

IronRuby

Orion Edwards

Web Application Developer - Open2view.com

1

I'm Orion Edwards. I work for o2v.

The entire o2v site runs off Ruby on Rails, so the majority of my work is actually done in ruby. I do have a healthy bit of .NET client applications on the side however, and worked with a lot of .net (client apps and ASP) in previous jobs, which is what drives my interest in this group.

Because Ruby and .NET are my 2 major focuses at the moment, it's really neat to see the progress IronRuby is making, as it combines these 2 things together.

As this is the DNUG, I'm kind of assuming you all know how to code in either C# or VB.net, and that the majority of you won't know much about ruby. As such, this might come across as a sales pitch for ruby. I figure someone else already sold you .net so I'll stick to where my opportunities are :-)

What is IronRuby?

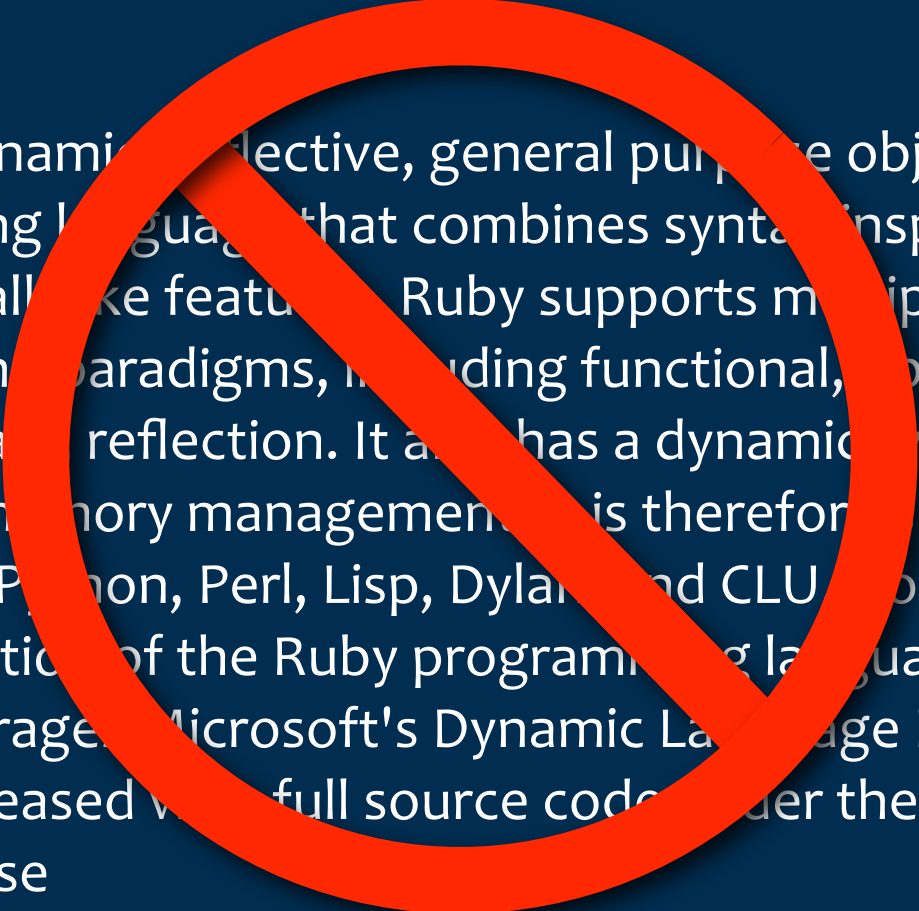
Ruby is a dynamic, reflective, general purpose object-oriented programming language that combines syntax inspired by Perl with Smalltalk-like features. Ruby supports multiple programming paradigms, including functional, object oriented, imperative and reflection. It also has a dynamic type system and automatic memory management; it is therefore similar in varying respects to Python, Perl, Lisp, Dylan, and CLU. IronRuby is a .NET implementation of the Ruby programming language. IronRuby heavily leverages Microsoft's Dynamic Language Runtime, and both are released with full source code under the Microsoft Public License

-- Wikipedia

Here's the wikipedia definition.

It's one of those descriptions that makes perfect sense to the people that already understand it, but if you don't, it's no use to you. So let's take a more practical look at it.

What is IronRuby?



Ruby is a dynamic, reflective, general purpose object-oriented programming language that combines syntax inspired by Perl with Smalltalk-like features. Ruby supports multiple programming paradigms, including functional, object oriented, imperative and reflection. It also has a dynamic type system and automatic memory management. It is therefore similar in varying respects to Perl, Python, Perl, Lisp, Dylan and CLU. IronRuby is a .NET implementation of the Ruby programming language. IronRuby heavily leverages Microsoft's Dynamic Language Runtime, and both are released with full source code under the Microsoft Public License

-- Wikipedia

Here's the wikipedia definition.

It's one of those descriptions that makes perfect sense to the people that already understand it, but if you don't, it's no use to you. So let's take a more practical look at it.

What is IronRuby?

Microsoft's implementation of the Ruby Programming Language using the .NET framework

IronRuby

4

I'm working under the assumption that most of you know nothing about ruby. For IronRuby to make any sense, we need to understand ruby itself, so I'll start by explaining a bit about ruby

Ruby

4

I'm working under the assumption that most of you know nothing about ruby. For IronRuby to make any sense, we need to understand ruby itself, so I'll start by explaining a bit about ruby

So why is Microsoft chosing to implement ruby on .NET – why not IronPHP, or IronCOBOL?

Why Choose Ruby?

So why is Microsoft choosing to implement ruby on .NET – why not IronPHP, or IronCOBOL?



Ruby is cool

Ruby is pretty trendy these days. All the cool kids are using it, but why?



Has anyone not heard of rails?, rails is a web app framework written in ruby

MASSIVE hype around rails. Developers from Java, PHP etc switching in all over the place. Rails brings a lot of cool concepts to the table, probably the main thing being the MVC web development model. MVC has been around for like 30 years, but rails was the first piece of software to bring it to the web.

So then, A lot of people looked at rails, and saw some of the cool things that it did, and thought 'Hey I can do that in .NET or in PHP or whatever'.

Attack of the Clones

- .NET

MonoRail, ASP.NET MVC

- PHP

CakePHP, Symfony

- Java

Grails, Trails

- More being written every day

We end up with loads of new web frameworks all copying bits and pieces of the rails stack.

They all use MVC and many borrow a lot of the other ideas like the ActiveRecord database access pattern, etc.

This gets you a long way, but as it turns out, none of these things end up being quite as good (from a development POV) as rails itself. Turns out there's more to rails than just a bunch of libraries which you can port.

I came for the Rails, but I stayed for the Ruby

Ezra Zygmuntowicz - Founder of EngineYard.com

9

A lot of what makes developing a rails app so good (and builds all the hype), comes down to the ruby language itself.

Having realised this, a lot of people wanted to use not only the cool stuff from rails, but the cool stuff from ruby itself, but they still needed to interoperate with all their existing stuff, and they did that by porting the Ruby language itself to run on-top of their current platforms.

All aboard the ruby bus

- MRI - “The Normal” Ruby
- JRuby - Sun
- MacRuby - Apple
- IronRuby - Microsoft
- Rubinius - EngineYard
- MagLev - Gemstone
- HotRuby

10

Here's a list of the different implementations of ruby right now.

- MRI stands for ‘Matz’s Ruby Interpreter’ – Matz is yukihiro matsumoto, a japanese guy who first created the ruby language back in 1993. It’s written in straight C. This what you get when you go to the official ruby website.
- JRuby is an implementation of Ruby running on the JVM – it interops natively with java. Commercially backed by sun.
- MacRuby is Ruby running on Apple’s Objective C runtime. It interops natively with ObjectiveC, which makes it ideal for writing OSX GUI applications. It’s in it’s early stages, commercially backed by apple.
- IronRuby is microsoft’s version. Commercially backed by microsoft who employ full time staff working on it. This is what our presentation is about
- Rubinius is a an implementation of Ruby running using a C++ Virtual machine. Commercially backed by engineyard, one of the biggest ruby on rails web hosting companies.
- Maglev is by GemStone, who are a smalltalk company. They’ve been around for years and years writing big enterprise systems using smalltalk. Maglev is their project to run ruby on their smalltalk VM
- Hotruby is a small open source thing which compiles ruby into javascript. Sounds ridiculous but people have run this using the V8 engine in Google Chrome and it actually works out to be faster than the other ruby implementations for whatever set of benchmarks they were running

All these companies are putting some serious resources behind getting ruby to run on the systems they want. This is evidence that there must be SOMETHING to it.

Philosophy

Instead of emphasising the what, I want to emphasise the how part: how we feel while programming. That's Ruby's main difference from other language designs

-- Yukihiro Matsumoto - Creator of Ruby

Turing completeness theorem states that any turing complete language can do the same work that any other language can, so why focus on what a language can do? Ruby has a ton of features which make it easier for people to read and write the code, which at the end of the day saves time, and lets you have more fun while you're at it.

This means...

Write less code, which is more
readable, and does more



Great Success!

How does Ruby Do It?

```
public class Person {  
    private string m_name;  
  
    public Person(string name){  
        m_name = name;  
    }  
  
    public string Name {  
        get{ return m_name; }  
        set{ m_name = value; }  
    }  
  
    public bool IsCool() {  
        if( m_name.Contains( "0" ) )  
            return true;  
        else  
            return false;  
    }  
}
```

First, here's some C# code defining a person class. It has a constructor, getter and setter for the Name, and a method which tells you if the person is cool or not.

Pretty basic stuff.

```
class Person
  def initialize(name)
    @name = name
  end

  attr_accessor :name

  def cool?
    @name.include? '0'
  end
end
```

16

Here's the same class written in ruby. It looks similar, but there's a few differences

NO SEMICOLONS OR CURLY BRACES

instead of `m_name`, we use `@name`, and we don't need to declare it anywhere. We just set the variable and go.

The constructor is called `initialize` instead of the class name

Ruby lets us use question marks as part of method names, so we can change `IsCool` to `cool?` which makes it clearer.

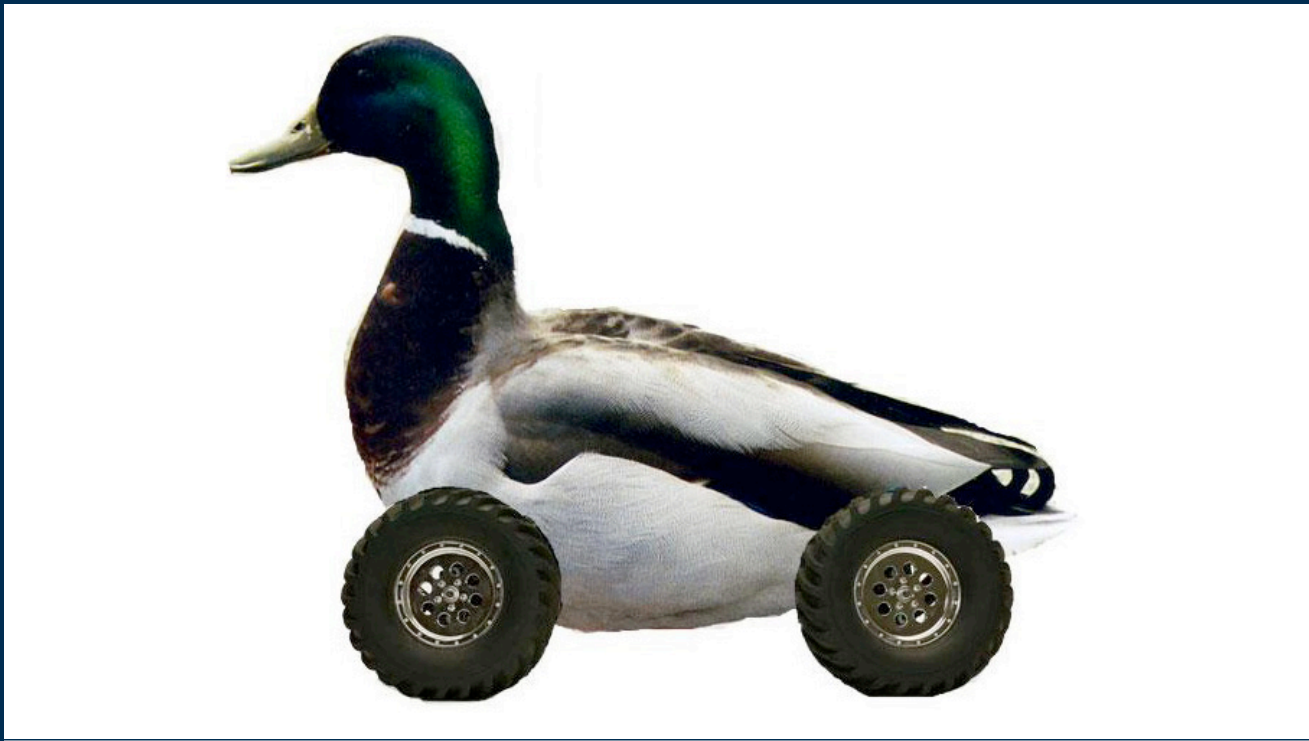
our `cool` function doesn't have `'return'` anywhere. Ruby just returns whatever the last line of code is, which is the check for the string include. We could type `'return'` there if we wanted to, and it's fine, but we don't need it

All the boilerplate of the standard getter and setter has been replaced with `attr_accessor`, which saves us some typing

Overall, there's quite a bit less code. Ruby has quite a few tricks up it's sleeve to cut down the amount of code you need to write. Each trick doesn't count for much, but when you add them all together, it pays off

How? Explained

- Duck Typing
- Literal Hash/Array syntax
- Flexible Syntax
- Object Oriented
- Interactive Shell
- Dynamic



Duck Typing?

What the heck is duck typing? Why do I want a duck in my code?

Duck Typing

```
public int Add(IComparable c, IComparable d) {  
    return c.compareTo(d);  
}
```

VS

```
def add( c, d )  
    c.compare_to(d)  
end
```

19

If it looks like a duck, and quacks like a duck, it might as well be a duck!
We don't need to type IComparable or anything like that. If c has a compare_to method, which accepts an argument, then it will work. If not, it won't

AUDIENCE QUESTION: Will that break things like intellisense?

A: Sort of. It prevents intellisense from working in the same way it does for C#, but there are some other things you can do. I think netbeans or eclipse or one of those manages to provide intellisense for ruby and it looked like it worked pretty well.

Hash/Array Syntax

```
IDictionary<string, int> people =  
    new Dictionary<string, int>() {  
        { "Orion", 26 },  
        { "John", 32 },  
    };
```

```
IList<string> names = new[]{ "Orion", "John" };
```

VS

```
people = {"Orion" => 26, "John" => 32}
```

```
names = ["Orion", "John"]
```

20

We don't have to babysit the compiler and tell it that our list contains strings. We can stick any old thing in the list, and away we go.

What if I stick bad data in the list? This is not any worse than a NULL object in the list. You still need TESTS

Flexible Syntax

```
10.times do  
  puts "I am cool"  
end
```

```
raise "Only digits allowed!" unless phone_number =~ /^[0-9]+$/
```

```
@car.should have_at_least(4).wheels
```

Here's 3 separate examples of proper working ruby code.

The first one is hopefully pretty obvious.

If you're not familiar with regular expressions the phone_number one is just a regex which checks if a string consists only of the numbers zero to 9.

The third one is an actual line from a unit test I wrote earlier. This is the actual code that runs.

The thing that I think is really nice is that even if you've never seen a line of ruby code before in your life, I think it's pretty easy to see what the code does. That's the really big benefit, that you can get away from brackets and IThis and IThat and write code which does what it says it does

Properly Object Oriented

```
nil.nil? # returns true
```

```
list << SomeClass  
# same as SomeClass.new  
obj = list.first.new
```

```
obj.class # returns SomeClass
```

Inheritance even works for static methods!

Even NULL is an object, and can have it's own methods.

You can chuck a class around like you do any other object

Interactive Shell

23

You can run 'irb' and just type some ruby code, and it runs as you go. This is a killer feature for when you're testing, or when you're learning a new API. Whenever I code in .net I always miss this

<DEMO OF LIVE IRB>

Dynamic

- Add new methods to existing classes
- Define new classes as needed
- Overwrite existing methods and classes
- Handle methods/classes which don't exist
- All on the fly, while your program is running

24

.NET: Dead code. Build it, give it a push, and watch the fireworks

Ruby: **THERE IS NO COMPILER. THE RUNTIME IS THE COMPILER.**

You can always create or modify code on the fly.

You can use this to work around bugs – for example there's a project that was getting affected by a bug in one of the core ruby libraries – they fixed the bug in their own code and used it to overwrite the built in broken code, until such time as the bug got fixed later on in the core ruby distribution.

You could also use this to redefine the built in classes like number and string and make it so $1 + 1 == 3$, but why would you do that? If someone is dumb enough to break the built in core code, the solution is to take them out in the parking lot and give them a swift kick up the backside so they don't do that any more :-)

Dynamic

```
class Booking < ActiveRecord::Base
  validates_presence_of :due_date
  has_many :attendees
end
```

```
Booking.find_all_by_due_date(Time.today,
  :include => :attendees)
```



Unfortunately...

Handicap: Ruby

- Terrible Performance
- Fake threads
- Hard to deploy as source-code-only
- No decent GUI Libraries
- These are all problems with the MRI runtime, not with the language itself. IronRuby can solve them

IronRuby

So now that we've seen what the ruby language gives us, lets have a look at the Iron part, and bringing it to .NET

Iron

So now that we've seen what the ruby language gives us, lets have a look at the Iron part, and bringing it to .NET

Integrating Ruby and .NET

- Can do everything that normal Ruby does.
- Can do everything that a normal CLR language does.
- Talk to any .NET code, and .NET can talk back

Ruby ON .NET

```
include System::Windows::Forms
```

```
f = Form.new
```

```
dialog_result = f.show_dialog
```

Here's an example of some ruby code accessing the built in windows forms classes to show a dialog. It's pretty much what you'd do from C#, with fewer brackets. Anything you can access from C# or VB you can access from ruby and use all the same frameworks and dlls etc

How'd they do that?

31

The DLR is a library which MS have written. It's basically a backend for writing dynamic languages on the CLR. The ironruby frontend reads the ruby code, and generates an abstract syntax tree which is basically like an intermediate representation of the code, then hands it to the DLR.

The DLR then converts it to native CLR bytecode, and handles interop and stuff so in theory it can be just as fast as C# code which has been compiled normally.

MS are putting some serious work into it, a lot of languages will be using it, and it'll ship with the next full release of the CLR, so even if you never use ironruby, you'll definitely run into the DLR sooner or later.

How'd they do that?

- The Dynamic Language Runtime!

31

The DLR is a library which MS have written. It's basically a backend for writing dynamic languages on the CLR. The ironruby frontend reads the ruby code, and generates an abstract syntax tree which is basically like an intermediate representation of the code, then hands it to the DLR.

The DLR then converts it to native CLR bytecode, and handles interop and stuff so in theory it can be just as fast as C# code which has been compiled normally.

MS are putting some serious work into it, a lot of languages will be using it, and it'll ship with the next full release of the CLR, so even if you never use ironruby, you'll definitely run into the DLR sooner or later.

How'd they do that?

- The Dynamic Language Runtime!
- Also making appearances in:
 - IronPython
 - JScript.NET
 - VB 10
 - C# 4 (maybe)
 - Silverlight

31

The DLR is a library which MS have written. It's basically a backend for writing dynamic languages on the CLR. The ironruby frontend reads the ruby code, and generates an abstract syntax tree which is basically like an intermediate representation of the code, then hands it to the DLR.

The DLR then converts it to native CLR bytecode, and handles interop and stuff so in theory it can be just as fast as C# code which has been compiled normally.

MS are putting some serious work into it, a lot of languages will be using it, and it'll ship with the next full release of the CLR, so even if you never use ironruby, you'll definitely run into the DLR sooner or later.

Ruby IN .NET

32

So that's ruby talking to .NET.

You can also embed ruby into existing pre-built .NET applications. The DLR has a hosting API which is pretty nice, and it's only about ten lines of code to get it up and running.

<DEMO OF EMBEDDED IRONRUBY CONSOLE, USE IT TO RESIZE AND SHIFT THE FORMS, CHANGE BGCOLOR, ETC>



Happy Rubying!

So that's it! If you'd like to play around with any of it you can get IronRuby from ironruby.net. It has all the instructions you need there, or feel free to email me etc.

Thanks!