Ruby - Feature #7091

Object#puts_to

09/30/2012 11:40 PM - prijutme4ty (Ilya Vorontsov)

Status:	Rejected	
Priority:	Normal	
Assignee:		
Target version:		

Description

I suggest a new method Object#puts_to(io_or_filename) (or BasicObject#puts_to)

It's usual that one-two-three-line scripts have big chains like
readlines.sort.map{...}.select{...} and so on and after you wrote such a monstruous expression to process input, you understand that you should output it. If script's written offhand, you wouldn't create a new variable just to use it at next line, so you write smth like

```
puts( readlines.sort.map{...}.select{...} )
```
or at least
```
readlines.sort.map{...}.select{...}.tap{|x| puts x}
```

It looks ugly and isn't readable. Thing get even worse when you are writing object info a file:
```
File.open('file.txt','w'){|f| f.puts( readlines.sort.map{...}.select{...} ) }
```

I write such constructions many times a day, just because my scripts are usually used once or twice and I can't waste my time to make this more clear.
Instead of such a pasta-code, one can make smth like this:

```
readlines.sort.map{...}.select{...}.puts_to
readlines.sort.map{...}.select{...}.puts_to($stderr)
readlines.sort.map{...}.select{...}.puts_to('')
readlines.sort.map{...}.select{...}.puts_to(filename:'', append:true)
```

Implementation can be smth like this:

```
class Object
def puts_to(io_or_filename = $stdout)
if io_or_filename.respond_to?(:puts)
io_or_filename.puts(self)
else
case io_or_filename
when String
File.open(io_or_filename,'w'){|f| f.puts self }
when Hash
File.open(io_or_filename[:filename],io_or_filename[:append] ? 'a' : 'w'){|f| f.puts self }
end
end
end
end
```

Or may be Hash-syntax for append-mode should be written simply as two arguments:
```
obj.puts_to('file.txt', true)
```

History

#1 - 10/01/2012 12:54 AM - drbrain (Eric Hodel)

- Status changed from Open to Rejected

=begin
There are several ways to do this just as easily:

Write to $stdout:

```
puts(*readlines.sort.map{...}.select{...})
```

Write to $stderr:

$stderr.puts(*readlines.sort.map{...}.select{...})

Write to 'file.txt':

File.write 'file.txt', readlines.sort.map{...}.select{...}.join

Append to 'file.txt':

File.write 'file.txt', readlines.sort.map{...}.select{...}.join, 'a'
=end

#2 - 10/01/2012 10:30 AM - nobu (Nobuyoshi Nakada)

=begin
readlines.sort.map{...}.select{...}.display($stderr)
=end

#3 - 10/01/2012 06:31 PM - prijutme4ty (Ilya Vorontsov)

Don't you feel that mixing function call and method-chain is ugly? This was the main reason why I proposed this method, not inability to do this.
In case that expression is full of curl-braces and so on, additional level of "brace-deepness" is much worse that additional method in chain. Compare a real-life example (discreting a matrix):

File.write('matrix.out', File.readlines('matrix.in').map{|line| line.strip.split.map(&:to_f).map(&:round).join("\t")})

File.readlines('matrix.in').map{|line| line.strip.split.map(&:to_f).map(&:round).join("\t")}.puts_to('matrix.out')

Both expressions are rather complex, but in the first form main operations are hidden inside braces of write or puts function. Also here input and output appear in adjacent expressions. In the second form main part of calculations is not shaded by output code. Also one can trace path of data from input file through calculations to output file.

#4 - 10/01/2012 06:43 PM - prijutme4ty (Ilya Vorontsov)

nobu (Nobuyoshi Nakada) wrote:

> =begin
> readlines.sort.map{...}.select{...}.display($stderr)
> =end

Thank you, Nobu. Hadn't seen this before. I wish to have similar syntax for output in file by filename, but this is also good enough.