HACKING TYPES AND PROVIDERS

PUPPETCONF 2015, PORTLAND OR

WHO IS THAT CAT?

Felix Frank

- Server Op from Berlin
- Puppet user/contributor since 0.24
- code contributor since ~3.4
- Twitter @felis_rex
- GitHub @ffrank

CAVEAT: MOSTLY SELF-TAUGHT

FOCUS WILL BE ON DEBUGGING
RATHER THAN DESIGNING

AGENDA

1. The Type class and DSL
2. Providers
3. Where to start
4. Live demo

TYPES

REMEMBER

- lib/puppet/type(.rb)
- lib/puppet/provider(.rb)

ALSO

- lib/puppet/property(.rb)
- lib/puppet/parameter(.rb)

DON'T BE MISLEAD

all the `Resource` classes hardly relate

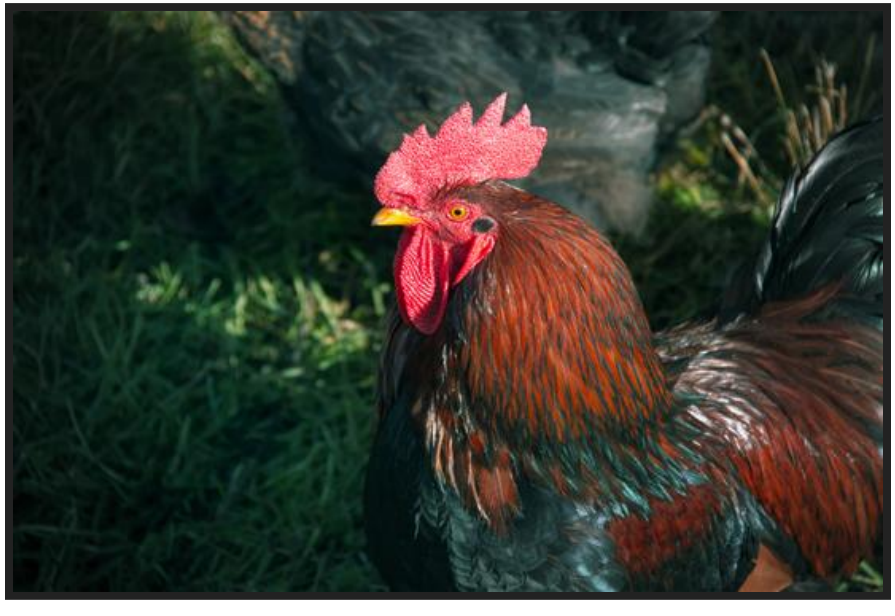
- lib/puppet/parser/resource.rb
- lib/puppet/parser/ast/resource.rb
- lib/puppet/face/resource.rb
- lib/puppet/resource.rb
- lib/puppet/application/resource.rb

WHEN READING TYPE CODE

KEEP SOME RUBY IDIOSYNCRACIES IN MIND

JAVA

CLASS



https://en.wikipedia.org/wiki/Rooster#/media/File:El_Gallo_Ukraine.jpg

INSTANCE

https://en.wikipedia.org/wiki/Otter#/media/File:Fischotter,_Lutra_Lutra.JPG

RUBY

CLASS

INSTANCE

---

http://www.creationscience.com/onlinebook/LifeSciences13.html

https://en.wikipedia.org/wiki/Otter#/media/File:Fischotter,_Lutra_Lutra.JPG

CLASSES ARE INSTANCES

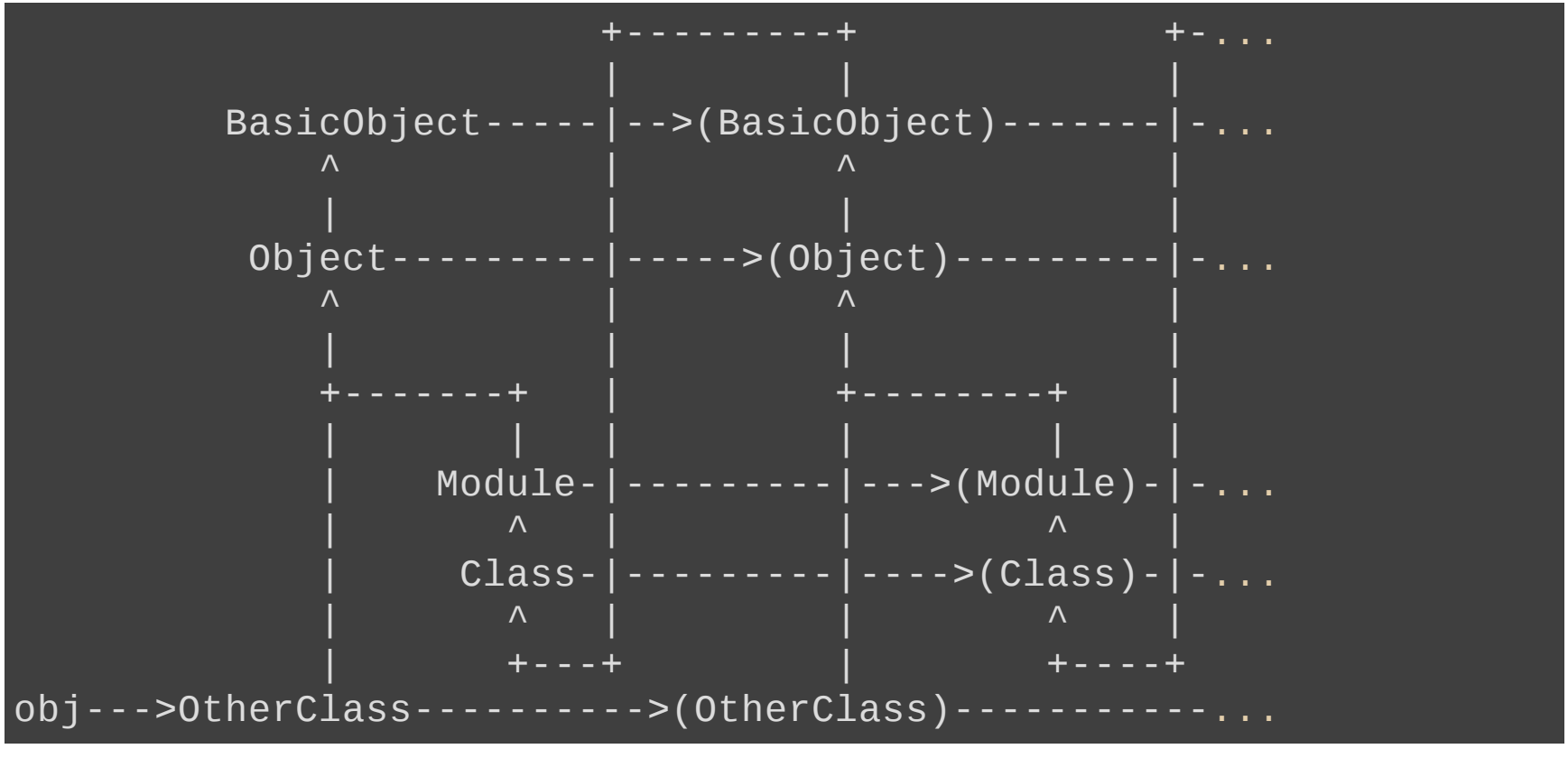
```
Class String ... end
```

same as

```
String = Class.new do ... end
```

CLASS CLASS : MODULE

AN "OVERVIEW"



http://ruby-doc.org/core-2.2.0/Class.html

EVERYTHING IS VERY CONFUSING

http://theawesomedaily.com/21-things-that-look-exactly-like-donald-trump/

WHAT'S IMPORTANT: BOTH CLASSES AND INSTANCES HAVE STATE AND BEHAVIOR

Puppet types take advantage of that

DEFINING NATIVE TYPES CREATES SUBCLASSES

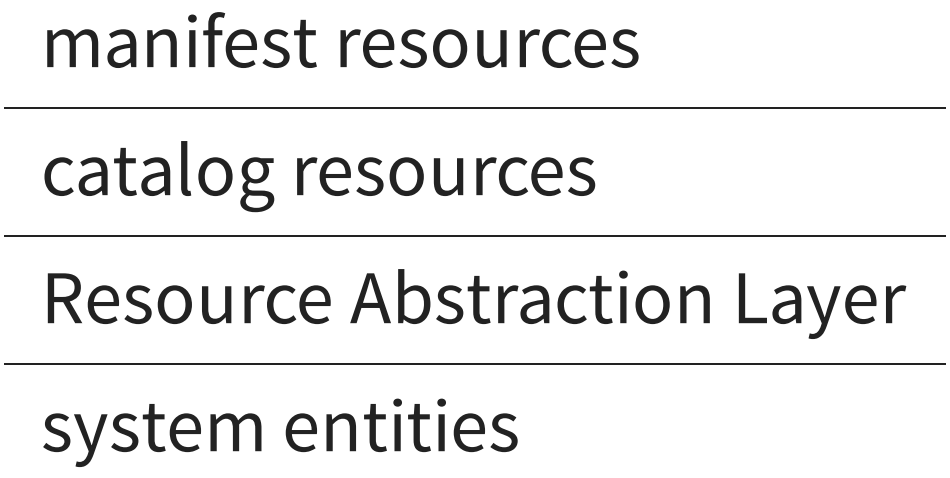
This code...

```
Puppet::Type.newtype(:cron) do
  # code!
end
```

...effectively gives you:

```
class Puppet::Type::Cron : Puppet::Type do
  # generated code!
end
```

LET'S GET META


manifest resources
catalog resources
Resource Abstraction Layer
system entities

I: MANIFEST RESOURCES

```
file { '/etc/motd':
    mode => 644
}
cron { 'break-all-the-things':
    command => '/opt/scripts/cleanup.rb'
}
```

II: CATALOG RESOURCES


```
{
  "type": "File",
  "title": "/etc/motd",
  "tags": ["file","class"],
  "file": "/tmp/example-manifest.pp",
  "line": 1,
  "exported": false,
  "parameters": {
    "mode": 644
  }
}
```

III: RAL RESOURCES

(resource abstraction layer)

- complete representation of a system entity
- properties and parameters
- does all the interesting work

IV: SYSTEM ENTITIES

```
-rw-r--r-- 1 root root 0 Feb  5  2011 /etc/motd
```

```
# HEADER: ...

# Puppet Name: break-all-the-things
1 1 2 * * /usr/scripts/cleanup.rb
```

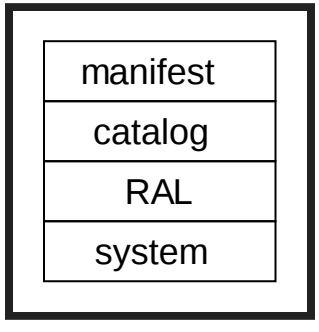
http://www.theawl.com/2010/02/church-boring

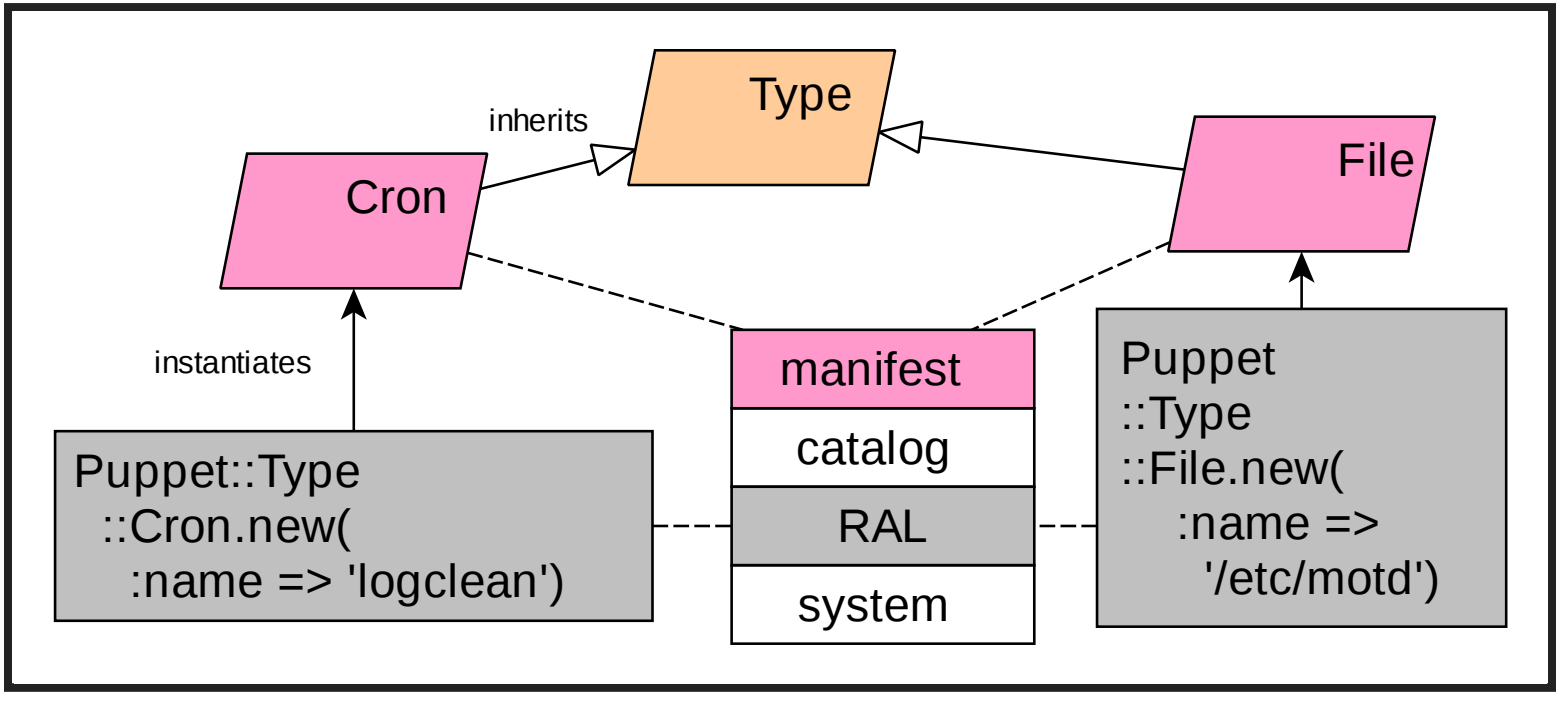
NOW BACK TO THE CODE

So what about this

```
class Puppet::Type::Cron : Puppet::Type do
  # generated code!
end
```

versus this




Type
inherits
Cron
File
instantiates
manifest
catalog
RAL
system
Puppet::Type
::Cron.new(
:name => 'logclean')
Puppet
::Type
::File.new(
:name =>
'/etc/motd')

WHEN READING THE PUPPET : : TYPE CODE, BE MINDFUL OF CLASS VS. INSTANCE SCOPE

RUBY - KNOW THYSELF

- class method:

```
def self.allattrs
```

- instance method:

```
def retrieve
```

CAUTION!

```
class Puppet::Parameter
  class <<self
    # ...
    def validate
      # ...
    end
  end
end
```

`Puppet::Parameter::validate` is a class method

READING TYPE CODE

DSL MAINLY CONSISTS OF CALLS TO CLASS METHODS OF PUPPET::TYPE

```
newproperty(:user) do
```

```
newparam(:name) do
```

IT IS GENERALLY SAFE TO READ TYPE CLASS METHODS TO LEARN THE DSL

Things like `self.newparam` or `self.ensurable`

SOME ARE LIFECYCLE METHODS THOUGH

`self.instances` is pretty neat

THIS TALK IS GOING GREAT

http://devopsreactions.tumblr.com/post/129272206864/how-i-picture-the-thoughts-of-someone-saying-that

INTERLUDE: PROPERTIES AND PARAMETERS

MNEMONIC: PROPERTIES CAN AND WILL SYNC

Parameters can never do that

E.G. FILE CONTENT

Classic property: Is the content on disk?

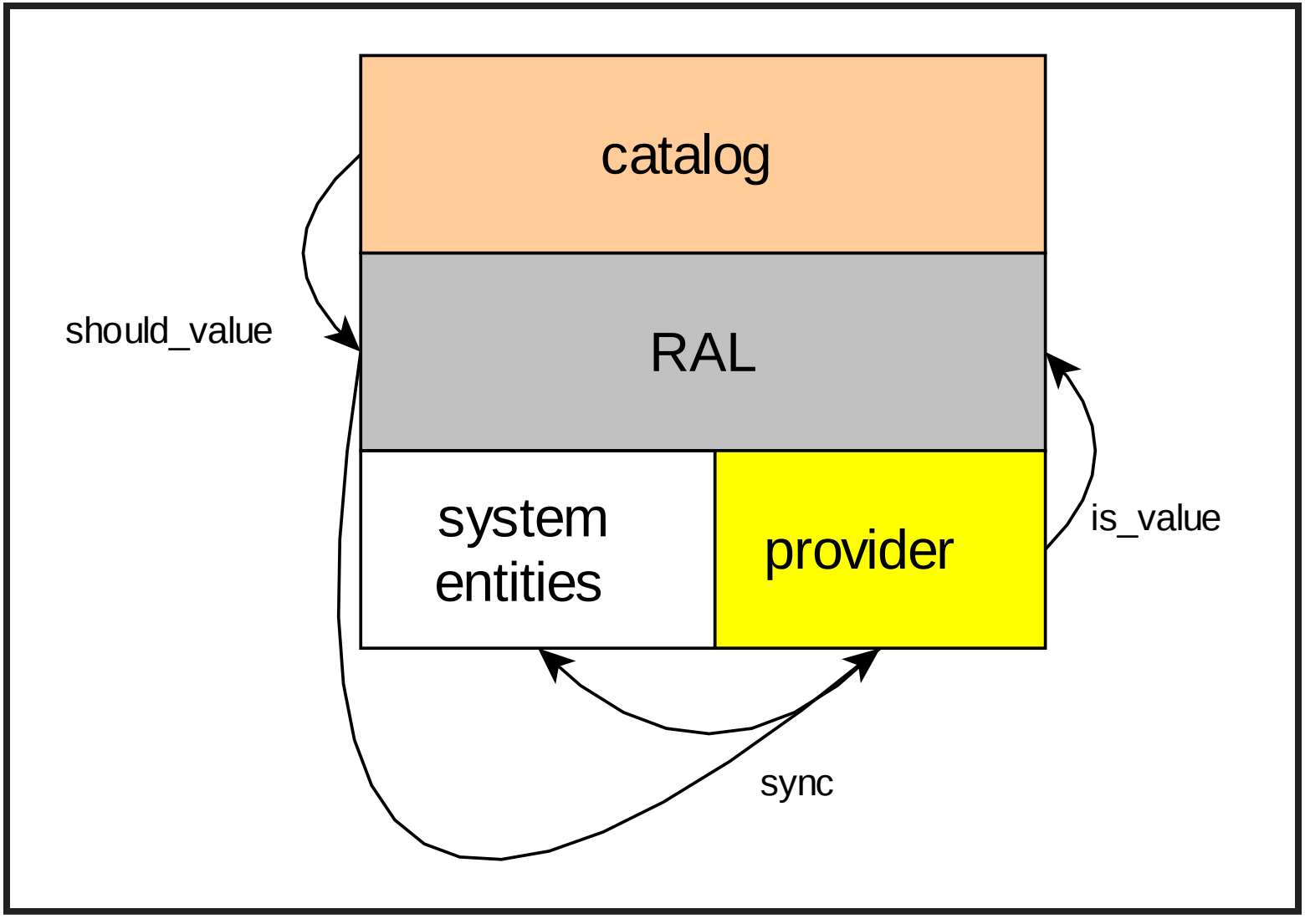
E.G. `INSTALL_OPTIONS`

Classic parameter: Unsyncable, since flags cannot be reset after the fact

PROVIDERS

TYPES VS. PROVIDERS

WHICH DOES WHAT?


catalog
RAL
should_value
system
entities
provider
is_value
sync

FOR EXAMPLE: CATALOG

```
{
  "type": "Package",
  "title": "cowsay",
  "parameters": {
    "ensure": "installed"
  }
}
```

SHOULD VALUE

`Package[cowsay]/ensure: present`

`Puppet::Provider::Package.prefetch`

takes a hash argument

```
{ 'cowsay' => Package[cowsay] }
```

...where `Package[cowsay]` is an instance of `Puppet::Type::Package`

`prefetch` in turn calls

`Puppet::Provider::Package::Dpkg.instances`

assuming Dpkg or Apt is the selected provider

the `instances` hook produces
a list of `Provider` instances

ENTITIES THAT CANNOT BE ENUMERATED OR JUST NOT FOUND GET A "FRESH" PROVIDER

```
Type::Package.provider(:apt).new
```

or in other words

```
Provider::Package::Apt.new
```

FINALLY PROPERTIES CAN BE CHECKED AND SYNCED ONE BY ONE

PROVIDERS ARE STRAIGHT FORWARD

...EXCEPT WHEN THEY INHERIT PARSEDFILE

`lib/puppet/provider/parsedfile.rb`

- cron
- mount
- host
- sshkey
- ...

ParsedFile peculiarities

THERE IS A SPECIFIC DSL

See `Util::FileParsing`

```
text_line   :comment, :match => /^\s*#/
text_line   :blank,   :match => /^\s*$/
record_line :parsed,
  :fields   => %w{options type key name},
  :optional => %w{options},
```

ParsedFile peculiarities

NOTION OF *RECORDS* TO REPRESENT RESOURCES

```
def self.prefetch_all_targets(resources)
  records = []
  targets(resources).each do |target|
    records += prefetch_target(target)
  end
  records
end
```

TARGETS ARE THE MANAGED FILES

ParsedFile peculiarities

INSTANCE METHODS WILL OFTEN RELY ON CLASS METHODS

```
def flush
  # ...
  self.class.flush(@property_hash)
end
```

ParsedFile peculiarities

ACTION IS DEFERED TO FLUSH

(true for some other providers as well)

```
def self.flush
  @modified.each do |target|
    flush_target(target)
  end
end
```

ParsedFile peculiarities

FLUSHING IMPLIES GATHERING GLOBAL STATE (RECORDS)

```
def self.flush_target(target)
  backup(target)
  records = target_records(target)
  target_object(target).write(to_file(records))
end
```

WHEN DEBUGGING PARSEDFILE PROVIDERS, KEEP THE CONVOLUTED FLOW OF LOGIC IN MIND

ASIDE: PLEASE AVOID CREATING NEW PARSEDFILE PROVIDERS

Look at augeas providers instead

SUMMARIZING

WITH TYPES, KEEP CLASS VS. INSTANCE LEVEL IN MIND

PROVIDER INSTANCES GET PAIRED WITH RAL RESOURCES, I.E. TYPE INSTANCES

PARSEDFILE PROVIDERS ARE A CAN OF WORMS

LEARNING THEIR STRUCTURE IS REWARDING BUT WILL NOT HELP YOU WITH OTHER PROVIDERS AT ALL

WHERE TO START

HOW DO I GET STARTED WITH RESOURCE TYPE AND PROVIDER DEVELOPMENT?

READ CODE, PERHAPS TRY AND FIX A BUG OR THREE

MOST OF YOU WILL WANT TO ENHANCE MODULES

WORKING ON CORE PUPPET HAS SOME ADVANTAGES

TEST COVERAGE IS GOOD

WEEKLY TRIAGE SESSIONS

BUGS ARE PLENTIFUL

ALMOST THERE

NOW SOME FINAL WORDS, THEN Q&A AND DEMO!

THE WHITE DUDE RATIO IS OURS TO CHOOSE

RECOMMEND MY BOOK?

USA

PLEASE DON'T MAKE MR. TRUMP YOUR PRESIDENT

Q&A