Ruby master - Feature #15975

Add Array#pluck

07/02/2019 02:05 PM - lewispb (Lewis Buckley)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

Inspired by https://github.com/rails/rails/issues/20339

While developing web applications I've often wanted to quickly extract an array of values from an array of hashes.

With an array of objects, this is possible:

```
irb(main):001:0> require 'ostruct'
=> true
irb(main):002:0> [OpenStruct.new(name: "Lewis")].map(&:name)
=> ["Lewis"]
```

This PR adds Array#pluck allowing this:

```
irb(main):001:0> [ {name: "Lewis"} ].pluck(:name)
=> ["Lewis"]
```

without this PR:

```
irb(main):001:0> [ {name: "Lewis"} ].map { |item| item[:name] }
=> ["Lewis"]
```

Implemented here:

https://github.com/ruby/ruby/pull/2263

History

#1 - 07/02/2019 04:14 PM - shevegen (Robert A. Heiler)

Hmm. I don't doubt that this may possibly be useful, but the method name is
a bit ... weird. My first association with this name, oddly enough, is to
associate duck typing with it, and then to "pluck the duck" (yes, strange
association but I could not help it ...).

I do not have a better alternative suggestion for a name, though. It
reminds me a little bit of a more flexible variant of .dig(), though.

#2 - 07/16/2019 07:13 AM - inopinatus (Joshua GOODALL)

I think that's pretty limited. #pluck is a fairly crude method, fine for Rails but hardly befitting the Ruby standard library. I'd much rather use a higher-order function and get somewhere much more interesting.

By instead implementing Array#to_proc (which doesn't currently exist) as something that applies to_proc to its own elements, before invoking them with passed-in arguments:

```
class Array
  def to_proc
    Proc.new do |head, *tail|
      collect(&:to_proc).collect do |ep|
        ep_head = ep[head]
        tail.empty? ? ep_head : [ep_head] + tail.collect(&ep)
      end
    end
  end
end
```

we can now do some nice things, including a pluck equivalent (and more besides) but using only existing syntax:

```
# data
people = [{name: "Park", age: 42}, {name: "Lee", age: 31}]
keys = people.flat_map(&:keys).uniq

# single item extraction
:name.then &people      #=> ["Park", "Lee"] and equivalent to
people.to_proc[:name]   #=> ["Park", "Lee"]

# multiple item extraction
keys.then &people            #=> [["Park", 42], ["Lee", 31]] and equivalent to
people.to_proc[:name, :age]  #=> [["Park", 42], ["Lee", 31]]

# multiple method invocation
:name.then(&people).map(&[:upcase, :length]) #=> [["PARK", 4], ["LEE", 3]]

# use with struct-like objects, and bonus inline lambda:
people.map(&OpenStruct.:new).map &[:name, :age, ->{ Digest::SHA2.hexdigest @1.name }]
```

Could work as Enumerable#to_proc instead.

#3 - 07/16/2019 07:47 AM - osyo (manga osyo)

> we can now do some very nice things just with existing syntax:

The sample code is invalid.
Is this?

```
class Array
  def to_proc
    Proc.new do |head, *tail|
      collect(&:to_proc).collect do |ep|
        ep_head = ep[head]
        tail.empty? ? ep_head : [ep_head] + tail.collect(&ep)
      end
    end
  end
end

# data
people = [{name: "Park", age: 42}, {name: "Lee", age: 31}]

# single item extraction
p :name.then &people      #=> ["Park", "Lee"]
p people.to_proc[:name]   #=> ["Park", "Lee"]

# multiple item extraction
p [:name, :age].then &people            #=> [["Park", 42], ["Lee", 31]]
p people.to_proc[:name, :age]  #=> [["Park", 42], ["Lee", 31]]

# multiple invocation
names = ["Park", "Lee"]
p names.map(&[:upcase, :length]) #=> [["PARK", 4], ["LEE", 3]]
```

https://wandbox.org/permlink/4oVOzULhwKsu4gB5

#4 - 07/17/2019 02:54 AM - inopinatus (Joshua GOODALL)

My apologies, yes, there was a cut-and-paste error on show for a few minutes, and you were quick enough to see it. It's now the code I intended to paste.

#5 - 09/02/2019 06:19 AM - knu (Akinori MUSHA)

ActiveSupport has Enumerable#pluck, so I don't think we want to diverge from that by adding a method with the same name in Array.

#6 - 09/02/2019 06:28 AM - matz (Yukihiro Matsumoto)

I am not positive for Array#pluck. ActiveSupport may add the method.

Matz.