

The plan

- What's JRuby?
- The Ruby language
- Why JRuby?
- How to use it
- JRuby on Rails
- Demos, demos, demos



Who am I?

Who are you?

What's JRuby?

It's just Ruby

but on the JVM

Any questions?

Demo time

font_size.rb

```
class JLabel
  import java.awt.Font
  def font_size=(size)
    self.font = Font.new("Serif", Font::PLAIN, size)
  end
end
```

What it ain't

- A JVM specific language variant
- Incompatible
- A slow Ruby implementation

History

- Ruby started in 1993
- Created by Yukihiro “matz” Matsumoto
 - to maximize programmer enjoyment
- JRuby Started in 2002
- “Resurrected” ~ 2005
- Main guys employed by SUN

Performance

- JRuby 1.0 focused on compatibility
 - 2x slower than Ruby 1.8 (MRI)
- JRuby 1.1 beta1 was 2x faster
- JRuby 1.1 RC2 was 5x faster

Ruby crash course by example

Hello Ruby

```
class Hello < Object

  attr_accessor :greeting

  def initialize(greeting)
    @greeting = greeting
  end

end

hello = Hello.new("Hello Las Vegas!")
puts hello.greeting
hello.greeting = "and to TSSJS"
puts hello.greeting
```

Hello Metaprogramming

```
attr_accessor :greeting
```

is equivalent to

```
def greeting  
  @greeting  
end
```

```
def greeting=(the_greeting)  
  @greeting = the_greeting  
end
```

Methods

```
class MethodMan
  def +(addend)
    2 + addend
  end

  def [](index)
    "bet you don't expect this."
  end
end
```

```
end
```

```
method_man = MethodMan.new
method_man2 = MethodMan.new
```

```
puts method_man + 98
puts method_man[7]
```

```
def method_man.my_method
  puts "you invoked my method"
end
```

```
method_man.my_method
method_man2.my_method
```

Blocks

```
class FunWithBlocks
  attr_accessor :string

  def initialize
    @string = "something"
  end

  def do_something_with_block
    puts "Did something"
    yield
    puts "Did something else"
  end

  def give_me_a_string
    yield @string
  end
end

fun_with_blocks = FunWithBlocks.new

fun_with_blocks.do_something_with_block do
  puts "Called my block"
end

fun_with_blocks.give_me_a_string { |string| string.upcase! }
puts fun_with_blocks.string
```

Duck typing

```
class QuackIt
  def make_it_quack(quacker)
    puts quacker.quack
  end
end

class Duck
  def quack
    "quack"
  end
end

class QuackingDog
  def quack
    "Ruff... err.. I mean, quack"
  end
end

quackit = QuackIt.new
quackit.make_it_quack(Duck.new)
quackit.make_it_quack(QuackingDog.new)
```

Class methods

```
class Wierdo
  puts "Self is #{self.inspect}"

  def self.new
    "Ha. I don't even return a wierdo"
  end
end

puts Wierdo.new
```

Modules

```
# A collection of objects
class MyProducts
  # Enumerable provides iteration methods
  include Enumerable

  # define an 'each' method that iterates
  def each
    # yield each element in turn
  end
end

list = MyProducts.new
list.select {|item| item.price > 5.00}
list.sort {|a, b| a.name <=> b.name}
list.max
```

Metaprogramming, let me count the ways

- open classes
- method_missing
- hooks
- eval

Where does this lead?

- Things you don't need in Ruby:
 - Dependency Injection
 - AOP
 - Bytecode enhancement
 - Loads o' design patterns

Why JRuby?

The real thing or the imitation

You can't confuse it with Java

Flexible syntax = Better DSLs

Cross Platform (but what's a platform?)

The Ruby ecosystem

How to use it

Getting started

- Download it
- Extract it
- Set JRUBY_HOME
- Add it to your path
- Enjoy!

Ruby from Java

- Java6 ScriptEngine (JSR-223)
- BSF for pre-Java6
- Or JRuby API

Java from Ruby

- include Java
- import your classes
- Some auto-rubification
- Easy to take it further

But what about Rails?

Rails in a nutshell

- Full stack MVC framework for web apps
- Convention over configuration
- Validations
- Migrations
- Plugins

Rails demo

Rails meets J2EE

- ActiveRecord-JDBC
 - Connects AR to JDBC
 - Can use JNDI connection pools
 - DBs still need some luvin' beyond just a driver
 - MySQL, Postgres, Oracle, DB2, supported (for starters)

Rails meets J2EE

- Goldspike
 - Servlet which runs the JRuby interpreter
 - Pools of JRuby runtimes on the ServletContext
- Warbler
 - Easy packaging of a Rails app as a war
 - Bundles everything you need
- Glassfish gem

Rails meets J2EE demo

But does anybody use it?

- Mingle
- Oracle Mix
- See wiki

But there's more to JRuby than Rails...

- Merb
- Datamapper
- Camping
- A growing horde of web frameworks
 - sound familiar?

JRuby Swings

- Ruby lacks good GUI support
- Swing is hard to code
- Smells like opportunity...

JRuby + Swing =

- Swing, Rubified
- Profligacy
 - wiki like layout
- MonkeyBars
 - Tool friendly

Gems

- Package manager for Ruby
- Using it is easy
 - `gem install rails`
 - dependencies handled for you
 - Think maven w/out all the XML ;)

Rake

- Build system for Ruby
- Simple, expressive, powerful
- Buildr, Raven both bring it to Java projects

ActiveHibernate

```
# define a model (or you can use existing)
class Project
  include Hibernate
  with_table_name "PROJECTS" #optional

  #column name is optional
  primary_key_accessor :id, :long, :PROJECT_ID
  hatr_accessor :name, :string
  hatr_accessor :complexity, :double
end

# connect
ActiveHibernate.establish_connection(DB_CONFIG)

# create
project = Project.new(:name => "JRuby", :complexity => 10)
project.save
project_id = project.id

# query
all_projects = Project.find(:all)
jruby_project = Project.find(project_id)

# update
jruby_project.complexity = 37
jruby_project.save
```

Links

- <http://www.jruby.org/>
 - Wiki has lots of good stuff
- <http://dist.codehaus.org/jruby/talks/>
 - Final slides will be posted there
- <http://monkeybars.rubyforge.org/>
- <http://www.netbeans.org/>
- Email me at
me@christophernelsonconsulting.com

Build automation

- Nobody enjoys maintaining Ant scripts
 - Ruby is designed to be enjoyable
- Maven can get over-complicated
 - Ruby helps simplify things
- Declarative build tools are inherently limiting
 - Ruby solutions allow normal Ruby code

AntBuilder

Buildr