Puppet Type and Provider Execution

Trevor Vaughan - Onyx Point, Inc.

What We're Doing Here

- Follow Along at Home!
- Introduction to Types and Providers
- Basic Type Code Walkthrough
- Basic Provider Code Walkthrough
- Type Execution Order
- Provider Execution Order

Follow Along at Home!

- Snag a copy of the walkthrough code at http://git.io/_Bjamg

```
git clone https://github.com/onyxpoint/onyxpoint-learning_custom_types.git
```

Why Custom Types?

I can just use execs!

```
# Custom Fact
if $facts['awesome_installed'] {
  exec { 'something_awesome':
    # Hopefully, this exists!
    command => '/usr/local/sbin/awesomeness',
    # Oh dear...
    onlyif  => '/bin/test ! -f /usr/local/share/awesome/conf -o \
                /usr/bin/pgrep awesome ...(you get it)'
  }
}
```

Different Operating Systems

```
if $facts['awesome_installed'] {
  if $facts['osfamily'] == 'RedHat' {
    $onlyif = '/bin/test ! -f /etc/awesome.conf -o \
               /usr/bin/pgrep awesome...'
  }

  if $facts['osfamily'] == 'Debian' {
    $onlyif = '/bin/test ! -f /etc/defaults/awesome -o \
               /usr/bin/pgrep deb_awesome...'
  }

  exec { 'something_awesome':
    command => '/usr/local/sbin/awesomeness',
    onlyif  => $onlyif
  }
}
```

Translation

```
# Confine Statement
if $facts['awesome_installed'] {
# Red Hat Provider
  if $facts['osfamily'] == 'RedHat' {
    $onlyif = '/bin/test ! -f /etc/awesome.conf -o \
               /usr/bin/pgrep awesome...'
  }

# Debian Provider
  if $facts['osfamily'] == 'Debian' {
    $onlyif = '/bin/test ! -f /etc/defaults/awesome -o \
               /usr/bin/pgrep deb_awesome...'
  }

  exec { 'something_awesome':
    command => '/usr/local/sbin/awesomeness',
    # insync?
    onlyif  => $onlyif
  }
}
```

Also

- More Flexibility
- Better Code Testing Capability
- Ability to Manipulate the Catalog
- Automatic Resource Chaining

But...

- More Complex
- More Difficult to Debug
- Requires Programming Ability

What are Types? - The Basics

Properties

Something that can be measured and changed on the system

Parameters

Change how Puppet manages resources through the Properties

Think of them as variables

Validation

Anything that you need to do to validate your resource

Autorequires

Provides the ability to require other catalog resources if they happen to be defined

Does **not** require resources to be defined!

What are Types? - Advanced

Initializer

Provides the ability to manipulate the resource mid-compile

Finish

Anything that you want to do to your resource at the end of the compilation

Post-Resource Evaluation

Magic! Allows for a resource cleanup (Puppet > 3.4.0 only)

What are Providers?

Initializer

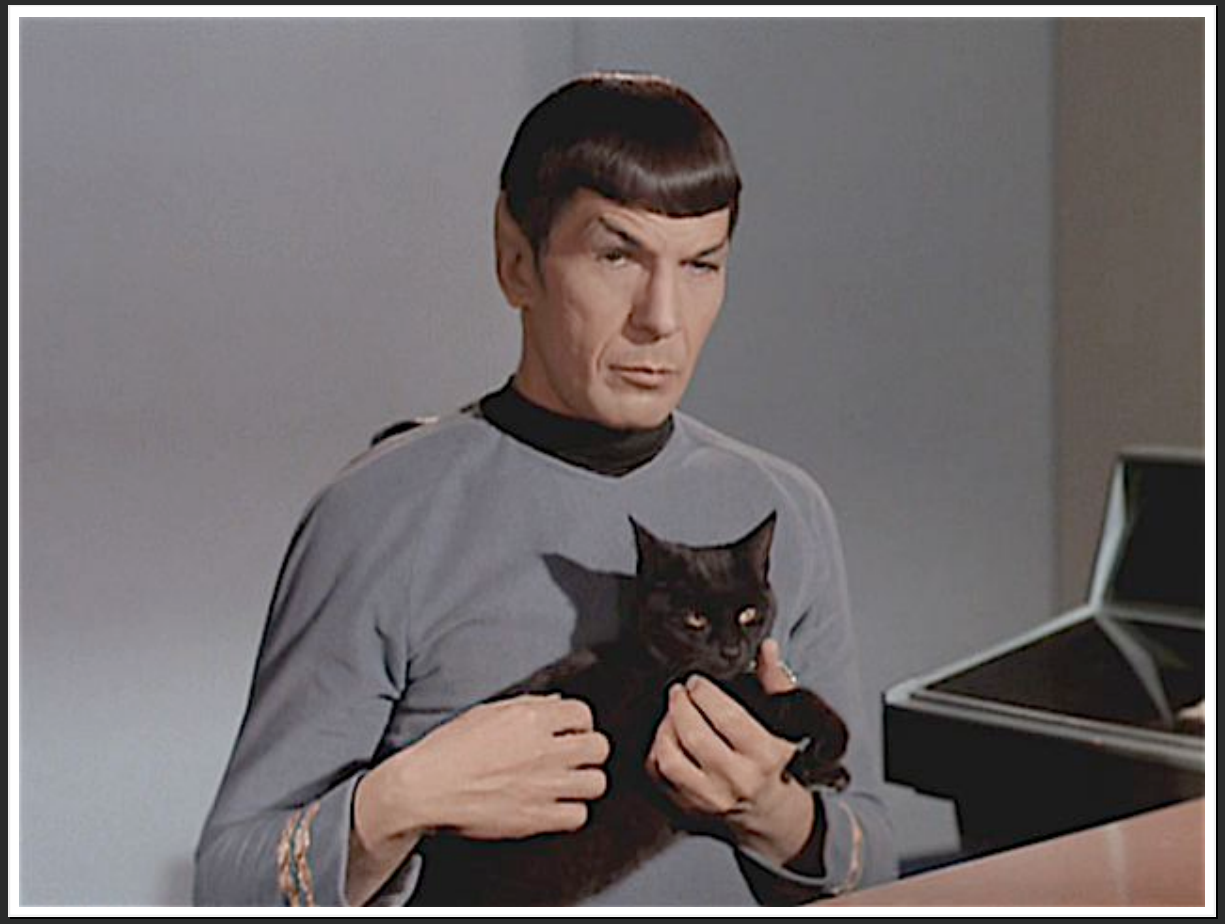
A place to do all sorts of fun things that persist across the provider

Property Methods

Automatically referenced methods for manipulating properties on the system

Miscellaneous

Any other method that is useful to your type



(Star Trek: The Original Series. Season 2, Episode 7. (Oct 27, 1967))

Type Scaffold

```
module Puppet
  newtype(:example_type) do
    def initialize(args) super(args) end

    def finish(args) super end

    newparam(:name) do
      desc "Everybody needs a name!"
      isnamevar
    end

    newparam(:foo) do desc "Foo!" end

    newproperty(:baz) do desc "Property Baz" end

    newparam(:bar) do desc "Parameter Bar" end

    validate do true end

    autorequire(:file) do ['/tmp/foo'] end
  end
end
```

Provider Scaffold

```ruby
Puppet::Type.type(:example_type).provide(:ruby) do
  $example_type_classvars = { :example => true }

  def initialize(*args)
    super(*args)
  end

  def baz
    # Get the state of the 'baz' property from the system
  end

  def baz=(should)
    # Set the system to the value of the 'baz' property in Puppet
  end

  def flush
    # If *any* property is not in sync, this will be called
  end

  def self.post_resource_eval
    # Do this afer *all* resources of this type have been called.
    $example_type_classvars = nil
  end
```

Sample Execution Manifest

```
example_type { 'test':
  foo => 'foo',
  bar => 'bar',
  baz => 'baz'
}
```

Full Execution Walkthrough

The code can be found at http://git.io/jUYx

You can find all sample output at http://git.io/ZPuZNQ

Type

Properties and Parameters

```
newparam(:name) do isnamevar end

newparam(:foo) do
  Puppet.warning("Param :foo -> Starting")
end

newproperty(:baz) do
  Puppet.warning("Property :baz -> Starting")
end

newparam(:bar) do
  Puppet.warning("Param :bar -> Starting")
end
```

Output

```
warning: Param :foo -> Starting
warning: Property :baz -> Starting
warning: Param :bar -> Starting
```

Provider

Globals

```
Puppet::Type.type(:example_type).provide(:ruby) do
  Puppet.warning("Setting Property Class Variables")

  $example_type_classvars = {
    :example => true
  }

 (...)
end
```

Output

```
warning: Setting Property Class Variables
```

WARNING

This may cause memory leaks unless you run in cron mode!

Type

Name Parameter Munge

```
newparam(:name) do
  isnamevar

  munge do |value|
    Puppet.warning("#{value}: In the name parameter.")
    value
  end
end
```

Output

```
warning: test: In the name parameter.
```

Provider

Initializer

```
def initialize(*args)
  super(*args)

  Puppet.warning("Provider Initialization :name= '#{@resource[:name]}'")
  Puppet.warning("Provider Initialization :foo = '#{@resource[:foo]}'")
  Puppet.warning("Provider Initialization :bar = '#{@resource[:bar]}'")
end
```

Output

```
warning: Provider Initialization :name= 'test'
warning: Provider Initialization :foo = ''
warning: Provider Initialization :bar = ''
```

Type

Property | Baz

```
# newparam(:foo) do
# newproperty(:baz) do
# newparam(:bar) do
```

```
newproperty(:baz) do
  validate do |value|
    Puppet.warning("#{resource[:name]}: :baz -> Validating")
    Puppet.warning("#{resource[:name]}: Foo -> '#{resource[:foo]}'")
  end
end
```

Output

```
warning: test: Property :baz -> Validating
warning: test: Foo -> ''
```

Type

Parameter|Foo

```
# newparam(:foo) do
# newproperty(:baz) do
# newparam(:bar) do
```

```
newparam(:foo) do
  validate do |value|
    Puppet.warning("#{resource[:name]}: Param :foo -> Validating")
  end

  munge do |value|
    Puppet.warning("#{resource[:name]}: Param :foo -> Munging")
    Puppet.warning("Where's Param :bar? Bar is '#{resource[:bar]}'")
    value
  end
end
```

Output

```
warning: test: Param :foo -> Validating
warning: test: Param :foo -> Munging
warning: Where's Param :bar? Bar is ''
```

Type

Parameter|Bar

```
# newparam(:foo) do
# newproperty(:baz) do
# newparam(:bar) do
```

```
newparam(:bar) do
  Puppet.warning("Param :bar -> Starting")
  validate do |value|
    Puppet.warning("#{resource[:name]}: Param :bar -> Validating")
  end
  munge do |value|
    Puppet.warning("#{resource[:name]}: Param :bar -> Munging")
    Puppet.warning("Where's Param :foo? Foo is '#{resource[:foo]}'")
    value
  end
end
```

Output

```
warning: test: Param :bar -> Validating
warning: test: Param :bar -> Munging
warning: test: Where's Param :foo? Foo is 'foo' <- Order Matters!
```

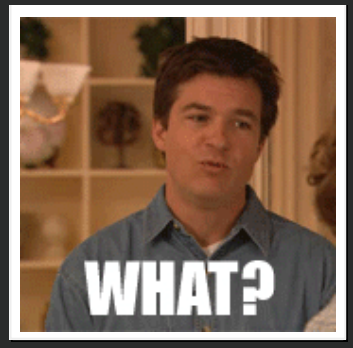
Type Validation

```
validate do
  required_params = [:foo, :bar, :baz]
  Puppet.warning("#{self[:name]}: Validating")
  required_params.each do |param|
    if not self[param] then
      # Note how we show the user *where* the error is.
      raise Puppet::ParseError,"Oh noes! #{self.file}:#{self.line}"
    end
  end
end
```

Output

```
Warning: test: Param :bar -> Validating
```

ARE WE THERE YET?



(Arrested Development, Fox/Netflix)

Type

Initializing

```
def initialize(args)
  super(args)

  Puppet.warning("#{self[:name]}: Type Initializing")

  num_example_types = @catalog.resources.find_all { |r|
    r.is_a?(Puppet::Type.type(:example_type))
  }.count
  Puppet.warning("Ex_order's in the catalog: '#{num_example_types+1}'")
end
```

Output

```
warning: test: Type Initializing
warning: Ex_order's in the catalog: '1'
```

Type

Finish

```
def finish
  Puppet.warning("#{self[:name]}: Type Finishing")

  # Don't forget to call this *at the end*
  super
end
```

Output

```
warning: test: Type Finishing
```

Type

Autorequires

```
autorequire(:file) do
  Puppet.warning("#{self[:name]}: Autorequring")
  ["/tmp/foo"]
end
```

Output

```
warning: test: Autorequring
```

New in 4.0!

Type

Auto*(before | subscribe | notify)*

```
autobefore(:file) do
  Puppet.warning("#{self[:name]}: Autobeforing")
  ["/tmp/foo"]
end

############################################################
# These all work the same way as Autorequire but add
# their respective capabilitites to the mix.
#
# This way you can have completely dynamic workflows
# from within your types.
############################################################
```

Output

```
warning: test: Autobeforing
```

Provider

Baz | Getter

```
Puppet::Type.type(:example_type).provide(:ruby) do
  def baz
    # This is what 'is' ends up being in insync?
    Puppet.warning("#{@resource[:name]}: In getter for :baz")
  end
end
```

Output

```
warning: test: In getter for :baz
```

Type

Baz | Insync?

```
newproperty(:baz) do
  def insync?(is)
    # This is simply what the native provider code would do.
    is == @should

    # Note, you have to use @resource, not resource here since
    # you're not in the property any more, you're in the provider.
    Puppet.warning("#{@resource[:name]}: In 'insync?' for :baz")

    # We're returning false just to see the rest of the components
    # fire off.
    false
  end
end
```

Output

```
warning: test: In 'insync?' for :baz
```

Provider

Baz | Setter

```
Puppet::Type.type(:example_type).provide(:ruby) do
  def baz=(should)
    # Called if insync? is false
    Puppet.warning("#{@resource[:name]}: In setter for :baz")
  end
end
```

Output

```
warning: test: In setter for :baz
notice: /Stage[main]/Main/Ex_order[test]/baz: baz changed
        'test: In getter for :baz' to 'baz'
```

Provider

Flush

```
Puppet::Type.type(:example_type).provide(:ruby) do
  def flush
    # This happens if *any* property was modified.
    Puppet.warning("Time to flush #{@resource[:name]}")
  end
end
```

Output

```
warning: Time to flush test
```

Provider

Post-Resource Eval

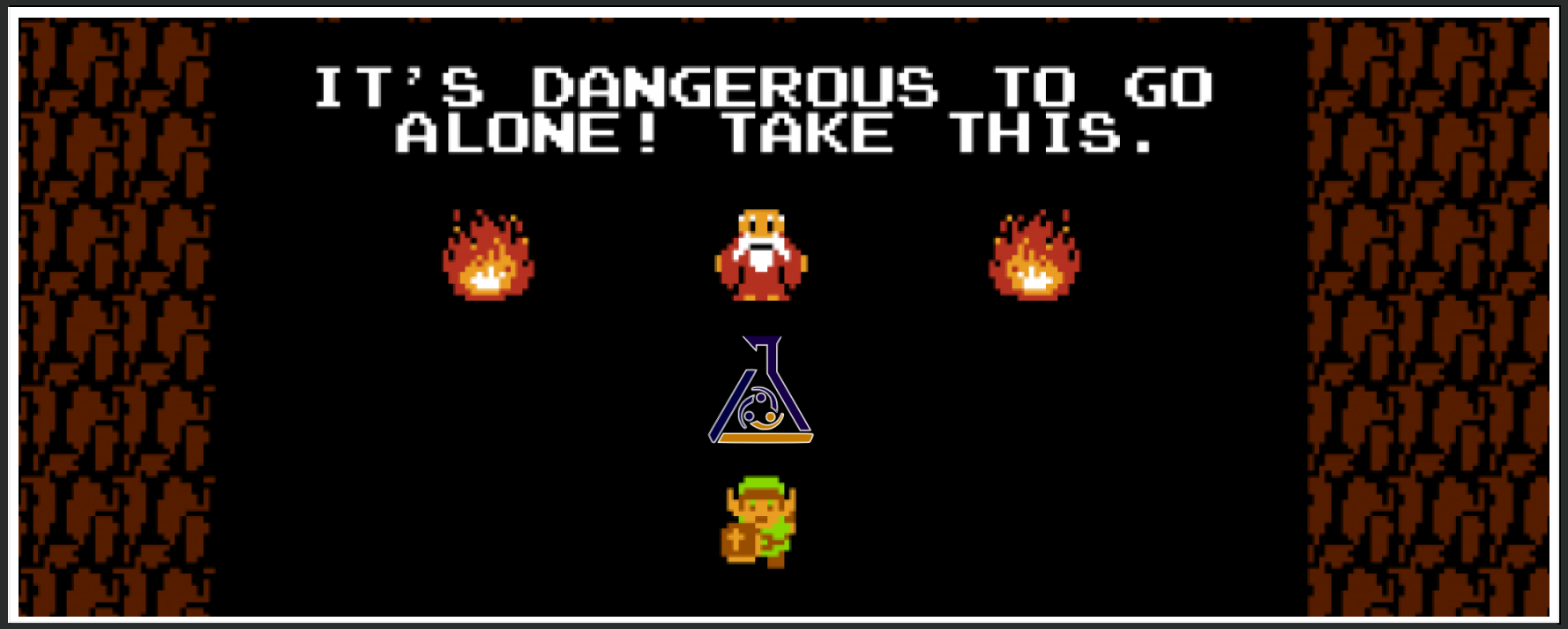
```
# This is new in Puppet 3.4.0 and can be used for
# cleaning up any resource persistence since it happens after
# *all* resources of this type have been evaluated.

def self.post_resource_eval
  # Only YOU can prevent memory leaks!
  $example_type_classvars = nil

  Puppet.warning("FINIS!!!!")
end
```

Output

```
warning: FINIS!!!!
```



(The Legend of Zelda, Nintendo)

Resources

- Type and Provider Execution Walkthrough
  - The original article upon which this presentation is based
- Puppet Labs' Custom Type Documentation
  - The official documentation on custom types.
- Puppet Types and Providers Book
  - The Puppet Types and Providers book by Dan Bode and Nan Liu
- Presentation Source
  - The source code for this presentation

Presentation Information

This presentation was made possible by:

- Reveal.js by Hakim El Hattab
- Reveal.js Modificationsby José Manuel Ciges Regueiro