Ruby master - Feature #10270

Hash#insert

09/20/2014 01:25 PM - atlas (Atlas Prime)

Status:	Feedback	
Priority:	Normal	
Assignee:		
Target version:		

Description

Hash doesn't appear to have a "safe" way to insert an entry that won't clobber another key if it is already there. So how about:

```
class Hash

  # Like Hash#store but only stores if the key isn't already
  # in the hash. Returns true if inserted, otherwise false.
  #
  def insert(name, value)
    if key?(name)
      false
    else
      store(name,value)
      true
    end
  end

end
```

History

#1 - 09/21/2014 11:50 AM - matz (Yukihiro Matsumoto)

- Status changed from Open to Feedback

1. I am not sure #insert is the best name for non clobbering merge.

2. do you know merge takes a block to resolve key conflict?

   h.merge(h2) {|key,v1,v2|v1}

works as your proposed #insert.

Matz.

#2 - 09/21/2014 01:01 PM - atlas (Atlas Prime)

The name "insert" seems good to me because it implies a *new* entry and unable to change an *old* entry. But key-value pair must be thought of as an *entry* too for it to make sense (I guess I get that notion from Java http://docs.oracle.com/javase/7/docs/api/java/util/Map.Entry.html).

I did not know that merge took this block. That could be useful. But it's not quite the same because insert returns true/false if successful of not. It's utility is mostly as a short cut for

```
if !h.key?(k)
  store(k,v)
  ...
end
```

Instead one can do

```
if h.insert(k,v)
  ...
end
```

However, you make me think it might be useful too if it could take a hash and a block.

```
def insert_merge(other, &block)
  other.each do |k, v|
    if key?(k)
      block.call(false, k, v)
    else
      store(k,v)
      block.call(true, k, v)
    end
  end
end
```