Ruby master - Feature #5044
#zip with block return mapped results
07/18/2011 12:35 PM - trans (Thomas Sawyer)

<table>
<thead>
<tr>
<th>Status</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee</td>
<td>matz (Yukihiro Matsumoto)</td>
</tr>
<tr>
<td>Target version</td>
<td></td>
</tr>
</tbody>
</table>

Description
Is there any reason that #zip doesn't return results like map?

\[ 1,2,3 \].zip([1,2,3]) \{ |a,b| a + b \} #=> [2,4,6]

Currently, it just returns nil, which seems rather contrary to the return result of the non-block form.

Related issues:
Related to Ruby master - Feature #4539: Array#zip_with
Assigned
Related to Ruby master - Feature #16261: Enumerable#each_splat and Enumerator...
Rejected

History
#1 - 07/18/2011 10:29 PM - Eregon (Benoit Daloze)

Hi,

Please have a look at [ruby-lang.org/issues/4539](http://redmine.ruby-lang.org/issues/4539) (or [ruby-core:35613]).

The reason is explained by Yusuke in [ruby-core:35682]:

2011/4/5 Benoit Daloze eregontp@gmail.com:

An unconditional nil is anyway not useful here, the only concern I see might be the cost of creating this Array.

It is actually a problem.
I have used Array#zip with block for iteration many times.

```
big_ary1.zip(big_ary2) do |x, y|
  p [x, y]
end
```

The change requires some people (including me) to rewrite such a code as follows:

```
big_ary1.size.times do |i|
x, y = big_ary1[i], big_ary2[i]
  ...
end
```

So I like zip_with, if it is really needed.

That issue kind of got lost, it is probably a good idea to continue it.

For your example, we could have [1,2,3].zip_with([1,2,3], :+)

#2 - 03/25/2012 05:02 PM - akr (Akira Tanaka)

Currently it can be implemented as follows.

```
% ruby -e 'p [1,2,3].zip([1,2,3]).map {[a,b] a + b }'
[2, 4, 6]
% ruby -e 'p [1,2,3].lazy.zip([1,2,3]).map {[a,b] a + b }.to_a'
[2, 4, 6]
```

#3 - 03/25/2012 05:03 PM - mame (Yusuke Endoh)

- Status changed from Open to Assigned
We cannot change the behavior. The change would increase the memory consumption (and decrease the performance).

Matz.

- Related to Feature #16261: Enumerable#each_splat and Enumerator#splat added