Float() and Rational() should take an optional block that serves as a callback when parsing failed.

Examples:

```
Integer('foo')                 #=> ArgumentError  (just as before)
Integer('foo') { 0 }           #=> 0              (same as 'foo'.to_i)
Integer('foo') { nil }         #=> nil
Integer('foo') { Float::NAN } #=> NaN
```

This resembles the way Hash#fetch or Hash#fetch_values work when a block is given.

Although not used in the examples above, any arguments should be passed along to the block, i.e. Integer('foo') { |arg, base| ... }

Related issues:

Related to CommonRuby - Feature #12732: An option to pass to `Integer`, `Floa...	**Closed**

History

#1 - 11/22/2016 10:07 AM - mudasobwa (Alexei Matyushkin)

Stefan Schüßler wrote:

```
Integer('foo') { 0 }           #=> 0              (same as 'foo'.to_i)
```

To make it possible to fallback to e.g. #to_i it would be great to receive an actual argument as block parameter:

```
#                  ⇓⇓⇓⇓⇓
Integer('foo') { |val| val.to_i }
```

#2 - 11/22/2016 10:08 AM - shyouhei (Shyouhei Urabe)

- Related to Feature #12732: An option to pass to `Integer`, `Float`, to return `nil` instead of raise an exception added

#3 - 11/22/2016 10:17 AM - sos4nt (Stefan Schüßler)

Alexei Matyushkin wrote:

> To make it possible to fallback to e.g. #to_i it would be great to receive an actual argument as block parameter:

Absolutely, I've mentioned that in the last sentence.

#4 - 11/22/2016 11:31 AM - shyouhei (Shyouhei Urabe)

+1 I like this idea. It must also be backwards compatible.

#5 - 11/23/2016 03:17 AM - nobu (Nobuyoshi Nakada)

Only "invalid value for Integer()" from a string?
Integer can raise exceptions other than ArgumentError against a string, for example.

```
Integer(nil)            #=> can't convert nil into Integer (TypeError)
Integer(Object.new)     #=> can't convert Object into Integer (TypeError)
Integer(Object.new, 2) #=> base specified for non string value (ArgumentError)
Integer("", 1)          #=> invalid radix 1 (ArgumentError)
X=Struct.new(:to_i)
Integer(X.new("a"))     #=> can't convert #X to Integer (X#to_i gives String) (TypeError)
```

#6 - 11/24/2016 02:09 PM - sos4nt (Stefan Schüßler)

Nobuyoshi Nakada wrote:

> Only "invalid value for Integer()" from a string?

I don't have a strong opinion on this.

#7 - 02/23/2017 08:43 AM - shyouhei (Shyouhei Urabe)

We looked at this issue at yesterday's developer meeting.

People there argued that explicit keyword argument is much easier to read than passing a block. They say it is not obvious what the block is going to do. So if the OP is OK we would like to to merge this issue with #12732. How do you feel, Stefan?

#8 - 06/28/2017 05:53 PM - sos4nt (Stefan Schüßler)

shyouhei (Shyouhei Urabe) wrote:

> we would like to to merge this issue with #12732. How do you feel, Stefan?

I'm fine with with that.