Class and Instance Methods in Ruby // RailsTips by John Nunemaker

Class and Instance Methods in Ruby

The other day I was explaining the difference between class and instance methods to a friend and I realized that I should probably write up a post. I figured since I'm on a plane headed back home, now was as good of time as ever. If you want a little history, you can read about the difference between class and instance variables.

One Line Summary

Class methods are methods that are called on a class and instance methods are methods that are called on an instance of a class. Here is a quick example and then we'll go into a bit more detail.

```
class Foo
  def self.bar
    puts 'class method'
  end

  def baz
    puts 'instance method'
  end
end

Foo.bar # => "class method"
Foo.baz # => NoMethodError: undefined method ‘baz’ for Foo:Class

Foo.new.baz # => instance method
Foo.new.bar # => NoMethodError: undefined method ‘bar’ for #<Foo:0x1e820>
```

See the difference? bar is a **class method**, so calling bar on the Foo class works fine. baz is an **instance method**, so calling baz on the Foo class raises a NoMethodError. Then, on the next couple lines, we call both methods on an **instance** of Foo (Foo.new).

Now that we have a base, let's show a few of the ways you can create class and instance methods and examples of what they would be used for.

Class Methods

Ruby is very flexible and as such it allows several ways to define a class method. The following is a sample of the most commonly used ways.

```
# Way 1
class Foo
  def self.bar
    puts 'class method'
  end
end

Foo.bar # "class method"

# Way 2
class Foo
  class << self
    def bar
      puts 'class method'
    end
  end
end

Foo.bar # "class method"

# Way 3
class Foo; end
def Foo.bar
  puts 'class method'
end

Foo.bar # "class method"
```

The first way is my preference. When I see self.method_name it is immediately apparent to me that this is a class method. A lot of people use way #2 and it is pretty heavily used in Rails.

There is nothing wrong with it, but when you have a class with a lot of class methods in a `class << self` block, it can be hard to tell if the method is a class or instance method because it is defined the same (`def bar`). If this doesn't make sense, feel free to use it for a while and you'll probably run into what I'm talking about.

Way 3 is not that common as far as I have seen and is more often a way to quickly add methods on the fly to a class. These are not the only three ways to define class methods, but they seem to be the ones that I see the most.

So when would you use a class method? Class methods are for anything that does not deal with an individual instance of a class. ActiveRecord::Base#find is one example. If you look in ActiveRecord::Base, you'll see something like this:

```
module ActiveRecord
```

```
  class Base
    # some stuff
    class << self
      def find(...)
        # blah
      end
    end
  end
end
```

Looks familiar, eh? Some other uses of class methods in Rails are validations and associations in ActiveRecord and before/after/around filters in ActionPack. The way this works is something like this (simplified for clarity):

```
module ActiveRecord
  class Base
    def self.validates_presence_of(...)
      # make sure present
    end
  end
end

class Foo < ActiveRecord::Base
  validates_presence_of :bar
end
```

When you say validates_presence_of, the class method in AR::Base is what gets called.

Instance Methods

Enough about class methods, lets move on. Instance methods are a bit more simple. Here are a few common ways that instance methods are defined.

```
# Way 1
class Foo
  def baz
    puts 'instance method'
  end
end

Foo.new.baz # "instance method"

# Way 2
class Foo
```

```
  attr_accessor :baz
end

foo = Foo.new
foo.baz = 'instance method'
puts foo.baz

# Way 3
class Foo; end

foo = Foo.new
def foo.bar
  puts 'instance method'
end

Foo.new.baz # "instance method"
```

The key difference is instance methods only work with an instance and thus you have to create a new instance to use them (Foo.new). Again, there are more ways to define instance methods than this, especially if you look into meta programming.

So what are some examples uses of instance methods in Rails, to give you a better idea? Ever do a find in a destroy action and then call destroy on the found instance? destroy is an instance method.

```
class FoosController < ActionController
  def destroy
    foo = Foo.find(params[:id])
    foo.destroy
    redirect_to foos_url
  end
end
```

So are save and update_attributes, which you have definitely used before if you've done any Rails.

```
foo = Foo.new(:title => 'Bar')
foo.save # is an instance method
```

Conclusion

Class methods can only be called on classes and instance methods can only be called on an instance of a class. It's simple when you understand it, but I remember being confused when I was learning Ruby. Hope this helps. If I was unclear or incorrect at any point above, let me know.

The next article that I also wrote on the plane ride home from Railsconf is Include verse Extend, which builds on this article.