Lisp ERP

Francesc Guillén

June 7, 2019

Lisp ERP

Get Hy with Tryton

Tryton Rocks

- local support
- open source / org / foundation
- community
- tech: architecture, postgres, scenarios, testing, ...

My path to Lisp

What's this thing?

- AI course

I want it

- Paul Graham - Hackers and painters *2005*
- Abelson & Sussman - SICP lectures and book

Yes I can!

- Rich Hickey - Clojure

Lisp?

- By John McCarthy 1958ish at MIT.
- Lisp : family of programming languages.
- Flavors. Common Lisp, Scheme, Racket...
- Functional. Dynamic. Garbage Collected. Code-as-data. REPL...

New Flavors of Lisp

- Java *1995* -> Clojure *2007*
- Erlang *1986* -> LFE *2008*
- Javascript *1995* -> Clojurescript (cljs) *2011*
- Python *1990* -> Hy *2013*

Clojure for JVM *vs* Hy for Python

Clojure Hosted (not Javaistic)

"Clojure is a robust, practical, and fast programming language with a set of useful features that together form a simple, coherent, and powerful tool."

- dynamic
- own data structure
- control mutation/state
- ...

Hy Embedded (Pythonistic)

"Hy is a wonderful dialect of Lisp that's embedded in Python."

"Since Hy transforms its Lisp code into the Python Abstract Syntax Tree, you have the whole beautiful world of Python at your fingertips, in Lisp form!"

- Python data structures
- Python truthiness

What's special about Lisp?

- S-expression
- simple syntax
- infx notation
- code-as-data - meta-programming

(operation operant1 op2 . . .)

```
(+ 1 2 3)
(if (event? 2) "yes" "what?")
(= 1 2)
(defn inc2 [x] (+ 2 x))
```

((((((()))))))

- buy one get two (paredit, lispy . . .)
- print('hello') -> (print "hello")

Test hy

Hy Tryton Modules

- pip install hy
- create a module
- __init__.py (as usual + import hy)
- Write hylang files .hy files instead of python.
- You still have to write the same xml files :)

Let's look at some code.

hello.hy

```
(import [trytond.model [ModelSQL ModelView fields]])

(defclass Hello [ModelSQL ModelView]
  "Hello World"
  [--name-- "hello"
   name (.Char fields "Name" :required True)
   greeting (.Function fields
                       (.Char fields "Greeting")
                       "get_greeting")]

  (defn get-greeting [self name]
    (.format "Hello {}" self.name)))
```

Example macro default

```
@classmethod
def default_company(cls):
    return Transaction().context.get('company')
```

```
(with-decorator classmethod
   (defn default-company [cls]
      (.get (. (Transaction) context) "company")))

(default company
    (.get (. (Transaction) context) "company"))
```

Macro source

```
(defn default-func-name [name]
  (+ "default_" (.replace name "-" "_")))

(defmacro default [field args &rest body]
  `(with-decorator classmethod
     (defn ~(HySymbol (default-func-name (name field)))
       ~(+ [(HySymbol "cls")] args) ~@body)))
```

Why learn Lisp?

"free your mind and the rest will follow"

Thanks / Questions?

fgui (github & bitbucket)

References

- Hy hylang
- Tryton
- Clojure
- Rich Hickey Transcripts Talks
- Paul Graham