Ruby Metaprogramming

CSCI400

21 September 2017

Color Key

- Clickable URL link
- Write down an answer to this for class participation
- Just a comment – don't confuse with yellow

Overview

Metaprogramming

- Treating programs as data
  - Writing *programs that write programs* (!)
- Common application: domain-specific languages (DSL)

TODO: include “you’ve all heard of at least one Ruby DSL” question?

Familiar Example (1/2)

```ruby
class Pikachu
    attr_accessor :name
    attr_reader :level

    def initialize(name, level)
        @name = name
        @level = level
    end
end
```

Familiar Example (2/2)

Let's replicate `attr_*`

Check out this Ruby source file

Quick Exercise

- **Goal:** Create small class with a few instance vars
- Start with source file on previous slide
  - Use to create getters/setters
- *Hint:* `require_relative` and `include`

`require` and `include`

Recall

- `require`
  - Like `include` in other languages (e.g. C++)
  - Runs another source file
  - Ensures its not `required` twice
- `include`
  - Imports modules for use as *mix-ins*

DSL Overview

Domain-Specific Language

- General-purpose language (GPL)
  - Used to solve problems in many domains
  - E.g. Python, Java, Ruby, C++, Haskell, ...
- \hl{Domain-specific languages)
  - Used to solve problems within in *specific domain*

DSL Examples

Language	Domain
HTML	Web pages
Mathematica	Symbolic math
GraphViz	Drawing graphs
VHDL	Hardware description
YACC	Define parsers
Regular Expressions	Lexers
SQL	Relational-database queries

Internal vs. External DSL

- External DSL (eDSL)
  - *Independent* of any other language
  - Compiled/interpreted (like a GPL)
- Internal DSL
  - AKA *embedded* DSL
  - DSL *within* a (more) general-purpose language

Short post describing the distinction

Don't need to know the difference for exam

eDSL Example