RubyConf 2009

# IRONRUBY: SCRIPT THE WORLD

40 minutes

IronRuby – a fast open-source Ruby implementation – is designed to script the world. IronRuby runs on most major operating systems, as well as most modern browsers as a replacement for JavaScript (as browsers are becoming platforms in their own right). It can also reach across the programming language barrier to use code written in other languages (static or dynamic), and allows those other languages to embed IronRuby.

This talk will give the 40-minute-tour of IronRuby; where it runs, what it can be used for, and how it works.

|  |  |
| --- | --- |
| 5min | Introduction, talk summary, and agenda description |
| 5min | Overview of IronRuby and .NET ecosystem |
| 10min | IronRuby is cross-platform: show running on Mac OS and/or Linux |
| 10min | IronRuby is cross-language: show interop with Python and C#  (Include here a quick DynamicObject overview – how all this works) |
| 10min | Project status and IronRuby ecosystem overview (RC announcement, compatibility, performance, cool projects, etc) |

This talk will show IronRuby performing on Mac OS, Linux, and Windows, inside and outside the browser, using a collection of technologies like Mono, Silverlight, and Moonlight, and discusses the possibilities this flexibility offers. Being a good cross-language citizen as well, IronRuby will show its friendliness with Python and C# code, both as consumers and being embedded in other languages. The new version of C# has special support for doing dynamic-method dispatch at the syntax-level, which will show how seamless static and dynamic languages can interact. All the language-interop is made possible by the Dynamic Language Runtime.

For the language implementers, this talk will also dive into the language design of how IronRuby integrates with the world, and how the language implementation makes IronRuby fast.

Last but not least, we’ll discuss the status and roadmap of the project, including an overview of the performance and compatibility compared with other Ruby implementations, as well as the every-growing ecosystem.

## IronRuby overview

### Project Goals

* Ruby 1.8.6 compatible implementation, except continuations, object space, etc (though that’s changing!)
* Great integration with the .NET framework while keeping true to the language.
  + .NET 2.0 SP1 and above
  + Easy hosting
* Be a good Ruby community citizen – open source, contribute tests back to RubySpec.

## Cross-platform

### Use Cool .NET API from IronRuby on Linux

#### Why?

.NET’s vast amount of functionality that is now available to Ruby developers on major platforms.

#### What?

Demo needs to be something that can’t be done with MRI, and would relate well to Ruby devs. Possibilities:

* Definitely showing Gestalt / Ruby script tags
* Windows Deployment (can deploy without IronRuby, so not a big-deal-demo)
* Mono-specific features (GTK#)
* **Anything else?**

#### How?

At a minimum, use the latest Mono VM. Ideally get this all installed in Mac OS.

#### Story

While there are ton of scenarios that IronRuby provides to Ruby developers that other implementations don’t provide today (or it’s not their sweat spot), the point is IronRuby doesn’t limit you to Windows. Running IronRuby on Mono gives you access to all the mono APIs (GTK#), as well as having choice for your stack. Like, Rails runs on Mono w/IronRuby.

## Cross-language

### Using Ruby for extensibility

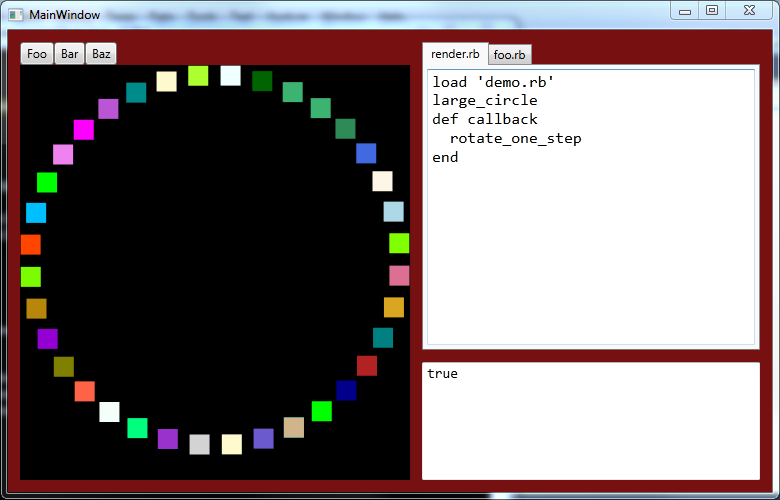
#### Why?

As much as we don't like it, not everyone uses Ruby to build their entire app. Imagine if you could use code written in other languages from Ruby, and your Ruby code could be used by other languages. This allows you to extend existing applications with Ruby, possibly to customize the app for a customer, or to give end-users extensibility.

#### What?

Demo of a pre-build C# application that has scripting capabilities added to it, allowing simple extension of the base application. Have a bunch of pre-written scripts that extend the functionality of the application.

A very simple game engine/IDE – use the scripts to write the games, as well as extend the UI. Here’s an early screenshot:



All the UI is to begin with is a canvas, and two textboxes – one that executes code and one that outputs the results. Drawing/Animating the canvas and adding buttons to the UI are done with scripts.

#### How?

Show base app in VS2010. Add scripting support with pure C# 3 APIs. Show various scripts that do cool and invasive things to the app. Then show C# 4 features and how C# integrates with Ruby at the language level. Then spice it up and call Python from Ruby for some reason =P

#### Story

Unfortunately not everyone uses Ruby, but wouldn’t it be great to use Ruby even if the app isn’t written in it? With IronRuby you can blur those lines and use Ruby from any language running on .NET or Mono.

Also, good to mention that Ruby and other dynamic languages are influencing the language design of other languages like C# … even to the point where calling a Ruby method is just like calling a C# method, so that deserves props to the Ruby community.

Where does Python fit it? What python library can I use? Is just showing that Python can be used as well good enough? I really think the Python integration should be shown: scope.python\_class.python\_method.

After Python is demoed, it might be useful to show a couple slides talking about DynamicObject, and how you can use C# to provide a dynamic object model to scripts (polyglot message).

## Project status

### Roadmap

The first lines of IronRuby code were written in January 2007, and by March 2007 IronRuby was shown running in the browser as part of the scripting story for Silverlight. IronRuby first showed up on RubyForge as an open-source project at that time, and the first binary release was in September 2007 (0.1). Since then there have been a bunch of quick releases, and today I’m proud to announce the **IronRuby 1.0 Release Candidate**. We’re more than accomplishing the goals we set out to reach for 1.0, and fully expect to exceed them more for the final 1.0 release.

#### 1.0 goals

* Performance: ~2x faster than MRI RubyInstaller builds, ~6x faster than MRI One-Click install.
  + 1.5x slower than JRuby, but expect to get much closer for 1.0
* Compatibility: **xx%** RubySpec pass rate (xx% language, xx% library, xx% built-ins)
* .NET Integration
  + Missing a few features here, but they all have work-arounds
* **Anything else to call out here?**

#### Roadmap to 1.0

* Today: IronRuby 1.0 RC1
* Will do RC2, 3, etc monthly until no must-fix-for-RC bugs are accepted for a full month, and then bless that release as IronRuby 1.0.
  + Estimated Q1 2010
* Work still left for 1.0 (in priority order):
  + Startup Perf – largest factor in “slowness” perception. Get RubyGems, Rake, and Rails startup numbers on par with JRuby. To accomplish this, we’re improving the interpreter.
  + Throughput Perf – very much related to startup, but mainly fixing issues in libraries.
  + Bugs – We have 50 open bugs, and I presume more to come with this RC announcement.

#### After 1.0

* Post-1.0 releases planned very soon after (month or 2 after, sticking to a 2 month cycle)
* Visual Studio Integration – largest requested VS2010 feature by users!

### Ecosystem

The IronRuby ecosystem has been pretty healthy ever since the beginning of this year. We have a core set of community members who are using IronRuby in their day-jobs as well as just for fun. However, the .NET crowd are not predominantly “early-adopters”; in fact, the usually wait years after a release to use it (people still using VS 2005).

#### ironruby-contrib

The community has really helped fill in various Ruby libraries for IronRuby:

* Cucumber on .NET with IronRuby
* Caricature – Mocking static types with IronRuby
* IronRack – rack-based web apps on IIS
* ironruby-dbi – port of dbi – enables current ActiveRecord/DataMapper SQLServer/SQLite# to just work with IronRuby
* IronMVC – IronRuby support for ASP.NET MVC applications
* Gestalt – replacing JavaScript with Ruby (now merged into IronRuby)
* Magic – a UI framework for Ruby
* **Anything else?**