

Enterprise JRuby



Except where otherwise noted, the content of this presentation is licensed under the Creative Commons Attribution-Share Alike 3.0 United States License (<http://creativecommons.org/licenses/by-sa/3.0/us/>).

Introductions

Me

- Charles Oliver Nutter
- Java dev since 1997
- Rubyist, JRuby dev since 2004
- Full time JRuby dev since 2006
 - Engine Yard since Monday!
- headius@headius.com, headius on Twitter
- blog.headius.com

JRuby

- Current version: 1.3.1
- Ruby 1.8.6 compatible (give or take)
- Some Ruby 1.9 support
- Solid performance (\approx Ruby 1.9)
- Real native threads
- Runs Ruby/Rails great!

The Present

Rails

- Generally, “just works”
- Almost all Rails core tests pass
- ActiveRecord is a challenge...
- Deployment is “pretty good”
- Integration with Java needs work

Demo: Running Typo

- Already done:
 - Unpack (gem has native dependencies)
 - Configure database.yml
 - Prepare database (create, migrate)
- Ready to deploy!

Java Integration

- Generally as easy as calling Ruby
- Massive number of libraries
- No porting, no building
- Makes working with Java actually **fun**

Ruby 1.9 Support

- All in one with --1.9 flag
- Maybe 80% of 1.9.1
 - 1.9.2 adds a bunch more
 - Interested in helping?
- 1.9 support desired?

Android

Android

- Open mobile platform
- Linux-based OS
- Java-based dev env
 - Java = JRuby = Ruby for mobiles!



RUBOTO

The Future

What's Missing?

- Finish 1.9 support
- More performance (neverending!)
- Better Java integration
- Adapting Rails to the Java platform
- Rubifying popular Java libraries
 - Hibernate, JPA, Maven, Ant, JAXP, EE, ...

Performance

- JSR 292 “invokedynamic”
 - Fast dynamic calls for the JVM
- New JRuby optimizing compiler
- Continuing optz for core classes
- Faster Java integration

Java Integration

- Improving existing call layers
 - Perf, memory, GC, coercion, wrapping
- Offline generation of Java classes
 - For “meta” APIs using annotations
- Runtime generation of Java classes
 - Many APIs just want a `java.lang.Class` obj

ruby2java

- Generates a Java .class
- Uses **runtime** definition of class
- Embeds source directly in .class

“become_java”

(working title)

- Generates a Java class in-memory
- Uses runtime definition of class
- Ruby object is instance of the Java class
- Annotations, signatures, whatever

Ruboto

- JRuby, tailored for Android
 - Reduced package size
 - Reduced memory footprint
 - Optimized for performance

Rubifying Java Libraries

- Persistence
 - Hibernate obviously; JPA to follow
- Build tooling
 - Ant and Maven really need help
- Services and components
 - Ruby everywhere, even in EE servers(?!?)

Hibernate

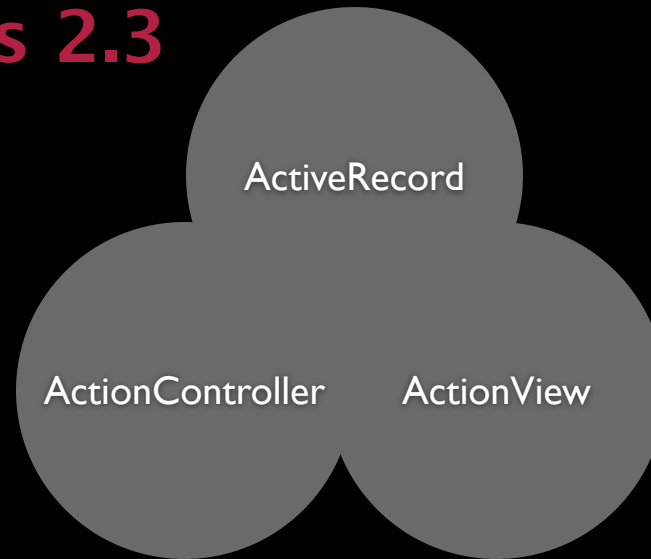
- Most extensive ORM package around
- Wide DB support
- Transactions, procedures, prepared stmts
- Caching, pooling, object identity
- DB-agnostic query language
- De-facto standard in Java world

Jibernate!

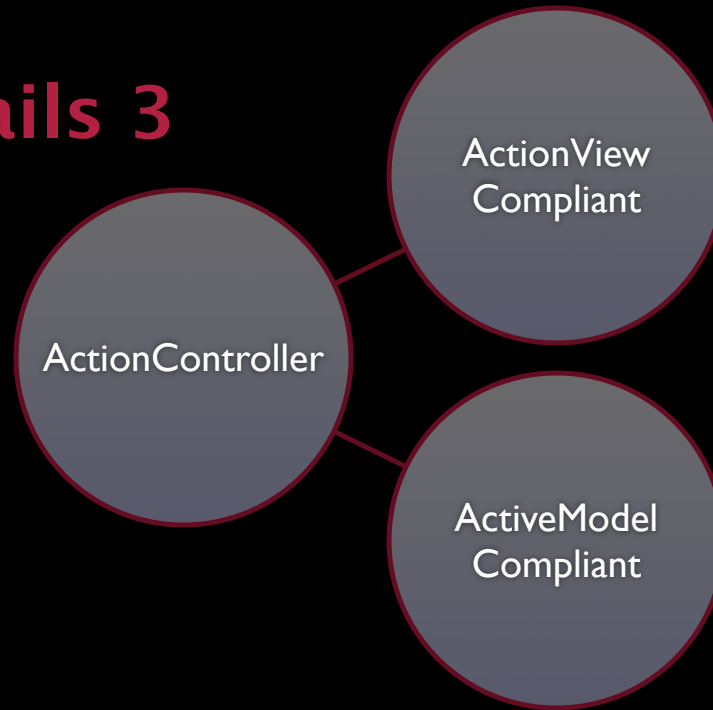
- Proof-of-concept Hibernate support
- Normal Ruby classes
 - ...turned into POJOs!
- Basically the beginner Hibernate tutorial
- Groundwork for the future
 - I build plumbing, you build porcelain

Rails 3?

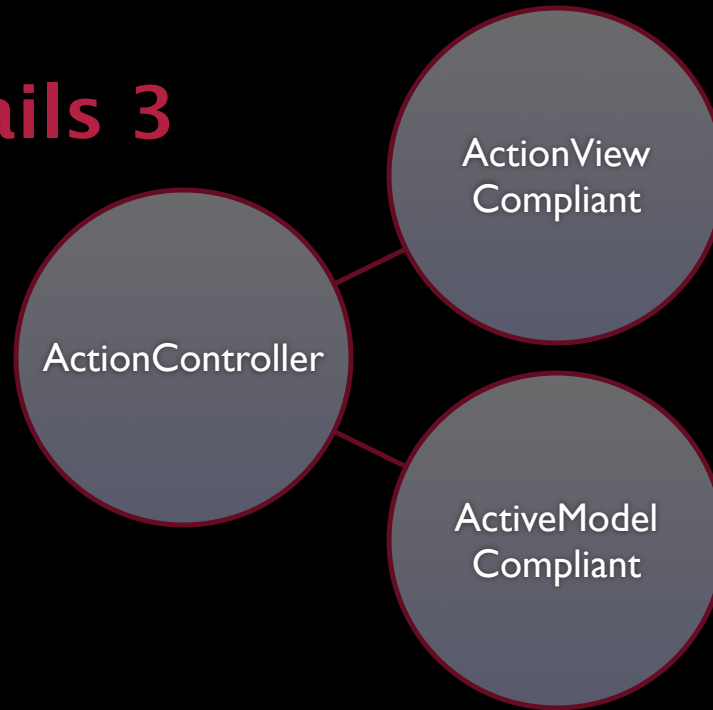
Rails 2.3




Rails 3



Rails 3



Rails 3



ActiveModel
Compliant

```
new_record?  
valid?  
errors  
model_name
```

ActiveModel

```
class Person
  include ActiveSupport::Validations

  validates_presence_of :name

  attr_accessor :name
  def initialize(name = nil)
    @name = name
  end
end

person1 = Person.new
person1.valid?      #=> false
person1.errors     #=> {:name => ["cannot be blank"]}

person2 = Person.new("matz")
person2.valid?     #=> true
```

API

```
class Person
  include ActiveRecord::Validations
  validates_presence_of :name

  attr_accessor :name
  def initialize(name = nil)
    @name = name
  end

  def persist
    @persisted = true
  end

  def new_record?
    @persisted
  end

  def to_model() self end
end
```

```
class Person
  include ActiveRecord::Validations
  extend ActiveRecord::Naming

  attr_accessor :name
  def initialize(attributes)
    @name = attributes[:name]
  end

  def persist
    @persisted = true
  end

  def new_record?
    @persisted
  end

  def to_model() self end
end
```

```
class PeopleController < ActionController::Base
  def new
    @person = Person.new(params[:person])
  end

  def create
    @person = Person.new(params[:person])
    if @person.persist
      redirect_to person_path(@person)
    else
      render :action => :new
    end
  end
end
```

```
<%= error_messages_for @person %>
<% form_for @person do |f| %>
  <%= f.text_field :name %>
  <%= f.submit_button "Create" %>
<% end %>
```

What's Next?

- The future of Ruby depends on JRuby
 - Java community is gigantic
 - Big Java shops will not use CRuby
- Java escapees are going back
 - Groovy, Clojure, Scala
 - Libraries, ecosystem

How Can You Help?

- Use JRuby every chance you get
- Help us improve JRuby (1.9 help, please!)
- Start evangelizing at *Java* conferences
- Study Groovy, Scala, and help us compete
- Study Java libraries and help us Rubify

More Info

- www.jruby.org
- wiki.jruby.org
- www.kenai.com/projects/jruby
- blog.headius.com, headius@headius.com
- github.com/headius/railsunder