Ruby trunk - Feature #8490

Bring ActiveSupport Enumerable#index_by to core

06/05/2013 01:30 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

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

Description

It seems to be a common sense to have the useful implementation of Enumerable#index_by as in ActiveSupport.

index_by acts like group_by but only maps a single record, instead of an array, keeping the last match only. It's usually used in places where you shouldn't have more than a single match for each key.

Would you consider its inclusion to core Enumerable module?

History

#1 - 06/05/2013 09:15 AM - matz (Yukihiro Matsumoto)

- Status changed from Open to Rejected

I don't think "index" is a proper term here. Array#index, Enumerable#find_index, Enumerable#with_index all treat "index" as an offset to an item in the enumerables. Whereas this method (index_by) is not related to offsets.

Matz.

#2 - 06/05/2013 11:53 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

if you read index as a name, then you're correct, but in index_by it is a verb, meaning that you're indexing an enumerable by something, by transforming it on a hash indexed by that something... Please reconsider.

#3 - 06/06/2013 12:16 AM - Eregon (Benoit Daloze)

rosenfeld (Rodrigo Rosenfeld Rosas) wrote:

> if you read index as a name, then you're correct, but in index_by it is a verb, meaning that you're indexing an enumerable by something, by transforming it on a hash indexed by that something... Please reconsider.

Yes, of course, but it still remains confusing and not consistent at first sight.
Also, semantics are not very clear, why last instead of first if multiple by key?

I do agree Hash construction from Enumerable is something to work on,
but this name is definitely not the best, neither are the semantics.

#4 - 06/06/2013 12:18 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

I don't see any issues with index_by, but how about map_by? And what is wrong with the semantics?

#5 - 06/06/2013 12:27 AM - Eregon (Benoit Daloze)

rosenfeld (Rodrigo Rosenfeld Rosas) wrote:

> And what is wrong with the semantics?

The name does not indicate clearly what it does (which is chosen if multiple yield the same key) and it is a possibly lossy operation. Can you give a couple examples where it is much better than #group_by?

#6 - 06/06/2013 02:25 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

Eregon (Benoit Daloze) wrote:

> The name does not indicate clearly what it does (which is chosen if multiple yield the same key) and it is a possibly lossy operation. Can you give a couple examples where it is much better than #group_by?

Sure, suppose you do a database search to fetch some records and you want them mapped by their "id" column. As I said previously, index_by is usually useful when you know beforehand that there should be a single match for the indexed key, like a search in a database table where id is a primary key (uniquely identified). In most ORM solutions that would mean something like:

record_by_id = records.index_by &:id

Since we're not worried about what should happen in case there are multiple matches, we prefer to just use any rule for dealing with those (like using latest match or whatever performs better or is easier to implement). You could also have some index_by! (or map_by!) alternative for raising an exception in case there would be multiple matches if that would be useful for someone to handle that case.

With group_by we'd need to access it like this:

```
records_by_id = records.group_by &:id
records = records_by_id[id]
record = records && records.first
```

Just compare to:

```
record_by_id = records.index_by &:id
record = record_by_id[id]
```

#7 - 06/07/2013 04:40 PM - Eregon (Benoit Daloze)

rosenfeld (Rodrigo Rosenfeld Rosas) wrote:

> Eregon (Benoit Daloze) wrote:
>
> > The name does not indicate clearly what it does (which is chosen if multiple yield the same key) and it is a possibly lossy operation. Can you give a couple examples where it is much better than #group_by?
>
> Sure, suppose you do a database search to fetch some records and you want them mapped by their "id" column. As I said previously, index_by is usually useful when you know beforehand that there should be a single match for the indexed key, like a search in a database table where id is a primary key (uniquely identified). In most ORM solutions that would mean something like:
>
> record_by_id = records.index_by &:id

I think your ORM should do it (or you should use a different technique in the code).
In this case it is highly preferable to raise an exception if there are 2 identical keys.

> Since we're not worried about what should happen in case there are multiple matches, we prefer to just use any rule for dealing with those (like using latest match or whatever performs better or is easier to implement). You could also have some index_by! (or map_by!) alternative for raising an exception in case there would be multiple matches if that would be useful for someone to handle that case.

I think the method without exception is more dangerous than useful.

#8 - 06/07/2013 09:50 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

I'd prefer index_by! to raise an exception instead... What if you know that any duplicate would actually map to the same object?

Also, you can't judge how my ORM should work (anyway, all the ORM I've worked with, both in Ruby, Groovy and Java have a very similar API when it comes to fetching a list of records, there's nothing special on this regular requirement)... Sometimes I'm working in something that would take advantage of some caching of records by id that I'll use somewhere, but that doesn't mean I always need to index by id... This was just a use case, it doesn't mean that every time I need to index a list of objects it comes from the database... So I can't understand why you're so sure that "I should use a different technique in the code" as you don't even know my code...

#9 - 06/07/2013 10:29 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

I can't show you my code, but here are some common usage you can find on Github open source code:

https://github.com/dim/plexigrid/blob/6358f11af7806849de756c226791c53f39521efc/rails_plugin/plexigrid/lib/plexi_grid/column_model.rb#L21
https://github.com/astrails/benyehuda-cases-server/blob/e3b797d2edf005f4a338179bd2b1ed8aa2935597/lib/custom_properties.rb#L16
https://github.com/zerglings/upside_web/blob/b2326f23673f300b47572665de8a7f7a67b4c992/app/models/stock_cache_line.rb#L58

Notice in this last example that the cache_fetch method would also benefit from index_by. It is the same as lines_for but it takes an expiration time argument.

#10 - 06/08/2013 01:53 AM - Eregon (Benoit Daloze)

rosenfeld (Rodrigo Rosenfeld Rosas) wrote:

> I'd prefer index_by! to raise an exception instead...

! is not used for exception-raising methods in ruby core and stdlib.

> What if you know that any duplicate would actually map to the same object?

There should be some possibility to merge duplicates like Hash#merge in that case.
But is it worth the complexity? If you need some merging strategy,
the operation is no more mere "indexing", and I would favor the current approach ("h={}; enum.each { |e| h[e.k] = e if ... }").

> Also, you can't judge how my ORM should work (anyway, all the ORM I've worked with, both in Ruby, Groovy and Java have a very similar API when it comes to fetching a list of records, there's nothing special on this regular requirement)...

Well, the ORM (or whatever you use between DB and code) could do this.
And having something like "index = {}; each_record { |record| index[record.key] = record }; index"
could be useful I guess. For instance, Sequel has some Dataset#select_hash which is somewhat similar.

> Sometimes I'm working in something that would take advantage of some caching of records by id that I'll use somewhere, but that doesn't mean I always need to index by id... This was just a use case, it doesn't mean that every time I need to index a list of objects it comes from the database...

Yes, of course. I agree it would be useful to have something like index_by.
But maybe, as in the #cache_fetch example, it would be useful to be able to change/map the values too.
(if the name has "map" in it, it should map values as well.)

> So I can't understand why you're so sure that "I should use a different technique in the code" as you don't even know my code...

I am not "so sure", I started with "I think".
Of course there are special cases and I can not know your code.
But to me using keys like that for databases record does not seem the most appropriate way to do it at first sight (probably best iterating on the properly sorted result set).

#11 - 06/09/2013 12:37 PM - trans (Thomas Sawyer)

key_by