Mockfight!

FlexMock vs. Mocha

Patrick Reagan
patrick@viget.com
Director, Application Development

Viget Labs
we build web business

July 18, 2007

© Copyright 2007 Viget Labs, LLC – www.viget.com



What are Mocks?

"Mock objects are simulated objects that mimic the behavior of real objects in controlled ways"
- Wikipedia

© Copyright 2007 Viget Labs, LLC – www.viget.com

When to Mock?

Use mocks when you want to:

- Control the behavior of internal method calls
- Remove dependence on external systems
- Improve test performance

© Copyright 2007 Viget Labs, LLC – www.viget.com

Control Behavior

What does this return?

```
class Die
  def roll
    rand(6) + 1
  end
end
```

```
>> die = Die.new
=> #<Die:0x358680>
>> die.roll
=> 6
>> die.roll
=> 3
```

Need to maintain consistency between calls

© Copyright 2007 Viget Labs, LLC – www.viget.com

Control Behavior

How do we test?

```
class DieTest < Test::Unit::TestCase
  def test_roll_should_return_rand_plus_one
    die = Die.new
    die.stubs(:rand).once.returns(5)
    assert_equal 6, die.roll
  end
end
```

Magic happens here

Stub the call to `Die#rand` (using Mocha)

© Copyright 2007 Viget Labs, LLC – www.viget.com

Remove Dependencies

Example: retrieve response data via HTTP

```
def self.exists?(url)
  uri = URI.parse(url)
  response = Net::HTTP.new(uri.host).head(uri.path)
  response ? true : false
end
```

To test this, we need to mock:

- Call to `URI#parse`
- Calls to `Net::HTTP#new` and `Net::HTTP#head`

© Copyright 2007 Viget Labs, LLC – www.viget.com

Mockfight!

Expanded example:

- Create class method to check for resource
- Handle errors and redirects
- Iteratively develop functionality while testing
- Examine the differences between libraries

© Copyright 2007 Viget Labs, LLC – www.viget.com

Round 1

Test scenario when `URI#parse` raises exception

```
class HTTPResource
  def self.exists?(url)
    begin
      URI.parse(url)
    rescue URI::InvalidURIError
      exists = false
    end
    exists
  end
end
```


© Copyright 2007 Viget Labs, LLC – www.viget.com

Round 1: Results

FlexMock

```
def test_exists_with_URI_parse_exception_should_return_false
  flexmock(URI).should_receive(:parse).once.and_raise(URI::InvalidURIError)
  assert_equal false, HTTPResource.exists?('http://www.example.com')
end
```

Mocha

```
def test_exists_with_URI_parse_exception_should_return_false
  URI.expects(:parse).once.raises(URI::InvalidURIError)
  assert_equal false, HTTPResource.exists?('http://example.com')
end
```


© Copyright 2007 Viget Labs, LLC – www.viget.com

Round 2

Check for a valid response

```
class HTTPResource
  def self.exists?(url)
    begin
      uri = URI.parse(url)
      response = Net::HTTP.new(uri.host).head(uri.path)
      exists = (response.code == '200')
    rescue URI::InvalidURIError
      exists = false
    end
    exists
  end
end
```


© Copyright 2007 Viget Labs, LLC – www.viget.com

Round 2: Results

FlexMock

```
def test_exists_with_valid_resource_should_return_true
  flexmock(Net::HTTP).should_receive(:new).once.and_return {
    flexmock(:head => flexmock(:code => '200'))
  }
  assert_equal true, HTTPResource.exists?(@url)
end
```

Mocha

```
def test_exists_with_valid_resource_should_return_true
  Net::HTTP.expects(:new).once.returns(mock(:head => mock(:code => '200')))
  assert_equal true, HTTPResource.exists?(@url)
end
```


© Copyright 2007 Viget Labs, LLC – www.viget.com

Round 3

Check for response type instead of status code

```
class HTTPResource
  def self.exists?(url)
    begin
      uri = URI.parse(url)
      response = Net::HTTP.new(uri.host).head(uri.path)
      exists = response.is_a?(Net::HTTPSuccess)
    rescue URI::InvalidURIError
      exists = false
    end
    exists
  end
end
```

* Our previous tests for response status code are now invalid

© Copyright 2007 Viget Labs, LLC – www.viget.com

Round 3: Results

FlexMock

```
def test_exists_with_valid_resource_should_return_true
  flexmock(Net::HTTP).should_receive(:new).once.and_return {
    flexmock(:head => Net::HTTPSuccess.new('1.2', '200', 'OK'))
  }
  assert_equal true, HTTPResource.exists?(@url)
end
```

Mocha

```
def test_exists_with_valid_resource_should_return_true
  Net::HTTP.expects(:new).once.returns(
    mock(:head => Net::HTTPSuccess.new('1.2', '200', 'OK'))
  )
  assert_equal true, HTTPResource.exists?(@url)
end
```


© Copyright 2007 Viget Labs, LLC – www.viget.com

Round 4

Follow an HTTP redirect response

```
class HTTPResource

  @redirect_count = 0

  def self.exists?(url)
    begin
      uri = URI.parse(url)
      response = Net::HTTP.new(uri.host).head(uri.path)
      case response
      when Net::HTTPSuccess
        exists = true
      when Net::HTTPRedirection
        @redirect_count+= 1
        exists = exists?(response['location'])
      else
        exists = false
      end
    rescue URI::InvalidURIError
      exists = false
    end
    exists
  end
end
```


© Copyright 2007 Viget Labs, LLC – www.viget.com

Round 4: Results

FlexMock

```
def test_exists_with_redirect_should_increment_redirect_count
  redirect = Net::HTTPRedirection.new('1.2', '301', 'Moved Permanently')
  flexmock(redirect).should_receive(:[]).with(String).and_return(
    'http://www.example.com/redirected'
  )

  success  = Net::HTTPSuccess.new('1.2', '200', 'OK')
  flexmock(Net::HTTP).should_receive(:new).times(2).and_return(
    flexmock(:head => redirect), flexmock(:head => success)
  )

  HTTPResource.exists?(@url)
  assert_equal 1, HTTPResource.instance_variable_get(:@redirect_count)
end
```


© Copyright 2007 Viget Labs, LLC – www.viget.com

Round 4: Results

Mocha

```
def test_exists_with_redirect_should_increment_redirect_count
  redirect = Net::HTTPRedirection.new('1.2', '301', 'Moved Permanently')
  success  = Net::HTTPSuccess.new('1.2', '200', 'OK')
  redirect.stubs(:[]).with('location').returns('http://www.example.com/redirected')
  Net::HTTP.expects(:new).times(2).returns(
    mock(:head => redirect), mock(:head => success)
  )

  HTTPResource.exists?(@url)
  assert_equal 1, HTTPResource.instance_variable_get(:@redirect_count)
end
```


© Copyright 2007 Viget Labs, LLC – www.viget.com

Round 5

Handling maximum number of redirects

```
class HTTPResource

  class MaximumRedirectError < StandardError; end

  @redirect_count = 0
  MAXIMUM_REDIRECTS = 5

  def self.exists?(url)
    begin
      uri = URI.parse(url)
      response = Net::HTTP.new(uri.host).head(uri.path)
      case response
      when Net::HTTPSuccess
        exists = true
      when Net::HTTPRedirection
        @redirect_count+= 1
        raise MaximumRedirectError if @redirect_count > MAXIMUM_REDIRECTS
        exists = exists?(response['location'])
      else
        exists = false
      end
    rescue URI::InvalidURIError, MaximumRedirectError
      exists = false
    end
    exists
  end
end
```


© Copyright 2007 Viget Labs, LLC – www.viget.com

Round 5: Results

FlexMock

```
def test_exists_with_maximum_redirects_should_return_false
  redirect = Net::HTTPRedirection.new('1.2', '301', 'Moved Permanently')
  flexmock(redirect).should_receive(:[]).with(String).and_return(
    'http://www.example.com/redirected'
  )
  flexmock(Net::HTTP).should_receive(:new).once.and_return(flexmock(:head => redirect))

  HTTPResource.instance_variable_set(:@redirect_count, HTTPResource::MAXIMUM_REDIRECTS)
  assert_equal false, HTTPResource.exists?(@url)
end
```


© Copyright 2007 Viget Labs, LLC – www.viget.com

Round 5: Results

Mocha

```
def test_exists_with_maximum_redirects_should_return_false
  redirect = Net::HTTPRedirection.new('1.2', '301', 'Moved Permanently')
  redirect.stubs(:[]).with('location').returns('http://www.example.com/redirected')
  Net::HTTP.expects(:new).once.returns(mock(:head => redirect))

  HTTPResource.instance_variable_set(:@redirect_count, HTTPResource::MAXIMUM_REDIRECTS)
  assert_equal false, HTTPResource.exists?(@url)
end
```


© Copyright 2007 Viget Labs, LLC – www.viget.com

Bonus Round

Mocking `ActiveRecord#find` in controllers

```
class AccountsController < ApplicationController
  def show
    @account = Account.find(params[:id])
    render :nothing => true
  end
end
```


© Copyright 2007 Viget Labs, LLC – www.viget.com

Bonus Round: Results

FlexMock

```
def test_show_should_set_expected_account_instance
  account = Account.new(:username => 'preagan')
  flexmock(Account).should_receive(:find).with(String).and_return(account)
  get :show, :id => 1
  assert_equal account.username, assigns(:account).username
end
```

Mocha

```
def test_show_should_set_expected_account_instance
  account = Account.new(:username => 'preagan')
  Account.expects(:find).with("1").returns(account)
  get :show, :id => 1
  assert_equal account.username, assigns(:account).username
end
```


© Copyright 2007 Viget Labs, LLC – www.viget.com

But Wait, There's More

Mocha has parameter matchers (in SVN)

You can write code like this:

```
def test_show_should_set_expected_account_instance
  account = Account.new(:username => 'preagan')
  Account.expects(:find).with(kind_of(String)).returns(account)
  get :show, :id => 1
  assert_equal account.username, assigns(:account).username
end
```


© Copyright 2007 Viget Labs, LLC – www.viget.com

Conclusion

- Newest versions of gems contain similar features
- FlexMock can accept blocks for further mocking
- Personal preference of syntax is the deciding factor

© Copyright 2007 Viget Labs, LLC – www.viget.com

It's a Tie!



Ultimately, two great libraries - just pick one

© Copyright 2007 Viget Labs, LLC – www.viget.com

Questions?



© Copyright 2007 Viget Labs, LLC – www.viget.com

Additional Information

FlexMock (Jim Weirich)

http://www.onestepback.org/software/flexmock

Mocha (James Mead)

http://mocha.rubyforge.org

http://www.slideshare.net/viget

© Copyright 2007 Viget Labs, LLC – www.viget.com