AboutFishbowl

January 16, 2015

1 Fishbowl: an enhanced interactive Shell for Common Lisp

This document is a short presentation of *Fishbowl* which is, technically-speaking, a Common Lisp implementation of an IPython kernel. So what does this means in practice ? Let's see.

1.1 A (somewhat poor man's) distributed Lisp REPL

Basically IPython is a better shell (understand REPL) for the Python programming language. So what the h.ck is the relationship with Lisp ? Well, the IPython architecture allows distributed (unsecure !) interactions between :

- clients – called *frontends* – that manage the user-part of the interactions : reading expressions, printing results and such, and
- servers – called *kernels* – that actually perform the computations.

So in an IPython REPL, the frontends play the R and P parts, while the kernels play the E part. Fishbowl is thus the Eval part while the existing IPython frontends can be used for Reading and Printing. Basic interactions would look like the following :

```
In [1]: (+ 2 12)

Out[1]: 14
```

Things happen like these :

- first, the frontend reads the Lisp expression `(+ 2 12)`
- this expression is sent, through the network (ZMQ sockets to be precise), to the Fishbowl kernel, which performs the corresponding computation (thanks to the `eval` function of course).
- the resuling value `14` is then sent back to the frontend for printing, and that's it.

In case the expression yields side effects, such as writing a file, these are actually performed by Fishbowl so one should take care servicing kernels only on private networks ! The standard output and error streams are captured and thus visible from the frontend side.

```
In [2]: (format t "I compute (+ 2 12), yielding ~A" (+ 2 12))

I compute (+ 2 12), yielding 14

Out[2]: NIL
```

The second expression, namely `In [2]`, yields the `NIL` value, namely `Out [2]`, but the formatted string written on the `*standard-output*` stream is also printed.

Errors are also printed,as the following (counter-)example shows :

```
In [3]: (/ 2 0)

DIVISION-BY-ZERO:
  #<DIVISION-BY-ZERO {1007988493}>

Out[3]: NIL
```

We remark that, at least in the current version, Fishbowl does *not* support interactions with the Lisp debugger.

This is a good moment to stay something important :

> Fishbowl is **not** intended as a replacement for SlimE or SlimV.

Indeed, this would imply replacing Emacs of Vim for starting with ...

At the very best, Fishbowl (together with the IPython frontend) could be seen as a replacement for the basic REPL of most Common Lisp implementations (as for now without the debugger). An interface with Swank (the "kernel" part of SlimE/V) would be a strong improvement here, but that is only for now a wish (contributors welcome !).

Some features might already prove useful. First and foremost, the distributed architecture of IPython/Fishbowl allows to connect multiple frontends to multiple kernels, as well as exchanging Data. Since IPython kernels exist for a growing number of programming languages (Python, Ruby, Julia, Ocaml, Haskell, etc.), this opens many possibilities such as writing interactive distributed applications developped in a multi-languages environment (which appeals much more to me than the *everything-in-javascript* trend of the moment).

But a question remains: is *Fishbowl* bringing something to the Lisp table ? Obviously, a not-really-better REPL does not. The main reason why *Fishbowl* was written in the first place is for supporting IPython *notebooks*.

1.2 A Lisp environment for interactive documents

So what is a notebook ? Well, that's easy, this is what you are reading now !

It is a document mixing :

- *Markdown*-formatted **text** (optionally including *Latex/Mathjax* formulas)
- **Computations** described in various language, Common Lisp as far as Fishbowl is concerned.

For example, *Leonardo* may write a fantastically clever algorithm for an important computation :

```
In [4]: (defun fibonacci (n)
            (if (<= n 1)
                1
                (+ (fibonacci (- n 2)) (fibonacci (- n 1)))))

Out[4]: FIBONACCI
```

And then, Leonardo can answer one of its most desired question :

```
In [5]: (fibonacci 10)

Out[5]: 89

In [5]: (loop for k below 10 collect (fibonacci k))

Out[5]: (1 1 2 3 5 8 13 21 34 55)
```

and then, comment about such computations, such that making conjectures or wonderings...

What about $\lim_{k\to+\infty} \frac{F_{k+1}}{F_k}$ with F_k the k-th term of the Fibonacci serie ?

Well, let's check this ...

```
In [7]: (loop for k from 2 below 20 collect (/ (fibonacci (1+ k)) (fibonacci k)))

Out[7]: (3/2 5/3 8/5 13/8 21/13 34/21 55/34 89/55 144/89 233/144 377/233 610/377 987/610 1597/987 2584/
```

Oops ... don't want the exact rational values but approximations ...

```
In [6]: (loop
          for k from 2 below 20
          collect (float (/ (fibonacci (1+ k)) (fibonacci k))))

Out[6]: (1.5 1.6666666 1.6 1.625 1.6153846 1.6190476 1.617647 1.6181818 1.6179775
         1.6180556 1.6180258 1.6180371 1.6180328 1.6180345 1.6180338 1.618034 1.618034
         1.618034)
```

What is this value 1.618034 ? ... seems not far from ... $\frac{1+\sqrt{5}}{2}$

```
In [7]: (/ (+ 1 (sqrt 5))
           2)

Out[7]: 1.618034
```

Don't you think this number looks shiny ?

This (probably unsuccessful) half-joke at least summarizes the way I am using notebooks in my own teaching and research work.

At the technical level, the IPython notebook server is simply a frontend developped as a complete web application that can connect to any kernel, among which of course Fishbowl.

1.3 Rich display

The notebook provide rich display for textual and graphical data.

```
In [4]: (svg-from-file "profile/fishbowl-small.svg")

Out[4]:
```

1.4 To be continued ...