Meta Programming

1 Method Missing

Whenever a method is called on a Ruby object that doesn't exist, method_missing is called on that object. We can override method_missing and apply our own logic to try and make sense of the missing method call.

Exercise 11

We'll create an object and override method_missing to allow that object to respond to some expressive method calls

Create a script called method_missing.rb

```
require 'date'
class EasyReader

    MATCH_DAYS = /^(print|write)_(\d+)_days_(ago|from_now)$/

    def method_missing(method, *args)
        method.to_s =~ MATCH_DAYS

        date = Date.today - $2.to_i if $3 == 'ago'
        date = Date.today + $2.to_i if $3 == 'from_now'

        message = "date is #{date}\n"

        File.open("easy_to_read.txt", 'a+') {|f| f.write(message) } if $1 == 'write'
        puts message if $1 == 'print'
        return if $&
        super
    end
end
```

1. Note ! the regular expression syntax MATCH_DAYS, load the script into IRB and try some methods that will trigger the method_missing logic.

2. Note the splat *argument 'args'. Splat turns a comma seperated list of arguments into an arry. Try this in IRB:
*args = 1,2,3,4
This allows method missing to accept multiple arguments in a generic way as a an array

WHERE ITS USED IN RAILS:

ActiveRecord

2 Alias & Alias Method

Alias simply provides an alternative name for a method so aliasing a method .count to .size will give the class two methods, .count and .size that both do the same thing.

Alias Method is more interesting. It let's us rename a method and then replace that with one of our own. Often used as 'hooks'.

Exercise 12

We're going to hook into the standard Ruby array and print out whenever someone adds or deletes from the array.

Create a script called 'array_watcher.rb'

```
class Array

    alias_method :oldpush, :<<
    alias_method :olddelete, :delete


    def <<(value)
      puts "someone's adding #{value}"
      oldpush value
    end

    def delete(value)
      puts "someone's trying to delete #{value}"
      obj = olddelete(value)
      puts "it's gone" if obj
      puts "delete failed, seems like we didn't have #{value}" unless obj
      obj
    end

end
```

load this script into IRB and try:

```
x = []
x << 77
x.delete(77)
x.delete(66)
```

WHERE ITS USED IN RAILS:

Delayed Job

3 Monkey Patching

Sometimes Ruby, Rails or a gem won't quite do what we want (sometimes because it's just a plain bug, sometimes because we need slightly different behaviour within the domain of our application). Like most OO languages It's possible to extend or encapsulate functionality into a new Class that adapts behavior but for a small change that's going to be used frequently then a monkey patch can be a pragmatic approach to clearer code. Use sparingly.

Exercise 13

We're going to monkey patch the Array class so that we have a method that returns a random member of the method.

Create a script called ‘random_array.rb'

```
class Array
  def random
    self[rand(self.size)]
  end
end
```

now load the script into IRB, create an array and get some results:

```
x = [1,2,3,4,5,6,7,8,9]
x.random
```

Exercise 14

We can also mony patch individual instances on the fly. Open irb:

```
x = [1,2,3]
y = [1,2,3]

def x.middle
  return self[1]
end

x.middle
y.middle

x.respond_to?(:middle)
y.respond_to?(:middle)
```

Exercise 15

Try to monkey patch String with a canonicalize method so that non aplhanumerics are converted to underscores.

“abc-def%123”.canonicalize = “abc_def_123” - quite a useful function on the web

Monkey Patching is used to enhance a number of basic ruby classes in Rails e.g.

- Years, months, weeks, days, minutes, seconds

Date.today - 3.years
Date.today - 2.months
Date.today - 4.weeks
Date.today - 2 days
Time.now - 5.minutes
Time.now - 10.seconds

- .blank?

nil.blank?
“”.blank?
[].blank?,
{}.blank?
false.blank?

all respond true in Rails but Ruby only has:

“”.empty?
[].empty?
{}.empty?

- .present?

the opposite of .blank?

- try

obj.try(:some_method)

will return the method value or nil if the method doesn’t exist. It won;t throw a method missing error.

- to_query

{:name => ‘john’, :age => 20}.to_query = “name=john&age=20”

converts a hash to a query string

The full list is here:

http://edgeguides.rubyonrails.org/active_support_core_extensions.html

1.9 A Ruby Application that scrapes the web.

A simple webscraper that starts at a given point on the web and then navigates randomly across the internet.

The scraper will navigate to a page, get a list of links on that page, reject any from the same domain, choose a link at random from what's left then move to that page and repeat.
It will also append each link it finds to a text file called scraper.txt.

Exercise 14

Create a script called scraper.rb - we'll also use the middle monkey patcher

```
require 'nokogiri'
require 'open-uri'
require 'net/http'

class Array
  def random
    self[rand(self.size)]
  end
end

class Scraper

 attr_accessor :scrapes

 def scraped
   @scraped ||= []
 end

 def uri=(uri)
   @uri = uri
   scraped << doc unless @doc.nil?
   @doc = nil
 end

 def uri
   @uri
 end

 def initialize(uri, scrapes)
   self.uri = URI.parse(uri)
   self.scrapes = scrapes
 end

 def scrape
   until scraped.size == scrapes
     next_uri
   end
 end
```

```
 def doc
    @doc ||= Nokogiri::HTML(open(@uri))
 end

 def anchors
  doc.css('a')
 end

 def next_uri
  puts scraped.size
  others = anchors.reject do |anchor|
    anchor_uri = uri_from_anchor(anchor)
    anchor_uri.nil? || anchor_uri.host.nil? || anchor_uri.host == @uri.host
  end
  puts "found #{others.size} external domains"

  begin
   anchor = others.random
   random_uri = uri_from_anchor(anchor)
   write(anchor)
   puts "#{anchor.content} links to #{random_uri}"
  end while random_uri.nil?

  self.uri = random_uri
 end

 def write(anchor)
   File.open("Scraper.txt", 'a+') {|f| f.write("#{anchor.content} >>>
#{anchor.attribute("href")}\n") }
 end

 def uri_from_anchor(anchor)
    return URI.parse(anchor.attribute("href").to_s)
 rescue
    nil
 end

end
```

Load the script into IRB, choose a web page such as 'http://www.bbc.co.uk' and get the scraper going.