Lab-6

Ibraimov Akzhol

RPC

In this laboratory work I implemented scientific calculator in form of RPC server and client. Which does the following tasks: Client enters the number(s) and choose some mathematical operation to do, they are “factorization”, “is prime”, “factorial”, “exponentiation”. All calculations are done on server side and return answer to client.

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/about.html.erb
<div style="width: 400px; margin: 0 auto;">
	<select name="type" style="float: left">
		<option value="factorization">Factorization</option>
		<option value="is_prime">Is Prime</option>
		<option value="factorial">Factorial</option>
		<option value="exponentiation">Exponentiation</option>
	</select>
	<br><br>
	<input id="val1" name="value" placeholder="Enter your value"
style="float: left" type="number">
	<input id="hid" name="value1" placeholder="Enter second value"
style="float: left; visibility: hidden" type="number">
	<br><br><br>
	<button id="sub">Submit</button>
	<br><br>
	<label>Answer: </label>
	<div id="answer"></div>
</div>
<script>
	$(document).ready(function(){
		$("select").change(function(){
			var val = $(this).val()
			if (val == 'exponentiation')
			{
				$("#hid").css('visibility', 'visible')
			}
			else
```

```
                {
                        $("#hid").css('visibility', 'hidden')
                }
        });
        $("#sub").click(function(){
                var type = $("select").val();
                var val1 = $("#val1").val();
                var val2 = $("#hid").val();
                $.ajax({
                  method: "POST",
                  url: "/lab5",
                  dataType: "json",
                  data: { type: type, value: val1, value1: val2 }
                })
                .success(function(data){
                        $("#answer").html(data.status)
                });
        });
  });
</script>
```

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

```
#lab5-1/app/controllers/home_controller.rb
 def lab5
      type = params[:type]
      value = params[:value]
      value1 = params[:value1]
      if type == 'exponentiation'
        value = value.to_s+" "+value1.to_s
      end

      require "bunny"
      require "thread"

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

      ch   = conn.create_channel

      client   = CalculatorClient.new(ch, "rpc_queue")
      puts " [x] Requesting "+value+""
      response = client.call(type.to_s+"+"+value.to_s)
      puts " [.] Got #{response}"
```

```
      ch.close
      conn.close

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

Here is class which makes calculations on server

```
#lab5-2/app/services/calculator_server.rb
class CalculatorServer

 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|
#     n = payload.to_i

    my_arr = payload.split('+')
    type = my_arr[0]
    value = my_arr[1]
    if type == "factorization"
      r = self.class.factorization(value)
    elsif type == "is_prime"
      r = self.class.is_prime(value)
    elsif type == "factorial"
      r = self.class.factorial(value)
    elsif type == "exponentiation"
      r = self.class.exponentiation(value)
    end
#     r = self.class.fib(n)
    @x.publish(r.to_s, :routing_key => properties.reply_to, :correlation_id =>
properties.correlation_id)
  end
 end

 def self.factorization(value)
  require 'prime'
```

```
    @pd = (value.to_i).prime_division
    return "factorization of "+value.to_s+": "+@pd.to_s
  end

  def self.is_prime(value)
    require 'prime'
    if (value.to_i).prime?
      return value.to_s+" is prime"
    else
      return value.to_s+" is not prime"
    end
  end

  def self.factorial(value)
      f = 1; for i in 1..value.to_i; f *= i; end; f
      return "factorial of "+value.to_s+": "+f.to_s
  end

  def self.exponentiation(value)
    my_arr = value.split(' ')
    val1 = my_arr[0].to_i
    val2 = my_arr[1].to_i

    return "exponentiation of "+val1.to_s+" to degree "+val2.to_s+" is equal to
"+(val1**val2).to_s
  end

  def self.fib(n)
    case n
    when 0 then 0
    when 1 then 1
    else
      fib(n - 1) + fib(n - 2)
    end
  end
end
```