

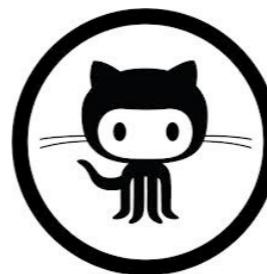
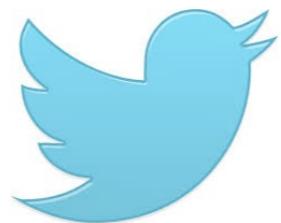


# Migrate to JRuby

## What We Learned

# ian yang

@doitian



# Agenda

- Background
- How to Migrate
- What We Learned

# Background

# Breadcrumb Payments

# The Codebase

- 4 Ruby on Rails applications
- Ruby 1.9.2 and Rails 3.2.1

# Why JRuby?

- Multithreading
- Memory Leak
- Easy integration of libraries on JVM

# How to Migrate

# Choose a Deploy Strategy

---

	Warbler	Trinidad	TorqueBox
Archive File	✓	✓	✓
Capistrano		✓	✓
Scheduler		✓	✓
Background Jobs		✓	✓
Clustering			✓

---

---

	Warbler	Trinidad	TorqueBox
Archive File	✓	✓	✓
Capistrano		✓	✓
Scheduler		✓	✓
Background Jobs		✓	✓
Clustering			✓

---

# JRuby-Lint

---

<https://github.com/jruby/jruby-lint>

# What We Learned

# Gems Alternatives

# Wiki: C Extension Alternatives

---

[https://github.com/jruby/jruby/wiki/C-Extension-  
Alternatives](https://github.com/jruby/jruby/wiki/C-Extension-Alternatives)

**Lock Compatible  
Version**

```
gem 'rubyzip', '<1.0.0'
```

**\*nix to JVM**

- No Kernel#fork
- Cannot trap all signals

Kernel#fork → Thread.new

# Thread-safety

# Avoid Sharing Mutable State Between Threads

---

**Global Variable**



**Class Variable**



**Class Instance Variable**

---



**Lazy Initialization →  
Preload**

```
class Cvv
  def self.redis
    @@redis ||= Redis.new(...)
  end
end
```

X

```
class Cvv
  def self.redis
    @@redis
  end
  def self.redis=(r)
    @@redis = r
  end
end
```



# config/initializers/cvv.rb  
Cvv.redis = Redis.new(...)

# Thread Local Storage

```
class Processor   
  class_attribute :skip  
  
  def execute  
    do_something unless self.class.skip  
  end  
end
```

```
class Processor
  def self.skip
    Thread.current['Processor.skip']
  end
  def self.skip=(s)
    Thread.current['Processor.skip'] = s
  end
end
```



# Atomic Variable

```
class Airlock
  class_variable :enabled
end
```

X

```
# gem 'atomic' ✓
class Airlock
  @@enabled = Atomic.new(false)
  def self.enabled
    @@enabled.value
  end
  def self.enabled=(e)
    @@enabled.update { e }
  end
end
```

# Locks

```
require 'thread'  
mutex = Mutex.new  
  
Thread.new do  
  mutex.synchronize do  
    ...  
  end  
end
```

# Reference: Concurrency in JRuby

---

<https://github.com/jruby/jruby/wiki/Concurrency-in-jruby>

# Reference: Ruby Mutithreading

---

[http://www.tutorialspoint.com/ruby/  
ruby\\_multithreading.htm](http://www.tutorialspoint.com/ruby/ruby_multithreading.htm)

# Resque

- No Fork
- Jobs run in the same process
- Free the resources

# OpenSSL

# Full OpenSSL Support

```
gem 'jruby-openssl'
```

# Different Cryptography Implementations

---

<https://github.com/jruby/jruby/issues/931>



jruby / jruby

Watch ▾ 124

Star 1,528

Fork 409

Browse Issues

Milestones

New Issue

[← Back to issue list](#)

Issue #931



andrenpaes opened this issue 3 months ago

## Cipher "des-ede" working differently between JRuby and MRI

No one is assigned

No milestone

[Open](#)

1 comment

Labels

[openssl](#)

[stdlib](#)

I'm facing a problem trying to decrypt some data using JRuby. I'm using the 'des-ede' cipher with no padding. The code works fine in MRI.

Here's an example:

```
require 'openssl'  
require 'base64'  
  
str = 'helloooo'  
key = "WVqcvjGqaD7XBVB1XYbJYw==\n"
```

# Cryptography Adapter

- Use OpenSSL in CRuby
- Use JCE directly in JRuby  
(`java_import`)

# Tuning

**TorqueBox + JBoss**

# **Connection Pool Size**

# Thread Pool Size

# JVM Memory

# How To?

- Benchmark
- Monitoring

# References



## Deploying with JRuby

Deliver Scalable Web Apps  
using the JVM



Joe Kutner  
Edited by Brian E. Hogan

The Facets of Ruby Series

# Deploying with JRuby

# JRuby Wiki

---

<https://github.com/jruby/jruby/wiki>

