

Ruby master - Feature #17165

Add `filter` and `flatten` keywords to `Enumerable#map`

09/12/2020 02:46 PM - sawa (Tsuyoshi Sawada)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	
Description	
<p>I had a use case to do map on an enumerable, with 1-level flattening, while skipping nil values.</p> <p>There are convenient <code>Enumerable#flat_map</code> and <code>Enumerable#filter_map</code> methods, but the problem is that they cannot be used at the same time. I had to chose to do either of the following:</p>	
<pre>array .filter_map do foo bar = baz(foo) next unless bar bar.map{...} end .flatten(1)</pre>	
<pre>array .flat_map do foo bar = baz(foo) next unless bar bar.map{...} end .compact</pre>	
<pre>array .flat_map do foo bar = baz(foo) next [] unless bar bar.map{...} end</pre>	
<p>The last one of the above may not look so bad, but it requires an extra consideration, and is a bit hacky. When you are in a hurry, it just might not come to your mind.</p> <p>This led me to realize that <code>flat_map</code> and <code>filter_map</code> should not be independent operations, but are rather some different modes of the operation <code>map</code>. There is no reason for the modes to be mutually exclusive of one another, and a use case that I mentioned above may arise.</p> <p>I propose to add <code>filter</code> and <code>flatten</code> as optional keyword arguments to <code>Enumerable#map</code>.</p>	
<pre>array .map(filter: true, flatten: 1) do foo bar = baz(foo) next unless bar bar.map{...} end</pre>	
<p>In fact, even when the two parameters are not used together, I believe it would be easier to the brain and I would feel much more comfortable to pass <code>filter: true</code> or <code>flatten: 1</code> to <code>map</code> when necessary rather than having to decide whether to use <code>map</code> or <code>flat_map</code> or use <code>map</code> or <code>filter_map</code>.</p> <p>Furthermore, this would make it possible to do flattening of an arbitrary depth (as specified by the parameter) during <code>map</code>.</p>	

History

#1 - 09/13/2020 09:38 AM - Eregon (Benoit Daloze)

What's the problem with the obvious:

```
array.map { |foo|
  baz(foo)
}.select { |bar|
  condition(bar)
}.flat_map { |bar|
  bar.map{...}
}
```

I think it's so much more readable.

And I don't think the extra allocations matter much.

IMHO Enumerable#map should map elements, and nothing else.

That's also seem to have been the opinion of many others.

#2 - 09/13/2020 09:41 AM - Eregon (Benoit Daloze)

And I'd argue if one wants to do everything in one block, just enjoy the freedom of imperative programming:

```
result = []
array.each do |foo|
  if bar = baz(foo)
    result.concat bar.map{...}
  end
end
```