ANF 2012 - Mathrice

Programmation événementielle

Exemple : « WebSocket »

David Delavennat

Définition (tirée de Wikipédia)

- La programmation événementielle se dit d'un type de programmation fondé sur les événements. Elle s'oppose à la programmation séquentielle. Le programme sera principalement défini par ses réactions aux différents événements qui peuvent se produire.

- La programmation événementielle peut également être définie comme une technique d'architecture logicielle où l'application a une boucle principale divisée en deux sections : la première section détecte les événements, la seconde les gère.

- À noter qu'il n'est pas ici question d'interruptions : le traitement d'un événement ne peut pas être interrompu par un autre, à part en des points précis et connus (points qui en fait, créent une seconde boucle événementielle par-dessus la première).

http://dev.w3.org/html5/websockets/

4 The WebSocket interface

IDL

```
[Constructor(DOMString url, optional (DOMString or DOMString[]) protocols)]
interface WebSocket : EventTarget {
  readonly attribute DOMString url;

  // ready state
  const unsigned short CONNECTING = 0;
  const unsigned short OPEN = 1;
  const unsigned short CLOSING = 2;
  const unsigned short CLOSED = 3;
  readonly attribute unsigned short readyState;
  readonly attribute unsigned long bufferedAmount;

  // networking
  [TreatNonCallableAsNull] attribute Function? onopen;
  [TreatNonCallableAsNull] attribute Function? onerror;
  [TreatNonCallableAsNull] attribute Function? onclose;
  readonly attribute DOMString extensions;
  readonly attribute DOMString protocol;
  void close([Clamp] optional unsigned short code, optional DOMString reason);

  // messaging
  [TreatNonCallableAsNull] attribute Function? onmessage;
           attribute DOMString binaryType;
  void send(DOMString data);
  void send(ArrayBuffer data);
  void send(Blob data);
};
```

INSMI

dépasser les frontières

Serveur

```ruby
%w(em-websocket pp socket).each do |lib| require lib end

class Client
  attr_reader :ip, :port
  def initialize(ws) @ws,@port,@ip = ws,*Socket.unpack_sockaddr_in(ws.get_peername) end
  def send(message)  @ws.send(message)       end
end
class Clients
  def initialize()   @clients = []           end
  def add(client)    @clients.push(client)   end
  def del(client)    @clients.delete(client) end
  def send(message)  @clients.each do |client| client.send message end end
end
class Chat
  def initialize()  @clients = Clients.new  end
  def start()
    EventMachine.run do
      EventMachine::WebSocket.start(:host => "127.0.0.1", :port => 4096) do |ws|
        client = Client.new(ws)
        ws.onopen do
          @clients.add(client)
          @clients.send("client #{client.ip}:#{client.port} joined the chat")
        end
        ws.onmessage do |message|
          EventMachine::add_timer(10) do @clients.send("#{Time.now} #{message}") end
        end
        ws.onclose do
          @clients.del(client); puts "WebSocket closed"
        end
      end
    end
  end
end
chat=Chat.new
chat.start
```

INSMI

dépasser les frontières

Serveur

```
%w(em-websocket pp socket).each do |lib| require lib end

class Client
  attr_reader :ip, :port
  def initialize(ws) @ws,@port,@ip = ws,*Socket.unpack_sockaddr_in(ws.get_peername) end
  def send(message)  @ws.send(message)       end
end
class Clients
  def initialize()   @clients = []           end
  def add(client)    @clients.push(client)   end
  def del(client)    @clients.delete(client) end
  def send(message)  @clients.each do |client| client.send message end end
end
class Chat
  def initialize()  @clients = Clients.new  end
  def start()
    EventMachine.run do
      EventMachine::WebSocket.start(:host => "127.0.0.1", :port => 4096) do |ws|
        client = Client.new(ws)
        ws.onopen do
          @clients.add(client)
          @clients.send("client #{client.ip}:#{client.port} joined the chat")
        end
        ws.onmessage do |message|
          EventMachine::add_timer(10) do @clients.send("#{Time.now} #{message}") end
        end
        ws.onclose do
          @clients.del(client); puts "WebSocket closed"
        end
      end
    end
  end
end
chat=Chat.new
chat.start
```

INSMI

dépasser les frontières

Serveur

```
%w(em-websocket pp socket).each do |lib| require lib end

class Client
  attr_reader :ip, :port
  def initialize(ws) @ws,@port,@ip = ws,*Socket.unpack_sockaddr_in(ws.get_peername) end
  def send(message)  @ws.send(message)       end
end
class Clients
  def initialize()   @clients = []           end
  def add(client)    @clients.push(client)   end
  def del(client)    @clients.delete(client) end
  def send(message)  @clients.each do |client| client.send message end end
end
class Chat
  def initialize()  @clients = Clients.new  end
  def start()
    EventMachine.run do
      EventMachine::WebSocket.start(:host => "127.0.0.1", :port => 4096) do |ws|
        client = Client.new(ws)
        ws.onopen do
          @clients.add(client)
          @clients.send("client #{client.ip}:#{client.port} joined the chat")
        end
        ws.onmessage do |message|
          EventMachine::add_timer(10) do @clients.send("#{Time.now} #{message}") end
        end
        ws.onclose do
          @clients.del(client); puts "WebSocket closed"
        end
      end
    end
  end
end
chat=Chat.new
chat.start
```

INSMI

dépasser les frontières

Client javascript

```
<html>
  <head>
    <style>
#debug   { background-color: red; border: 1px solid #000; }
#message { background-color: green; border: 1px solid #000; height: 200px; overflow: auto;
}
    </style>
    <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js'></script>
    <script type="text/javascript">
$(document).ready(function(){
  function debug(str) { $("#debug").append("<p>"+str+"</p>"); };
  function log(msg)   {
    $("#message").append("<span>"+msg+"</span><br />");
    $("#message").animate({ scrollTop: $("#message").attr("scrollHeight") }, 2000);
  };
  var ws = new WebSocket("ws://127.0.0.1:4096/websocket");
  ws.onmessage = function(evt) { log(evt.data); ws.send('toc toc');             };
  ws.onclose   = function(evt) { debug("socket closed");                        };
  ws.onopen    = function(evt) { debug("connected..."); ws.send("hello server"); };
});
    </script>
  </head>
  <body><div id="debug"></div><div id="message"></div></body>
</html>
```

INSMI

dépasser les frontières

Client javascript

```
<html>
  <head>
    <style>
#debug   { background-color: red; border: 1px solid #000; }
#message { background-color: green; border: 1px solid #000; height: 200px; overflow: auto;
}
    </style>
    <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js'></script>
    <script type="text/javascript">
$(document).ready(function(){
  function debug(str) { $("#debug").append("<p>"+str+"</p>"); };
  function log(msg)   {
    $("#message").append("<span>"+msg+"</span><br />");
    $("#message").animate({ scrollTop: $("#message").attr("scrollHeight") }, 2000);
  };
  var ws = new WebSocket("ws://127.0.0.1:4096/websocket");
  ws.onmessage = function(evt) { log(evt.data); ws.send('toc toc');              };
  ws.onclose   = function(evt) { debug("socket closed");                         };
  ws.onopen    = function(evt) { debug("connected..."); ws.send("hello server"); };
});
    </script>
  </head>
  <body><div id="debug"></div><div id="message"></div></body>
</html>
```

INSMI

dépasser les frontières

Client javascript

```
<html>
  <head>
    <style>
#debug   { background-color: red; border: 1px solid #000; }
#message { background-color: green; border: 1px solid #000; height: 200px; overflow: auto;
}
    </style>
    <script src='http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js'></script>
    <script type="text/javascript">
$(document).ready(function(){
  function debug(str) { $("#debug").append("<p>"+str+"</p>"); };
  function log(msg)   {
    $("#message").append("<span>"+msg+"</span><br />");
    $("#message").animate({ scrollTop: $("#message").attr("scrollHeight") }, 2000);
  };
  var ws = new WebSocket("ws://127.0.0.1:4096/websocket");
  ws.onmessage = function(evt) { log(evt.data); ws.send('toc toc');              };
  ws.onclose   = function(evt) { debug("socket closed");                         };
  ws.onopen    = function(evt) { debug("connected..."); ws.send("hello server"); };
});
    </script>
  </head>
  <body><div id="debug"></div><div id="message"></div></body>
</html>
```

INSMI

dépasser les frontières

Démo

Références

- http://www.html5rocks.com/en/tutorials/websockets/basics/
- http://fr.wikipedia.org/wiki/Programmation_événementielle
- http://eventmachine.rubyforge.org/EventMachine.html
- http://kisdigital.wordpress.com/2009/11/12/using-jquery-to-scroll-to-the-
- http://stomp.github.com//stomp-specification-1.1.html
- http://www.slideshare.net/ismasan/websockets-and-ruby-eventmachine
- http://www.igvita.com/2009/12/22/ruby-websockets-tcp-for-the-browser/