Ruby for Java Programmers

Mike Bowler
President, Gargoyle Software Inc.
Why learn another language?
History

- Originated in Japan in 1993
  - Created by Yukihiro “Matz” Matsumoto
- First English language book published in 2000
  - Programming Ruby by Dave Thomas and Andy Hunt (the Pragmatic Programmers)
- The catalyst, Ruby on Rails, appeared in 2004
The Ruby Language

- Python
- Smalltalk
- Perl
- Lisp

© Copyright 2006, Gargoyle Software Inc.
class Sample
  def print
    for i in 1..3
      puts i
    end
  end
end

sample = Sample.new
sample.print
Defining Strong/Weak Typing

- **Strong typing**
  - Objects are of a specific type and will not be converted automatically
  - Java: “4”/2 results in a compile error

- **Weak typing**
  - Objects can be converted under the covers at any time
  - Perl: ‘4’/2 => 2

- **Ruby is strongly typed**
Early/Late Binding

- Early binding
  - All method invocations must be defined at compile time
  - Java: `foo.getUser()`

- Late binding
  - The runtime does not check that a given method exists until an attempt to invoke it
  - Smalltalk: `foo getuser`

- Ruby uses late binding
The Parts of Ruby

- Virtual machine with garbage collection
- Standard libraries
  - Core API
    - Base classes and modules, such as String, Array, Symbol, etc.
  - Standard API
    - Additional libraries such as CGI, OpenURI, and REXML
Inheritance

```
garfield = Cat.new
```

Diagram:
- Object
- Animal
- Class
- Module
- Garfield

Garfield inherits from Class, which is an instance of Cat.
class ZebraCage < Cage
  attr_accessor :capacity
  @@allCages = Array.new

  def initialize maximumZebraCount
    @capacity = maximumZebraCount
    @@allCages << self
  end

private
  def clean_cage
    # do some stuff here
  end
end

cage = ZebraCage.new 10
puts cage.capacity
do_something if list.empty?

name.strip
name.strip!
Syntactic Sugar: Array

```ruby
list = Array.new
list << 'one' << 'two' << 'three'
list.push('one', 'two', 'three')
list = ['one', 'two', 'three']
list = %w(one two three)
```
Syntactic Sugar: Hash

```ruby
hash = Hash.new
hash[:a] = 4
hash[:b] = 5

hash = {:a => 4, :b => 5}
```
Syntactic Sugar: Hash

def dump number, hash
    puts hash.keys.sort { |a,b|
        a.to_s <=> b.to_s
    }
end

dump 30, {:a => 5, :b => 3}
dump 100, :a => 5, :b => 3
line = $1 if line =~ /\s*(.+)\s*$/

regex = Regexp.new '^
\s*(.+)
\s*$'
matchData = regex.match line
line = matchData[1] unless matchData.nil?
## Nil and Null

<table>
<thead>
<tr>
<th>Java’s null</th>
<th>Ruby’s nil</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absence of an object</td>
<td>An instance of NilClass</td>
</tr>
<tr>
<td>if( a != null ) {...}</td>
<td>unless a.nil? {...}</td>
</tr>
<tr>
<td>null.toString() -&gt; NPE</td>
<td>nil.to_s -&gt; &quot;&quot;</td>
</tr>
<tr>
<td>null.getUser() -&gt;</td>
<td>nil.get_user -&gt;</td>
</tr>
<tr>
<td>Exception in thread &quot;main&quot;</td>
<td>NoMethodError: undefined method</td>
</tr>
<tr>
<td>java.lang.NullPointerException</td>
<td>‘get_user’ for nil:NilClass</td>
</tr>
</tbody>
</table>
Message != Method

Customer → Person → Object

first_name()
What if there isn’t a method for the specified message?

Message

search()

Customer

method_missing()

Person

Object

Method
method_missing Example from ActiveRecord

user = Users.find_by_name(name)

user = Users.find(:first, :conditions => ["name = ?", name])
Creating Proxy Objects

- Mock object for testing
- Proxy object to allow distributed objects across machines
- Wrapper to record usage of a given object
Implementing a Proxy

class Proxy
  def method_missing name, *args, &proc
    puts name, args
  end
end

Proxy.new.foo_bar 'a'

foo_bar
a
Implementing a Proxy

class Proxy
  instance_methods.each do |method|
    undef_method method unless method =~ /^__/ 
  end

  def method_missing name, *args, &proc
    puts name, args
  end
end

Proxy.new.to_s
Duck Typing

- Duck typing implies that an object is interchangeable with any other object that implements the same interface, regardless of whether the objects have a related inheritance hierarchy. – Wikipedia

- "If it walks like a duck and quacks like a duck, it must be a duck." – Pragmatic Dave Thomas
int a = Integer.MAX_VALUE;
System.out.println("a="+a);
System.out.println("a+1="+(a+1));

a=2147483647
a+1=-2147483648
Numeric Overflow: Ruby

```ruby
number = 1000
1.upto(4) do
  puts "#{number.class} #{number}"
  number = number * number
end

Fixnum 1000
Fixnum 1000000
Bignum 1000000000000
Bignum 1000000000000000000000000
```
Closures

- An object that is a block of code
- Can be assigned to variables and passed to methods
- Has access to the scoped variables at the point it is created

```ruby
number = 1000
1.upto(4) do
  puts "#{number.class} #{number}"
  number = number * number
end
```
Closures

def foo &proc
    proc.call 3
    proc.call 2
    proc.call 1
end

a = 2
foo { |number|
    puts number*a
}

foo do |number|
    puts number*a
end

Results
6
4
2
Closures – Cleaner Syntax

```ruby
def foo &proc
  proc.call 3
  proc.call 2
  proc.call 1
end

def foo
  yield 3
  yield 2
  yield 1
end
```
Closures

```ruby
file = File.new(fileName, 'w')
begin
  file.puts 'some content'
rescue
  file.close
end

File.open(fileName, 'w') do |file|
  file.puts 'some content'
end
```
// Read the lines and split them into columns
List<String[]> lines = new ArrayList<String[]>()
BufferedReader reader = null;
try {
    reader = new BufferedReader(new FileReader("people.txt"));
    String line = reader.readLine();
    while (line != null) {
        lines.add(line.split("\t"));
    }
} finally {
    if (reader != null) {
        reader.close();
    }
}

// then sort
Collections.sort(lines, new Comparator<String[]>() {
    public int compare(String[] one, String[] two) {
        return one[1].compareTo(two[1]);
    }
});

// then write them back out
BufferedWriter writer = null;
try {
    writer = new BufferedWriter(new FileWriter("people.txt"));
    for (String[] strings : lines) {
        StringBuilder builder = new StringBuilder();
        for (int i = 0; i < strings.length; i++) {
            if (i != 0) {
                builder.append("\t");
            }
            builder.append(strings[i]);
        }
        builder.append("\n");
        writer.write(builder.toString());
    }
} finally {
    if (writer != null) {
        writer.close();
    }
}

# Load the data
lines = Array.new
IO.foreach('people.txt') do |line|
    lines << line.split
end

# Sort and write it back out
File.open('people.txt', 'w') do |file|
    lines.sort { |a,b| a[1] <=> b[1]}.each do |array|
        puts array.join("\t")
    end
end
# Load the data
lines = Array.new
IO.foreach('people.txt') do |line|
  lines << line.split
end

# Sort and write it back out
File.open('people.txt', 'w') do |file|
  lines.sort {|a,b| a[1] <=> b[1]}.each do |array|
    puts array.join("\t")
  end
end
Closure-like Things in Java

```java
final String name = getName();
new Thread( new Runnable() {
    public void run() {
        doSomething(name);
    }
}).start();
```
Closures for Java 7?

- Proposal for closures in Java
  - [http://www.javac.info/closures-v03.html](http://www.javac.info/closures-v03.html)

```java
<T extends java.io.Closeable, throws E>
void closeAtEnd({T=>void}throws E block, T t) throws E {
    try {
        block.invoke();
    } finally {
        try {
            t.close();
        } catch (IOException ex) {}
    }
}

closeAtEnd(FileReader in : makeReader()) closeAtEnd(FileWriter out : makeWriter()) {
    // do something
}
```
Inheriting Behaviour from Multiple Places

- C++ has multiple inheritance
- Java has interfaces
- Ruby has mixins
C++: Multiple Inheritance
Java: Inheritance

Class A

Interface 1

Class B

Interface 1

Class C

Class A
Ruby: Mixins

Class A

Mixin 1

Class B

Mixin 2

Mixin 1

Class A
Mixins

```ruby
module SampleModule
  def do_something
    puts 'doing something'
  end
end

class SampleClass
  include SampleModule
end

SampleClass.new.do_something
```
Enumerable

- Requires that the class implement `each()`
  - For `max`, `min` and `sort` the `<=>` operator is also needed
- Adds many methods for modifying, searching, sorting the items
  - `all?`, `any?`, `collect`, `detect`, `each_cons`, `each_slice`, `each_with_index`, `entries`, `enum_cons`, `enum_slice`, `enum_with_index`, `find`, `find_all`, `grep`, `include?`, `inject`, `map`, `max`, `member?`, `min`, `partition`, `reject`, `select`, `sort`, `sort_by`, `to_a`, `to_set`, `zip`
Enumerable Examples

```ruby
people.find { |person| person.last_name == 'Smith' }
people.sort { |a, b| a.last_name <=> b.last_name }

lastNames = people.collect { |p| p.last_name }
```
A domain-specific programming language (domain-specific language, DSL) is a programming language designed to be useful for a specific set of tasks. – Wikipedia
DSL Samples

ANT

```xml
<target name="compile" depends="init" description="compile" >
    <!-- Compile the java code from ${src} into ${build} -->
    <javac srcdir="${src}" destdir="${build}"/>
</target>
```

Google Maps

pizza near keystone resort, keystone co
DSL Samples

```ruby
task :clobber => [:clean] do
  rm_r "tempdir"
end
```
DSL: ActiveRecord

class ListItem < ActiveRecord::Base
  belongs_to :amazon_item
  acts_as_taggable
  acts_as_list :scope => :user
end
Meta Programming

- attr_accessor
- has_many
- once_only
class Foo
  attr_accessor :bar
end

class Foo
  def bar
    @bar
  end
  def bar=(newBar)
    @bar = newBar
  end
end
Possible Implementation of attr accessor

class Foo
  def self.my_attr_accessor symbol
    name = symbol.id2name
    module_eval <<-DONE
      def #{name}()
        @#{name}
      end
      def #{name}=(newValue)
        @#{name} = newValue
      end
    DONE
  end
my_attr_accessor :bar
end
class ListItem < ActiveRecord::Base
  belongs_to :amazon_item
  acts_as_taggable
  acts_as_list :scope => :user
end
def once(*ids) # :nodoc:
  for id in ids
    module_eval <<-"end;", __FILE__, __LINE__
    alias_method :__#{id.to_i}__, :#{id.to_s}
    private :__#{id.to_i}__
    def #{id.to_s}(*args, &block)
      if defined? @__#{id.to_i}__
        @__#{id.to_i}__
      elsif ! self.frozen?
        @__#{id.to_i}__ ||= __#{id.to_i}__(*args, &block)
      else
        __#{id.to_i}__(*args, &block)
      end
    end
  end
end;
Continuations

Imagine...
Continuations

A snapshot of the call stack that the application can revert to at some point in the future
Continuation Web Server

class SampleServlet < ContinuationServlet
  def do_work
    for i in 1..35
      yield_now if i % 5 == 0
      @response.body << " : #{i}"
    end
    yield_now
    @response.body = 'done'
  end
end
class Sample < WEBrick::HTTPServlet::AbstractServlet
  def service(request, response)
    @response.body = "hello world"
  end
end

server = WEBrick::HTTPServer.new(:Port => 9005)
server.mount('/', Sample)
trap("INT") { server.shutdown }
server.start
class ContinuationException < Exception
end

class ContinuationServlet < WEBrick::HTTPServlet::AbstractServlet
  @@cc = nil
  def yield_now
    @request, @response = callcc do | cc |
      @@cc = cc
      raise ContinuationException.new
    end
  end
  def service(request, response)
    @@cc.call [request, response] unless @@cc.nil?

    @request, @response = request, response
    begin
      do_work()
      rescue ContinuationException => e
    end
  end
end
Continuation Web Server

class SampleServlet < ContinuationServlet
def do_work
  for i in 1..35
    yield_now if i % 5 == 0
    @response.body << " : #{i}"
  end
  yield_now
  @response.body = 'done'
end
end
Recap

- Learning a new language will make you better with all the languages you know
- Ruby has a much more concise syntax which means that it takes much less code to solve the same problems

Concepts
- Closures
- Domain specific languages
- Metaprogramming
- Continuations
Ruby on the Java™ Platform

- JRuby
- See Roberto Chinnici’s presentation on “Scripting in the Java™ Platform”
More Information

- Ruby homepage

- Ruby doc
  - Extensive documentation for all core and standard libraries
    - http://www.ruby-doc.org/

- Ruby Central
  - Home of the one-click Windows installer
    - http://www.rubycentral.com/
Contacting Me

- Mike Bowler
- mbowler@GargoyleSoftware.com
- I’ll be here all week
- Please fill out your evaluations