DATA SCRAPPING

INSTALLER REST CLIENT => GEM INSTALL REST-CLIENT

INSTALLER PRY => GEM INSTALL PRY

ECRIRE BINDING.PRY avant d’exécuter les fonction

http://ruby.bastardsbook.com/chapters/html-parsing/

Pour récupérer les données d’une page internet on doit installer la gem nokogiri :

Gem install nokogiri

Puis créer un nouveau fichier .rb et commencer ce fichier par ces deux lignes :

Require ‘rubygems’

Require ‘nokogiri’

Pour récupérer les données d’une page sur le disque dur on peut procéder de la sorte :

page = Nokogiri::HTML(open("index.html"))

puts page.class # => Nokogiri::HTML::Document

Pour récupérer les données d’une page sur internet on utilise cette méthode :

require 'rubygems'

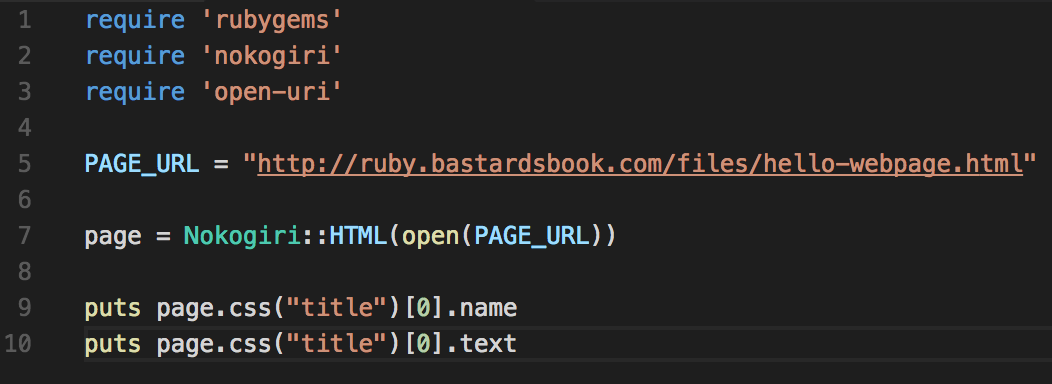
require 'nokogiri'

require 'open-uri'

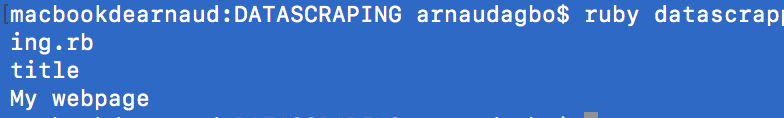
page = Nokogiri::HTML(open("http://en.wikipedia.org/"))

puts page.class # => Nokogiri::HTML::Document

Pour récupérer le texte associé à un élément il faut procéder de la sorte :



Ce qui nous donnera :



On peut récupérer les url des sites vers lesquels pointent les liens dans le site visé via cette méthode :

# set URL to point to where the page exists

page = Nokogiri::HTML(open(PAGE\_URL))

links = page.css("a")

puts links.length # => 6

puts links[0].text # => Click here

puts links[0]["href"] # => http://www.google.com

On peut aussi filtrer la recherche, par exemple si on veut rechercher exclusivement la class news on peut procéder de cette manière :

page = Nokogiri::HTML(open(PAGE\_URL))

news\_links = page.css("a").select{|link| link['data-category'] == "news"}

news\_links.each{|link| puts link['href'] }

#=> http://reddit.com

#=> http://www.nytimes.com

puts news\_links.class #=> Array

Mais retenons plutôt cette méthode :

news\_links = page.css("a[data-category=news]")

news\_links.each{|link| puts link['href']}

#=> http://reddit.com

#=> http://www.nytimes.com

puts news\_links.class #=> Nokogiri::XML::NodeSet

Ceci nous donnera le texte associé au lien ainsi que l’url contenu dans le lien

page = Nokogiri::HTML(open(PAGE\_URL))

news\_links = page.css("div#references a")

news\_links.each{|link| puts "#{link.text}\t#{link['href']}"}

Si on a du mal à identifier le bon chemin pour l’élément voulu, l’inspecteur web est notre ami!!!

links = page.css(“#toc > ul“)

On peut cibler les éléments aussi avec la méthode xpath

doc = Nokogiri::HTML(html\_document)

doc = Nokogiri::HTML(open("http://www.google.com/search?q=doughnuts"))

Pour parser la page, puis maintenant pour cibler les éléments avec Xpath :

doc = Nokogiri::HTML(open("http://www.google.com/search?q=doughnuts"))

doc.xpath('//h3/a').each do |node|

puts node.text

end

veut dire : Find all "a" tags with a parent tag whose name is "h3"