Report 1: Seminar 2 - Rudy: a small web server

Fleitz Pierre

September 16, 2015

1 Introduction

During this seminar, I tried to implement a small web server in Erlang. For the last few years I've been working a lot with client and server in some previous classes in France. All the time mostly in C. Therefore this seminar was a chance for me to discover how to code a web server in Erlang, a small one but still enough for me to understand. This is with that mind setup that I started this seminar. The main goals of that seminar was to try to give an answer to the following questions : - How is the Socket API made in Erlang? How is it different from what I am used to ? - How to structure a server process ? - How to implement (a part of) the HTTP Protocol ?

And of course it was a chance to code in Erlang and practice a little bit with that new language.

2 Main problems and solutions

The first problem I had to face in this seminar was with the first reply.

How to fill the missing pieces of the code the subject gave us so I can retrieve a page from my browser ?

It requiered to take a more interesting look at the Socket API and at this moment I was quite happy to see that it was really similar from the Socket API in C. Actually I founded it really interesting to compare Erlang and C on that point. After that, when you understand more how the Socket API works and what we expect from our functions it is quite easy to fill the code and to retrieve a page (in my example a simple phrase saying "Ezpz"). I used Safari, even if it's not relevant at all.

Later on we needed to implement a server able to run and provide a service until we ask him to stop. Indeed if we remember the goal of this seminar, we want to code a little web server and a web server is not something that shuts down after only one request. The code is given and the only thing we need to do is to call handle/1 recursivly in rudy.erl. What handle/1 is going to do is that it will receive the socket from the request connexion from init/1, it will accept the connexion with gentcp:accept/1 and then send

the request to request/1 who will parse the request and manage to send a reply. After all of this, we don't want the server to stop, so we call again handle/1 who will wait another request connexion from init/1. The code is not really hardcore it's only a recursivly call but still :

```
...
{ok, Client} ->
  request(Client),
  handler(Listen) ; %% We call handler recursivly
...
```

Now that we have all of this, we want to be able to run the server on a machine, let's call it machine A and to access A from another machine called B.

I can understand why it is important to be able to reach a server from a different machine.. This is the point of a web server. I think I have done everything in my code to test it but I didn't reach that point of the subject during the first session of the seminar and I had to finish it at home, so I did not had another machine to try...

But again for that part the code is given the only thing that I did was to try to understand the small benchmark program.

And if I want someone to access my server from B I just have to start it on a port for example 8080 and from another machine lunch test:bench(A,8080) with A the IP address of the machine A (where the server is running) and it will measure the time it took to send a request and to receive the answer from A.

So what I did is that I launched rudy:start(8080). in a terminal 1 and test:bench(A,8080) in a terminal 2 at least to see what was the output of bench/2. With A = 192,168,1,13 because it is a inet:ipaddress() type. I have been able to see that when I looked on the gentcp:connect() function in the Socket API.

I'll explicit my results and experimentations in the next part of the report.

3 Evaluation

I decided to look at the result of test:bench(A,8080) with different values of sleep(delay).

The function timer:nowdiff(Finish, Start) return the time in microseconds according to http://www.erlang.org/doc/man/timer.html.

Delay added (ms)	Return from test:bench(A,8080) (ms)	Return from test:bench(A,8080) (s)
0	27309	0.03
10	1274737	1,27
20	2274782	2,27
30	3271629	3,27
40	4276108	4,28

Table 1: Results from test 1

As we can see with the timer:sleep(40) (who is supposed to add a delay of 40 ms to our server) the result of timer:bench(A,8080) is 4,28 seconds. It sounds pretty long to me. It seems a little bit weird to me to have 4 seconds of delay when I only added 40 ms with the sleep. I tried to check there http://www.erlang.org/doc/man/timer.html but it does say that timer:sleep(time) is in milliesconds so timer:sleep(40) should stop the process for 40 ms and not 4 seconds like right now. I have to admit that I don't know where this is coming from.

Then I decided to lunch different requests from different terminals and see how long it took (with a timer:sleep(20) so I can have the time to lunch different requests):

Number of similar request	Rtest:bench(A,8080) avg (s)	Return for 1 request (20ms delay)
2	3,41	2,27
3	4,90	2,27
4	5,52	2,27

Table 2: Results from test 2

We can see how the throughput is affected, of course as we could expected it's way longer for the server to respond at different simultaneous requests than only one.

After different research I still can't explain why for sleep(40) we have 4 seconds of delay and not 40 ms as expected. And that is one of the first question I am going to ask during the second session.

4 Conclusions

I think this seminar has been really usefull for me. It helped me to improve a little bit my Erlang skill, to understand a little bit more the philosophy behind this language.

It gave me more experience with Erlang and errors you can face. It gave me some knowledge or reminders about the HTTP Protocol that I studied in the past, and the Socket API. I have been able to see how to code a little web server in Erlang, how to use TCP for that and most importantly how different it is from another language like C that I used a lot to solve some client/server problems in the past few years.

Another thing is that this seminar is the first real seminar I had here in Sweden. As an exchange student from France I have to admit that I really enjoyed this type of exercice and how the subject is really helpful to understand what we are doing. This is very different from France and I feel like I truly learnt something from that seminar.