SIS

Ibraimov Akzhol

Website archiving service

In this work I made website archiving client-server service. Which does the following tasks: Client enters the url of website which he wants to download. On server side used RPC, which makes workers to download pages, and return all links in it. It was realized by BFS algorithm.
This project was written on Ruby on Rails. For working with message queuing I used RabbitMQ server. And Bunny gem, which is ruby client for working with RabbitMQ.
Let's analyza code:

This is our client view, which sends messages to the HomeController#lab5(I updated lab5, so action's name remained “lab5”)

```
#lab5-1/app/views/home/client.html.erb
<div style="width: 400px; margin: 0 auto;">
	<input id="site" name="site" placeholder="Enter web site address" style="float: left" type="text">
	<br><br><br>
	<button id="sub">Submit</button>
</div>
<script>
	$(document).ready(function(){
		$("#sub").click(function(){
			var val = $("#site").val();
			$.ajax({
			  method: "POST",
			  url: "http://localhost:3000/sis",
			  dataType: "json",
			  data: { val: val }
			})
			.success(function(data){
			   document.getElementById('my_iframe').src = "/assets/"+val+"/"+val+".zip";
			});
		});
	});
</script>
<iframe id="my_iframe" style='display: none;'></iframe>
```

HomeController, which creates channel and calls server, answer comes to response

```
#lab5-1/app/controllers/home_controller.rb
```

```
def sis
      val = params[:val]
      path = val
      require "bunny"
      require "thread"

      conn = Bunny.new(:automatically_recover => false)
      conn.start

      ch   = conn.create_channel
      ch.prefetch(2)

      @dir = "/home/akzhol/Repositories/lab5/lab5-1/app/assets/htmls/"

      if File.directory?(@dir+val)
        puts val
      end
      FileUtils.mkdir_p @dir+val
      @used = Hash.new
      @queue = Array.new
      @tmp = 0
      @queue.push(val)
      @used[val] = true
         while @tmp < @queue.length
            if @tmp > 100
               break
            end
            puts @tmp
            val = @queue[@tmp]
            @tmp += 1
            client   = SisClient.new(ch, "rpc_queue")
            puts " [x] Requesting "+val.to_s+""
            response = client.call(path+"+"+val.to_s)
            puts " [.] Got #{response}"
            responses = response.split("+")
            responses.each do |response|
               unless response.length == 0
                  unless response[0] == '/' || response[0] == '#'
                     response = '/'+response
                  end
                  if @used.has_key?(response)
                     next
                  end
                  if response[0..4] == "http"
                     @queue.push(response)
                  else
```

```
              @queue.push(path+response)
            end
            @used[response] = true
          end
        end

      end

    ch.close
    conn.close

    path = @dir+path
    gem 'rubyzip'
    require 'zip/zip'
    require 'zip/zipfilesystem'

    path.sub!(%r[/$],'')
     archive = File.join(path,File.basename(path))+'.zip'
     FileUtils.rm archive, :force=>true

     Zip::ZipFile.open(archive, 'w') do |zipfile|
      Dir["#{path}/**/**"].reject{|f|f==archive}.each do |file|
       zipfile.add(file.sub(path+'/',''),file)
      end
     end


    @s = {}
    @s[:status] = response
    render json: @s
  end
```

Here is class which downloads page and returning all links in it

```
class SisServer

 def initialize(ch)
  @ch = ch
 end

 def start(queue_name)
  @q = @ch.queue(queue_name)
  @x = @ch.default_exchange

  @q.subscribe(:block => true) do |delivery_info, properties, payload|
   r = self.class.parse(payload)
   @x.publish(r.to_s, :routing_key => properties.reply_to, :correlation_id =>
properties.correlation_id)
  end
 end

 def self.parse(url)
  url1 = url.split("+")
  url = url1[1].to_s
  path = url1[0].to_s
  require 'rubygems'
  require 'nokogiri'
  require 'open-uri'
  require 'fileutils'

  @dir = "/home/akzhol/Repositories/lab5/lab5-1/app/assets/htmls/"

  response = ""
  begin
   @main = Nokogiri::HTML(open("http://"+url, :proxy => nil, 
:read_timeout=>10))
   url = url.gsub("/", "\\")
   File.write(@dir+path+"/"+url+".html".to_s, @main.to_html(encoding: 'UTF-
8'))
   @links = @main.css("a")
   @links.each do |link|
     response += "+"+link['href']
   end
  rescue
   dosomething = 1
  end
   return response
 end
end
```