Techniques for lightweight DSL development in Converge

Laurence Tratt
http://tratt.net/laurie/

2007/07/02

Background

- Renewed talk of DSLs, particularly in Ruby.

Background

- Renewed talk of DSLs, particularly in Ruby.
- Are library calls really all a DSL is?

Background

- Renewed talk of DSLs, particularly in Ruby.
- Are library calls really all a DSL is?
- Rich DSLs require new syntax...
- ...but parsing, compilation, error checking etc. are often *hard*.

Background

- Renewed talk of DSLs, particularly in Ruby.
- Are library calls really all a DSL is?
- Rich DSLs require new syntax...
- ...but parsing, compilation, error checking etc. are often *hard*.
- A solution: an extensible programming language. Converge.

What is Converge?

Converge has a number of influences. Relevant ones include:

- is dynamically, but strongly typed (think Python).
- is compiled to bytecode and run by a VM (think Java).
- can perform compile-time meta-programming (as Template Haskell, but probably easiest to think of macros in LISP/Scheme).
- can have its syntax extended (think MetaBorg).

Compile-time meta-programming

This is the tricky, interesting bit!

Compile-time meta-programming

This is the tricky, interesting bit!

Expression	`2 + 3`	evaluates to `5` as one expects.
Splice	`$<x>`	evaluates `x` at compile-time; the AST returned overwrites the splice.
Quasi-quote	`[| 2 + 3 |]`	evaluates to a *hygienic* AST representing `2 + 3`.
Insertion	`[| 2 + ${x} |]`	'inserts' the AST `x` into the AST being created by the quasi-quotes.

An example

```
func expand_power(n, x):
  if n == 0:
    return [| 1 |]
  else:
    return [| ${x} * ${expand_power(n - 1, x)} |]

func mk_power(n):
  return [|
    func (x):
      return ${expand_power(n, [| x |])}
  |]

power3 := $<mk_power(3)>
```

means that `power3` looks like:

```
power3 := func (x):
  return x * x * x * 1
```

by the time it is compiled to bytecode.

What does this have to do with DSLs?

- Macros allow us to ‘compile out’ DSLs at compile-time.
- Converge has lots of support for DSL creation, debugging etc.
- Talk #1: Introduction to Converge, macros, and simple DSL creation.
- Talk #2: Quick introduction to Converge and macros, and slightly more advanced DSL creation.