agilie

Тестовое задание для позиции Ruby on Rails developer

Создайте класс обработчика, который примет массив элементов и выполнит действие для каждого элемента, но только если элемент еще не обработан. Идея заключается в том, что этот класс будет полезен для обработки нескольких партий элементов, гарантируя, что каждый элемент обрабатывается один раз, даже если элемент включен в несколько партий.
Обработчик будет поддерживать следующие методы:

proc_items - вызывается с массивом в качестве аргумента и требует блока. Передаваемый блок используется для обработки элемента, который будет вызываться только в том случае, если определяет, что элемент должен быть обработан.

procd_items - возвращает массив обработанных элементов.

idefine - Цель этого метода - определить каким образом разные экземпляры объектов будут считаться одинаковыми. Если он вызван, он определит, какой хэш-ключ использовать в качестве критерия сравнения (если переданы хеши) или какой метод использовать (если предоставляется объект). Он принимает символ или строковый аргумент, который определяет имя хеш-ключа / метода, которое будет использоваться для идентификации элемента. Если этот метод не вызывается, экземпляры будут сравниваться друг с другом обычным образом.

should_proc - Цель этого метода - настроить передачу необязательного блока, который будет использоваться для определения того, должен ли элемент обрабатываться. Этот необязательный блок возвращает true, если элемент должен быть обработан, иначе он
вернет false. Когда этот метод используется, блок, переданный ему, будет использоваться совместно с логикой «этот элемент уже обработан?»

reset - Цель этого метода - сбросить состояние обработанных элементов, позволяя повторно обрабатывать элементы.

Пример того, как это должно работать:

agilie

```ruby
pack_handler = PackHandler.new
pack_handler.proc_items([1,2,3,4]) do |item|
 # будет обрабатывать 1, 2, 3 и 4
end

pack_handler.proc_items([3,4,5,6]) do |item|
 # будет обрабатывать 5 и 6
end

pack_handler.reset #сбросить состояние обработанных элементов


pack_handler.proc_items([{'id' => 1}, {'id' => 1, 'test_key' => 'Some data'}]) do
|item|
# будет обрабатывать оба элемента
end

pack_handler.reset
pack_handler.identify('id')
pack_handler.proc_items([{'id' => 1}, {'id' => 1, 'test_key' => 'Some data'}, {'id' =>
2}]) do |item|
 # будет обрабатывать первый и последний элемент.
 # второй элемент будет пропущен после вызова идентификатора, а
его идентификатор совпадает с первым элементом.
end

pack_handler.proc_items([{'id' => 2}, {'id' => 3}]) do |item|
 # будет обрабатывать только {'id' => 3}.
end

pack_handler.reset

# теперь мы будем использовать процессор только для обработки элементов с
четными значениями. Мы не хотим обрабатывать одно и то же значение
дважды pack_handler.identify(:value)
pack_handler.should_proc do |item|
 item[:value] % 2 == 0
end

pack_handler.proc_items([{value: 2}, {value: 3}]) do |item|
 # будет обрабатывать только {value: 2},
end

pack_handler.proc_items([{value: 2}, {value: 6}]) do |item|
 # будет обрабатывать только {value: 6} так как {value: 2} был
обработан end

# Пример поведения обработчика в ситуации, когда когда обрабатываются
```

```ruby
не не-хэш-элементы.
class SomeClass
 attr_reader :main_field
 def initialize(main_field)
   @main_field = main_field
 end
end

a = SomeClass.new('a')
b = SomeClass.new('b')

pack_handler = PackHandler.new
pack_handler.identify(:main_field)
pack_handler.proc_items([a, b]) do |item|
 # будет обрабатывать оба элемента
end
pack_handler.proc_items([SomeClass.new('a')]) do |item|
 # ничего не обработает, так как объект с атрибутом main_field,
имеющим значение 'a' уже обработан
end
```